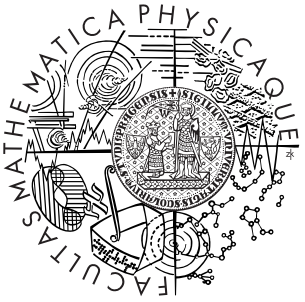
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jakub Kinšt

Systém pro podporu výuky

Ústav formální a aplikované lingvistiky, MFF UK

Vedoucí bakalářské práce: Mgr. Miroslav Týnovský

Studijní program: Informatika

Studijní obor: Obecná Informatika

Praha 2012

TODO: Zde bude poděkování

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne podpis

Název práce: Systém pro podporu výuky
Autor: Jakub Kinšt
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí bakalářské práce: Mgr. Miroslav Týnovský
e-mail vedoucího: tynovsky@ufal.mff.cuni.cz

Abstrakt: V předložené práci studujeme ... Uvede se abstrakt v rozsahu 80 až 200 slov. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut sit amet sem. Mauris nec turpis ac sem mollis pretium. Suspendisse neque massa, suscipit id, dictum in, porta at, quam. Nunc suscipit, pede vel elementum pretium, nisl urna sodales velit, sit amet auctor elit quam id tellus. Nullam sollicitudin. Donec hendrerit. Aliquam ac nibh. Vivamus mi. Sed felis. Proin pretium elit in neque. Pellentesque at turpis. Maecenas convallis. Vestibulum id lectus. Fusce dictum augue ut nibh. Etiam non urna nec mi mattis volutpat. Curabitur in tortor at magna nonummy gravida. Mauris turpis quam, volutpat quis, porttitor ut, condimentum sit amet, felis.

Klíčová slova:

Title: Learning management system
Author: Jakub Kinšt
Department: Institute of Formal and Applied Linguistics
Supervisor: Mgr. Miroslav Týnovský
Supervisor's e-mail address: tynovsky@ufal.mff.cuni.cz

Abstract: In the present work we study ... Uvede se anglický abstrakt v rozsahu 80 aľ 200 slov. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut sit amet sem. Mauris nec turpis ac sem mollis pretium. Suspendisse neque massa, suscipit id, dictum in, porta at, quam. Nunc suscipit, pede vel elementum pretium, nisl urna sodales velit, sit amet auctor elit quam id tellus. Nullam sollicitudin. Donec hendrerit. Aliquam ac nibh. Vivamus mi. Sed felis. Proin pretium elit in neque. Pellentesque at turpis. Maecenas convallis. Vestibulum id lectus. Fusce dictum augue ut nibh. Etiam non urna nec mi mattis volutpat. Curabitur in tortor at magna nonummy gravida. Mauris turpis quam, volutpat quis, porttitor ut, condimentum sit amet, felis.

Keywords:

Obsah

Kapitola 1

Úvod

1.1 Motivace

Profesoři, učitelé, cvičící a lektoři kurzů se často potýkají s problémem, jak své studenty spolehlivě informovat o průběhu výuky, jak jim poskytnout studijní materiál a jiné doplňující informace. Rádi by také čas od času studenty rychle a hlavně jednoduše vyzkoušeli z probrané látky. Studenti chtějí mít k dispozici způsob, kde mohou lektorovi nebo samotnému učiteli položit otázku a případně na dané téma diskutovat. Chtějí dostávat aktuální informace o blížících se termínech nebo testech.

Řešením je pro některé lektory nepraktické rozesílání aktuálních informací pomocí hromadných e-mailů, které v poštovní schránce rychle upadnou v zapomnění. Pokud vznese student dotaz, dostane se odpovědi jen jemu a ostatní se o ní vůbec nedozvědí.

Iniciativnější lektoři s nemalou trochou úsilí vytvoří jednoduchý web, který se snaží pravidelně aktualizovat a přidávat aktuality. Často ale stejně chybí možnost diskuze, nahrávání materiálů, správa událostí a online vypracovávání testů a úkolů. Přestože lze některé z těchto vlastností nahradit externími webovými službami, bylo by dobré mít po ruce komplexní a zároveň uživatelsky jednoduché řešení, které všechny zmíněné problémy řeší.

Zároveň, v době chytrých telefonů a pohodlnosti mobilního internetu, by se hodilo, aby měl uživatel veškerý obsah na dosah i ve svém mobilním telefonu (nebo v jiném mobilním zařízení) a mohl si tak číst aktuality na cestách nebo vyplňovat test po cestě do školy ve vlaku.

Výsledkem práce je implementace kompletního univerzálního řešení webové služby zajišťující podporu pro standardní prezenční výuku. Služba ulehčí práci jak lektorům, tak i studentům libovolného kurzu. Umožní lektorovi jednoduše kontaktovat všechny studenty, informovat je o průběhu a obsahu jednotlivých

lekcí/přednášek/cvičení, oznámit dosavadní výsledky studentů, poskytnout jim libovolné elektronické učební materiály ke stažení, zadat jim elektronický test nebo úkol s různými typy otázek a informovat je o nadcházejících událostech či termínech. Studenti mohou diskutovat pod jednotlivými lekcemi/-přednáškami/cvičeními, diskutovat v diskuzním fóru na jakékoliv relevantní téma jak s ostatními studenty, tak i se samotným lektorem/lektory a můžou si mezi sebou posílat soukromé zprávy.

V případě rozšíření aplikace má navíc jak student, tak i lektor všechny studované (resp. vyučované) kurzy k dispozici pohromadě, na jednom místě a stejně jednoduše ovladatelné.

Většinu z těchto aktivit je také možné provozovat skrze nativní aplikaci pro mobilní platformu Google Android[1] z pohodlí svého mobilního telefonu nebo jiného mobilního zařízení (z tabletu, přehrávače apod.).

Součástí práce je také samostatná implementace univerzálního řešení propojení webové aplikace (postavené na Nette Frameworku[8]) s mobilní aplikací pro platformu Google Android.

1.2 Struktura práce

TODO

Kapitola 2

Existující implementace

Kapitola 3

Architektura MVC

Architektura *MVC (Model-View-Controller)[14]* se poslední dobou stává standardem při vývoji webových aplikací. Přináší totiž mnohé výhody a klady.

Architektura byla poprvé definována již v sedmdesátých letech minulého století společností *Smalltalk*. Rozděluje implementaci objektově orientované aplikace do tří základních vrstev: Model, View (Pohled), Controller (Řadič). Vrstvy v zásadě izolují ty součásti implementace, které spolu souvisí a musí o sobě navzájem vědět konkrétní charakteristiky.

Model je vrstva, která obvykle obstarává nebo udržuje informace a implementuje nějaký obecný problém nezávisle získávání vstupních dat nebo prezentování výsledků. Obvykle se jedná o jednu, nebo několik tříd, které například zajišťují data z databáze, provádí výpočty a jiné podobné úkony. Model by na ostatních vrstvách neměl být vůbec závislý.

View (Pohled), jak název napovídá je vrstva, která má za úkol zobrazovat výsledná data. Zobrazovat je nemusí jen uživateli, ale například také jinému programu (zobrazuje API). Zpravidla je pohled reprezentován definicí uživatelského rozhraní - HTML šablony, GUI[1] aplikace.

Controller (Řadič) je vrstva, o které lze říci, že leží mezi vrstvami Model a View. Stará se převážně o zpracování uživatelského vstupu a o propojení modelu s pohledem. Získává parametry ze vstupu, ty předá modelu, který na jejich základě vrátí potřebná data nebo učiní jiné úkony. Případná získaná data zase předá pohledu, který je vhodným způsobem předá například uživateli.

Výhod následování pravidel MVC je mnoho. Tou hlavní je nepochybně obrovská přehlednost výsledného kódu. Ta je způsobena striktním oddělením uživatelského rozhraní (View) od kódu obsluhy (Controller) a aplikační logiky (Model). Díky tomu jsou také aplikace daleko lépe připravené na případné

[1] Graphical User Interface

úpravy či rozšíření - například změna vzhledu uživatelského rozhraní proběhne pouze na úrovni vrstvy View. Výhodou je také to, že při testování je možné testovat jednotlivé vrstev naprosto odděleně.

Kapitola 4

Problematika propojení s mobilní aplikací

4.1 Motivace

V posledních letech je viditelný významný přesun práce s internetem z domácích počítačů na mobilní zařízení. Mobilními zařízeními jsou myšleny především mobilní telefony a tablety. Provozovatelé webových služeb jsou tak stále častěji nuceni umožnit klientovi užívat aplikaci právě z těchto zařízení.

Někteří řeší tento trend cestou nejmenšího odporu - optimalizací webových stránek aplikace pro menší obrazovky přenosných zařízení. Toto řešení spolu ale nese mnoho nevýhod. Aplikace především nemůže naplno využít potenciál zařízení. Neposkytují takový komfort jako aplikace integrované do systému, které mohou využívat kompletní sadu frameworků dodávaných výrobcem operačního systému. Nemohou například přistupovat k datům z jiných aplikací, jako je seznam kontaktů, kalendář a další. Nemohou ani naplno využívat některé hardwarové technologie zabudované v moderních zařízeních, jako jsou senzory polohy, světla, zrychlení, fotoaparát a podobně.

Toto řešení také kompletně znepřístupní obsah v případě nedostupnosti internetového připojení. Uživatel tedy může aplikaci používat pouze pokud je v tu chvíli připojen k internetu.

Jistou nadějí pro tento přístup může být technologie webových stránek v HTML5[15]. Ta přidává do tradičních HTML stránek možnost využít některé z těchto pokročilých funkcí. Je zde možnost obsah ke klientovi uložit a pak ho zobrazit offline - tedy krátkodobě bez připojení k internetu. Pomocí HTML5 elementů je také možné zjistit polohu uživatele pomocí zabudovaných senzorů. Další novinkou je element *canvas*, který umožňuje do okna prohlížeče vykreslovat libovolnou grafiku. HTML v této verzi již také nativně podporuje

přehrávání videa nebo audia - už není nutné vytvářet proprietární přehrávače například pomocí technologie Flash. Vylepšeny byly i standardní formuláře a práce s nimi.

Všechna tato vylepšení se zatím ukazují jako nedostatečná. Integrace s operačním systémem zařízení je minimální a v porovnání s nativními aplikacemi ty webové (HTML5) stále hodně zaostávají. Navíc je nutná podpora ze strany internetového prohlížeče, což se v praxi ukazuje jako velký problém. Různé prohlížeče interpretují jen různé podmnožiny pravidel HTML5.

Programování aplikace pro mobilní telefony pomocí technologie HTML5 ale přináší jednu velkou nespornou výhodu. HTML5 je totiž technologie multiplatformní. Provozovatelé služeb nemusejí platit vývoj pro všechny rozšířené mobilní platformy - Android, iOS a Windows Phone 7. Otázkou je, zda není tato výhoda zastíněna zmiňovanými nevýhodami.

Budoucnost pravděpodobně spěje k vývoji univerzálních multiplatformních aplikací, ale současná nabídka nástrojů k jeho docílení se zatím ukazuje jako nedostatečná.

4.2 Otevření API aplikace

Otevřít API[1] webové aplikace znamená zpřístupnit obsah aplikace jinému programu, tedy především navrhnout rozhraní, pomocí kterého bude možné jednak strojově přečíst data a také aplikaci poslat vstup - ovládat ji.

Pro řešení takového problému existuje mnoho odladěných a hojně používaných implementací, ze kterých lze vybírat. Výběr se však zužuje pokud se vezme v úvahu implementace webové aplikace v PHP a také to, že aplikace je již naprogramovaná a vývojář chce co nejvíce využít již implementované služby.

4.2.1 Architektura SOAP

Jedna z možností je využít architekturu pro webové služby s názvem SOAP[10]. Tato architektura využívá jako formát přenášených dat standard XML a k vlastnímu přenosu zpravidla HTTP nebo SMTP. Implementace je ale náročnější a v podstatě při ní nelze použít nic z implementace samotné webové služby.

[1] Application Programming Interface

4.2.2 Architektura REST

Druhou možností je využití architektury REST[17]. Tato architektura je na rozdíl od SOAP orientovaná datově a ne procedurálně. REST nedefinuje vzdálené volání procedur, ale rovnou říká jak přistupovat ke vzdáleným datům.

K přenosu je využit protokol HTTP (autor REST je spoluautorem protokolu HTTP). Každý datový zdroj je identifikovaný unikátním identifikátorem URI[2]. K získání a ke vkládání dat se požívají standardní metody HTTP GET a POST. REST definuje i využívání konkrétnějších metod HTTP jako DELETE, PUT, ale jelikož většina implementací HTTP klientů podporuje pouze metody GET a POST, využívá se v praxi k mazání a aktualizaci dat metoda POST. Jako formát přenášených dat je možné použít prakticky cokoliv - XML, text oddělený oddělovačem, nebo formát JSON.

V podstatě stejně ale funguje typická aplikace naprogramovaná v jazyce PHP. Data se získávají GET požadavkem ve formátu HTML, vstupní data se posílají přes formulář pomocí požadavku POST. Díky tomu se RESTová architektura nabízí jako nejvhodnější varianta při požadavku využití stávajícího kódu v co největší míře.

K otevření API tedy stačí aby server poznal, že klientem není člověk, ale program (konkrétně mobilní aplikace), a na každé podstránce aplikace místo obsahu zformátovaného do HTML zaslal obsah zformátovaný tak, aby byl strojově čitelný.

4.2.3 Formát přenosu dat

Zbývá tedy zvolit standard, pomocí kterého budou data ze serveru do mobilní aplikace přenášena a zároveň bude jejich obsah snadno strojově čitelný. Nabízejí se dva nejrozšířenější a nejvíce podporované standardy - XML a JSON. Výběr záleží na použitých technologiích, konkrétně na tom, na jaké úrovni je existující implementace pro práci s těmito formáty. Dále může rozhodovat to, zda je nějaká potřeba u přenášených dat definovat strukturu. Tento požadavek lépe splňuje formát XML (XML Schema, DTD a další).

Ukázka formátu XML:

```
<auto id="auto-1">
        <vin>128988129823</vin>
        <spz>2A29288</spz>
        <znacka>Skoda</znacka>
        <typ>Superb</typ>
        <rok-vyroby>2009</rok-vyroby>
```

[2]Uniform Resource Identifier

```
        <parametry>
                <motor>
                        <spotreba>9.2</spotreba>
                        <objem>3,0</objem>
                        <maxrychlost>210kmh</maxrychlost>
                </motor>
        </parametry>
        <porizovaci-cena>750000</porizovaci-cena>
        <pujcovne>1230</pujcovne>
        <!-- Poznamky k vozidlu + zname poruchy -->
        <poznamky>
                Potreba vymenit olej pri 20000km
                Technicke kontroly:
        </poznamky>
        <nahrada autoID="auto-2" />
</auto>
```

Ukázka stejných dat ve formátu JSON:

```
{
        "auto": {
                "id": "auto-1",
                "vin": "128988129823",
                "spz": "2A29288",
                "znacka": "Skoda",
                "typ": "Superb",
                "rok-vyroby": "2009",
                "parametry": {
                        "motor": {
                                "spotreba": "9.2",
                                "objem": "3,0",
                                "maxrychlost": "210kmh"
                        }
                },
                "porizovaci-cena": "750000",
                "pujcovne": "1230",
                "poznamky": "
                        Potreba vymenit olej pri 20000km
                        Technicke kontroly:
                ",
                "nahrada": {
                        "autoID": "auto-2"
```

```
                    }
            }
}
```

4.2.4 Odesílání dat na server

Odesílání dat na server nemusí v případě použití RESTové architektury projít prakticky žádnou změnou na straně serveru. Pokud vstup probíhá pomocí GET požadavku, nejedná se o nic zvláštního - pokud jsou parametry součástí URL, server si s nimi poradí stejně jako v případě normálního užívání aplikace v prohlížeči.

Většina vstupu ve webové aplikaci probíhá přes formuláře za pomoci POST požadavku. Úkolem klienta v mobilní aplikaci bude tedy nasimulovat odeslání formuláře. To ale znamená provést jednoduchý POST požadavek na URL, které je nastaveno jako parametr *action* konkrétního formuláře, a k požadavku přidat jednotlivé hodnoty položek formuláře jako POST argumenty. Může se jednat o primitivní datové typy, ale musí být možné přes POST požadavek poslat i libovolný soubor, tak jako to lze udělat v HTML formuláři.

Pokud se vše správně implementuje, server zpracuje data stejně jako kdyby přicházela ze standardního formuláře z HTML.

4.3 Přenos Cookies

Problém, který nastane například při zachovávání možnosti přihlášení uživatelů do aplikace, je nutnost přenášet mezi požadavky hodnoty HTTP hlaviček Cookies[12].

Cookie a *Set-Cookie* jsou standardní hlavičky HTTP, které mají za úkol udržovat stavové informace mezi serverem a konkrétním klientem. Server může v HTTP odpovědi zaslat klientovi hlavičku *Set-Cookie* s nějakou hodnotou a do své paměti si uloží libovolnou stavovou informaci. Pokud v budoucnosti (během platnosti Cookie) klient ke svému HTTP požadavku přidá hlavičku *Cookie* s přijatou hodnotou, server klienta podle hodnoty identifikuje a z paměti zjistí potřebnou stavovou informaci.

V praxi se tato technika používá například právě udržení informace o přihlášení uživatele. Když se uživatel přihlásí, server si o něm vytvoří záznam, a aby se uživatel nemusel přihlašovat v každém následujícím požadavku, odešle klientovi (prohlížeči) hlavičku *Set-Cookie* s (tajným) identifikátorem a s určitou dobou platnosti. Pokud klient potom (před vypršením platnosti) pošle požadavek kde hodnotou jedné z hlaviček *Cookie* bude i zmíněný identifiká-

tor, pozná server o jakého klienta se jedná a už po něm nebude požadovat opětovné přihlášení - uživatel je přihlášen po celou dobu platnosti Cookie.

Přenos Cookies porušuje jedno z pravidel typických webových služeb - aplikace by měla být bezstavová. V případě již existující implementace přihlašování uživatelů lze však udělat výjimku a Cookies k tomuto účelu využít.

Pokud tedy požadujeme maximální využití stávajícího kódu standardní webové aplikace, je potřeba v tomto ohledu simulovat chování webového prohlížeče a s Cookies nakládat stejně. Z odpovědí serveru je tedy potřeba sbírat hlavičky *Set-Cookie*, kontrolovat jejich platnost a v následných dotazech na server posílat hlavičky *Cookie* se správným obsahem.

4.4 Zabezpečení API

Otevřené API aplikace může představovat určité riziko. Může se stát, že je potřeba do mobilní aplikace posílat data, která by jindy byla uživateli nepřístupná, nebo prostě nechceme, aby bylo API aplikace veřejně přístupné. V tuto chvíli přichází otázka zabezpečení API.

Aby si byl server jist tím, že komunikuje s oprávněným klientem, je třeba, aby se prokázal tzv. API klíčem. API klíč je tajný řetězec dostatečné délky a složitosti, kterým se klient při požadavcích na server prokáže, že je k získání dat oprávněn.

API klíč typicky nemůže být nahrazen přihlášením uživatele, jelikož je často potřeba přístup k API omezit i v případě veřejných částí aplikace s povoleným anonymním přístupem.

Klíč je možné přenášet v každém HTTP požadavku, nebo jen v prvním a pak využít možnosti Cookies stejně jako při přihlašování. Může být v požadavcích předáván jako GET parametr, POST parametr nebo může být uložen v hlavičkách. Všechny varianty mají své výhody i nevýhody.

Samotné použití API klíče nezaručuje naprostou bezpečnost API. Při sledování síťového provozu je stejně jako při přihlašování uživatelů možné API klíč zjistit. Proto je doporučeno provozovat webovou aplikaci na zabezpečeném protokolu *HTTPS*, kde je přenos šifrován pomocí technologií *SSL*[3] nebo *TLS*[4].

[3] Secure Sockets Layer
[4] Transport Layer Security

Kapitola 5

Použité technologie a frameworky

5.1 Nette Framework

Nette Framework[8] je pokročilý framework pro jazyk PHP. Přestože je PHP původně jazykem skriptovacím, v moderní době v něm lze naplno využít většiny výhod objektově orientovaného programování (OOP). Nette Framework staví své základy právě na objektovém modelu a k tomu také směřuje implementaci k využívání architektury MVC.

Nette přináší PHP programátorům maximální usnadnění vývoje webových aplikací a umožňuje jim orientovat se při vývoji na funkčnost a ne na řešení základních bezpečnostních nebo syntaktických problémů. Snaží se potlačit mnohé známé nešvary spojené s jazykem PHP nebo se skriptovacími jazyky obecně. Nette Framework také naprosto redefinuje ladění programů v PHP - nabízí plnohodnotné ladící nástroje, jaké jsou dostupné u moderních programovacích jazyků a tím značně urychluje vývoj.

Framework je velmi kompaktní (komprimovaná velikost je okolo 250 KB) a má tu výhodu, že programátor není nikdy nucen využívat kompletní framework. Běžně lze pomocí několika řádků kódu využít například jen poskytované ladící nástroje.

Hlavní přednosti a výhody lze shrnout do několika bodů:

1. Zabezpečení

   Při použití Nette se programátor nemusí při vývoji starat o řešení jindy obvyklých bezpečnostních záležitostí jako je například ochrana před Cross-site scripting, session hijacking atd. Framework typické bezpečnostní díry automaticky eliminuje, nebo vůbec nedovolí jejich výskyt.

2. Nástroje na ladění kódu

   Nette knihovna s názvem *Debugger* (laděnka) nabízí mnoho nástrojů, kterými lze zpříjemnit a hlavně povýšit ladění kódu v PHP. Laděnka umí při případné chybě (oproti klasickému chybovému hlášení PHP) přesně ilustrovat místo chyby a dokonce vypíše i tzv. callstack - zásobník zobrazující zanoření spuštěných metod v době chyby. Chybová stránka poskytne i aktuální hodnoty relevantních proměnných. V podstatě nabízí všechny pokročilé funkce, které lze najít v moderních vývojových prostředí u klasických programovacích jazyků.

   Laděnka dokáže zobrazit fixní panel zobrazený na vrchu každé stránky, který dokáže vypisovat hodnoty libovolných proměnných, délku načtení stránky nebo množství spotřebované paměti. Pokud je potřeba chyby, nebo cokoliv jiného zapisovat do externího souboru, lze využít funkce pro výpis do logu.

3. Formuláře

   Framework přináší výrazné usnadnění práce s HTML formuláři. Usnadňuje jejich vytváření, zpracování odeslaných dat (včetně validace) a zároveň poskytují zabezpečení proti mnoha zranitelnostem.

   Struktura formulářů se vytváří v PHP kódu pomocí třídy *Form*, což za prvé ochrání aplikaci proti zneužití formuláře na straně HTML (aplikace ví, co formulář obsahuje a pozná, pokud byl změněn), a za druhé umožní validaci vstupních dat jak na straně serveru, tak na straně klienta (JavaScript). Nette obsahuje velmi dobře vypadající výchozí způsob vykreslování formuláře, ale v případě potřeby lze pomocí maker v šabloně vykreslit formulář naprosto libovolně.

   Příklad formuláře v Nette Frameworku:

```
protected function createComponentRegisterForm() {
        // funkce pro validaci pouziteho e-mailu
        function myValidator($item){
                $result = dibi::query('SELECT id FROM user WHERE
                return (count($result) === 0);
        }

        // novy formular
        $form=new AppForm;

        // nastaveni jazyka podle prekladace
        $form->setTranslator($this->translator);
```

```
        // pridani textoveho pole pro jmeno
        // spolecne s overenim vyplneni a pripadnou hlaskou
        $form->addText('firstname', 'First name:*')
                        ->addRule(Form::FILLED, 'Fill the firstna
        $form->addText('lastname', 'Last name:*')
                        ->addRule(Form::FILLED, 'Fill the lastnam

        // Nette umi rovnou kontrolovat e-mailovou adesu
        // Nastavime vlastni validator obsahu - libovolnou funkc
        $form->addText('email', 'E-mail:*')
                        ->setRequired()
                        ->setEmptyValue('@')
                        ->addRule(Form::EMAIL, 'Enter valid e-ma
                        ->addRule('myValidator', 'E-mail is alrea
        $form->addPassword('password', 'Password:*')
                        ->addRule(Form::FILLED, 'Fill in the pass
                        ->addRule(Form::MIN_LENGTH, 'Minimal pass
        $form->addPassword('password2', 'Verify password:*')
                        ->addRule(Form::FILLED, 'Fill in the pass
                        ->addRule(Form::EQUAL, 'Passwords don\'t
        $form->addText('web', 'Webpage:');

        // pridame tlacitko k odeslani
        $form->addSubmit('send', 'Register');

        // nastavime callback, ktery se zavola po uspesnem odesla
        // spolecne s jeho hodnotami
        $form->onSubmit[] = callback($this, 'registerFormSubmitte
        return $form;
    }
```

4. Komponenty

   Rozdělení aplikací do samostatných komponent se hodí tehdy, je-li potřeba určitou část kódu použít vícekrát na různých místech. Kupříkladu lze do komponenty zapouzdřit přihlašovací formulář a v místech použití vždy vykreslit pouze komponentu s několika málo parametry. Komponenta funguje v podstatě velmi podobně jako Presenter (třídy mají stejného předka) a také je její vykreslení definováno externí šablonou.

5. Tvar URL (směrování)

Routování je proces překládání URL na konkrétní akci presenteru a naopak. Nette aplikace musí definovat tzv. Routy, které říkají, jakým způsobem překlad probíha. Základní routování funguje tak, že URL vždy obsahuje cestu k souboru *index.php* a všechny potřebné informace (název presenteru, název akce a jiné volitelné parametry) jsou předávány způsobem typickým pro GET požadavek. URL tedy může vypadat například takto: *http://example.com/index.php?presenter=product&action=detail&id=123*. Takové URL jsou ale v praxi zbytečně dlouhé a hůře čitelné. Proto lze v Nette jednoduše pomocí definování Rout zprovoznit takzvané „hezké URL". Adresa pak může vypadat třeba takto: *http://example.com/product/detail/123*.

Obrovskou výhodou je to, že při psaní aplikace programátor odkazuje na konkrétní akce presenterů a routování vůbec neřeší. Tedy až na konci vývoje se může rozhodnout pro styl routování a může ho kdykoliv změnit bez zásahu do aplikačního kódu.

6. Šablonový systém

   Pro zachování principů MVC architektury je nutné oddělit vzhled uživatelského rozhraní od aplikační logiky. K tomu v nette slouží systém tzv. *latte* šablon. Šablony jsou soubory s pčíponou *.latte*, které obsahují HTML kód doplněný tzv. *latte* makry. Makra jsou označena tím, že je „obalují" složené závorky. Nette v základu definuje mnoho standardních maker - od těch jednoduchých (prostý výpis proměnné, makro pro vytvoření URL odkazu a další) až po složitější (for-cykly, přiřazení proměnné, if podmínky, vykreslení formuláře a další). Proměnné se do šablony dostanou typicky tak, že je do ní pošle presenter.

   Šablony mohou samozřejmě do svého těla vkládat obsah jiných šablon, takže není potřeba kód duplikovat.

   Příklad použití šablony:

```
<html>
        <head>
                <title>Titulek stranky</title>
                <script src="{$basePath}/script.js" />
        </head>
        <body>
                <span>{$user->isLogged() ? $user->name : 'neprihl
                <ul id="menu">
                        {foreach $menuItems as $item}
                                <li><a href="{link $menu->link}">
                        {/foreach}
```

```
                </ul>
                {include content}
        </body>
</html>
```

7. Rozšíření (Pluginy)

   Komunita programátorů je velmi aktivní a na oficiálním webu je k dispozici mnoho doplňků, které Nette rozšiřují a často mohou ušetřit při vývoji aplikací mnoho času a práce. Většinou se jedná o zapouzdřené komponenty, které stačí zkopírovat do adresáře aplikace a pomocí několika řádku kódu rychle implementovat.

8. Architektura MVC

   Aplikace postavená na Nette frameworku by měla respektovat zásady architektury MVC - aplikace se totiž skládá zpravidla z presenterů (třída Presenter reprezentuje v architektuře vrstvu Controller), z modelových tříd (libovolná třída s veřejnými metodami) a z *latte* šablon (view).

9. Automatické načítání souborů tříd

   Každý PHP programátor se zkušeností s objektově orientovaným programováním má zkušenost s otravným nahráváním souborů tříd do kódu. Vždy, když je využita nějaká třída, musí být v souboru ručně naimportován soubor třídu obsahující. S tím je při použití Nette frameworku konec. Nette automaticky načítá (jen potřebné) soubory obsahující definice tříd z adresáře projektu a již není nutné nahrávat je ruřně pomocí příkazů *require*, *require_once* nebo *include*.

5.2 Google Android SDK

Google Android je relativně nový (2007) operační systém určený pro mobilní zařízení - zejména mobilní telefony, dnes i tablety. Za dobu své existence prošla platforma řadou inovací a změn - již je ve verzi 4.0.4. Android SDK[1] je balík nástrojů, kterým společnost Google usnadňuje (umožňuje) vývoj nativních aplikací pro tuto platformu. SDK je naprogramované v jazyce Java, tudíž i vývoj aplikací probíhá primárně v Javě. Android obsahuje většinu standardních knihoven Javy SE (Standard Edition), z nichž některé jsou podle potřeb upravené a dále knihovny vlastní. Knihovny nabízí mnoho nástrojů, které mají za cíl usnadnit vývojáři práci se zařízením a jeho funkcemi.

[1] Software Development Kit

SDK má ale i další součásti - nástroje na ladění aplikací, emulátor Android zařízení na případné testování, zásuvný modul do vývojového prostředí Eclipse a další.

5.2.1 AndroidManifest.xml

Základním kamenem aplikace je soubor *AndroidManifest.xml*, který slouží k tomu aby dokázal aplikaci popsat ještě před tím, než je spuštěna - aby systém věděl, co může od aplikace očekávat a co bude vyžadovat. Jde o jakousi deklaraci aplikace. Pomocí formátu XML jsou v souboru popsány základní informace o aplikaci - jaký má název, ikonu, titulek, minimální verzi Android OS, verze aplikace a další. V manifestu jsou také popsány jednotlivé části aplikace - Activity („okno aplikace"), Service („služby běžící na pozadí") nebo Receiver („služba čekající na nějaký signál") a jiné. U každé je popsáno, jaká třída je implementuje, jejich titulek a další nastavení (například typ orientace obrazovky u Activity).

Mimo to jsou v manifestu popsaná i práva, která aplikace od systému vyžaduje. Systém přidělování práv se snaží chránit uživatele zařízení před podvodnými aplikacemi, které mohou zařízení zneužít. Při instalaci aplikace se uživatel vždy dozví, jaká práva aplikace vyžaduje. Tedy pokud například aplikace, která zobrazuje počasí, požaduje práva k odesílání SMS zpráv, něco zřejmě není v pořádku a jedná se s velkou pravděpodobností o aplikaci podvodnou.

K podobným úkonům (přístup k internetu, zjištění polohy, práce s telefonními hovory, využití fotoaparátu apod.) je tedy vždy potřeba mít udělené právo, o které se žádá právě v souboru *AndroidManifest.xml.*

Příklad souboru *AndroidManifest.xml*:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.kinst.jakub.coursemanager"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_

    <supports-screens android:resizeable="true" />

    <application
        android:allowClearUserData="true"
        android:debuggable="true"
```

```
        android:hardwareAccelerated="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:uiOptions="splitActionBarWhenNarrow" >
        <activity
            android:name=".CourseList"
            android:label="@string/app_name" >
                    <!-- tato aktivita bude výchozí a bude zobraz
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNC
            </intent-filter>
        </activity>
    </application>
</manifest>
```

5.2.2 Resources (zdroje)

Při vývoji aplikací pro Android je pro přehlednost dobré, aby co nejvíce zdrojů, které aplikace používá, bylo umístěno jinde než v samotném kódu programu. Z toho důvodu Android poskytuje rozhraní pro práci se zdroji (resources). Zdrojem jsou myšleny například bitmapy, zvuky, fonty, styly prvků uživatelského rozhraní, pole hodnot, řetězce a překlady řetězců a podobně. Pokud jsou tyto zdroje uloženy v těch správných podsložkách (a ve správných XML souborech v případě řetězců, polí nebo dalších textových dat) složky */res*, zásuvný modul vývojového prostředí během automatického sestavení vygeneruje třídu s názvem *R*, která posléze slouží jako přístupový bod ke zdrojům ze zdrojového kódu. K identifikátoru konkrétního řetězce se pak lze dostat například pomocí výrazu *R.string.nazev_retezce*.

Další důležitou součástí aplikace, která je považována za zdroj jsou definice uživatelského rozhraní (layout). Rozvržení jednotlivých obrazovek je také uloženo ve formátu XML a Android SDK nabízí praktické vizualizované rozhraní pro vytváření těchto souborů. Jednotlivé prvky uživatelského rozhraní musejíé mít nastavené ID, pomocí kterého k nim lze přistupovat ze zdrojových kódů a dále s nimi pracovat.

Příkladem definice uživatelského rozhraní může být následující příklad:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/andro
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical" >

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" /

    <Button
        android:id="@+id/solveButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/solve" />

        <Button
        android:id="@+id/correctButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/correct" />

</LinearLayout>
```

Načítání zdrojů má ale kromě přehlednosti i další výhody. Tak například systém automaticky řeší překlad řetězců v aplikaci podle nastavneného jazyka zařízení. Do souboru *res/values/strings.xml* se běžně ukládají zdrojové řetězce aplikace. Pokud chce vývojář poskytnout překlad do češtiny, stačí aby vytvořil soubor *res/values-cs/strings.xml* ve kterém jsou uloženy překlady libovolné podmnožiny řetězců (zbytek je načten z výchozího jazyka). Když potom bude aplikaci používat uživatel s nastaveným českým jazykem, automaticky se budou zobrazovat překlady. Podobně lze rozlišit zdrojové soubory podle rozlišení displeje zařízení nebo verze OS Android.

5.2.3 Zdrojový kód

Samozřejmě nesmí chybět místo pro samotnou logiku aplikace, což je složka *src*, do které se ukládají jednotlivé třídy aplikační logiky. Asi nejdůležitější třídou při vývoji aplikace pro Android je třída *Activity*, která reprezentuje jednu „obrazovku" nebo „okno". Standardní aplikace je tvořena převážně právě z potomků této třídy. Implementace třídy poskytuje přístup ke všem funkcím spojeným se zařízením. Nabízí také možnost odchytávat událost různého druhu, jako je dotyk obrazovky (kliknutí), stisknutí hardwarových kláves a podobně. Kromě třídy *Activity* se často implementují také třídy potomkové tříd *Service* nebo *Receiver*. První v pořadí se používá, pokud je potřeba provádět libovolný proces, který nepotřebuje zobrazovat žádné uživatelské rozhraní (běží na pozadí). *Receiver* je v podstatě přijímač libovolných signálů ať už ze stejné aplikace, nebo z jiné či ze samotného operačního systému. Všechny použité „aktivity", služby a receivery musejí být definovány v souboru *AndroidManifest.xml*.

Kapitola 6

Architektura implementace webové aplikace

Celá aplikace je postavena na Nette Frameworku ve verzi 2.0dev pro PHP 5.2. Aplikace se snaží využívat většiny předností frameworku jako jsou pokročilé formuláře, šablonovací systém, routování a další. Důraz je kladen na dodržování zásad architektury MVC ve všech místech aplikace.

Uživatelské rozhraní je nakódováno pomocí jazyka HTML s použitím kaskádových stylů CSS. Některé prvky uživatelského rozhraní jsou naprogramovány v JavaScriptu za pomoci frameworků *jQuery[5]* a *jQuery UI[6]* a několika veřejně dostupných zásuvných modulů do těchto frameworků (formulářový prvek pro výběr data, kalendář apod.).

Databázová vrstva je implementována pomocí velmi rozšířené platformy MySQL, která však implementuje pouze úložiště dat. Nad databázovou vrstvou stojí ještě knihovna *Dibi[3]*, která usnadňuje práci s databází. Díky ní může modelová vrstva aplikace fungovat tak, že modelové třídy obsahují veřejné statické metody, které staticky dokáží přes *Dibi* přistupovat k připojené databázi. To usnadňuje možné komplikace s přenášením instancí připojení k databázi.

Aplikace je vyvíjena jako vícejazyčná, proto jsou jednak všechny řetězce v kódu „obaleny" překladovou funkcí a také využito překladových nástrojů, které jsou součástí Nette Frameworku (překlad formulářů apod.). V presenterech je dostupný objekt *Translator*, který využívají překladové funkce k překladu do aktuálně nastaveného jazyka. Samotný překlad probíhá pomocí technologie *GetText[4]* - z externích souborů s příponou *.mo* se získávají překlady řetězců v daném jazyce. Výchozím jazykem aplikace je angličtina a většina řetězců je strojově přeložena i do českého jazyka. Další jazyky lze snadno do implementace doplnit.

6.1 Návrh databáze

Návrh databázového modelu nejlépe popíše grafické znázornění tabulek a jejich závislostí. Viz obrázek v přílohách na straně 48.

6.2 Logické moduly

Implementace aplikace je od začátku rozdělena do logických celků (modulů), mezi kterými probíhá určitá forma interakce.

Zpravidla je každý logický modul reprezentován Nette Presenterem, který obvykle zpracovává uživatelský vstup, komunikuje s modelem a definuje formuláře, několika šablonami, které určují výsledné zobrazení v HTML a modelovou třídou, která obstarává zejména komunikaci s databázovou vrstvou.

TODO: diagram závislostí

6.2.1 CourseList

Modul CourseList má za úkol vypisovat seznam kurzů, jichž je přihlášený uživatel členem. Dalším úkolem bude zavádění nových kurzů do databáze. Modul také zobrazuje pozvánky do nových kurzů čekající na odpověď. Po přidání kurzu převezme práci modul Course, kde je teprve možné zvát uživatele a provozovat další aktivity. Model nabízí metody na výběr a správu kurzů z databáze - úpravy, mazání a další možnosti.

6.2.2 Course

Modul Course, jak již název napovídá, má za úkol zobrazovat titulní stránku kurzu, včetně seznamu lekcí. Na titulní stránce je zobrazen i seznam studentů, popř. jejich e-mailové adresy. Dále zde může lektor upravovat vlastnosti kurzu, zvát další studenty, registrovat další lektory, zaznamenávat další lekce atp. V detailu lekce jsou zobrazeny také komentáře studentů/lektorů a soubory, které lektor k lekci přiložil. Tento modul se stará i o přidávání studentů do kurzu. Přidání probíhá za pomoci pozvánky. Předpokladem je to, že lektor zná e-mailovou adresu studenta. Po vytvoření kurzu přes modul CourseList zadá tvůrce e-mailové adresy všech studentů a tím studenty do kurzu pozve. Pokud je již e-mailová adresa v systému vedena jako uživatelské jméno, objeví se uživateli v systému žádost o začlenění do kurzu. Pokud student systém ještě nepoužíval, bude na adresu zaslána pozvánka k registraci do systému. Po zaregistrování už jen potvrdí pozvánku do kurzu. Model nabízí metody na načtení/upravování vlastností kurzu, metody na získávání seznamu lekcí

z databáze, zanášení nových lekcí, úprava lekcí, vkládání komentářů u lekcí, nahrávání souborů k lekci a další.

6.2.3 Assignments

Modul Assignments vypisuje k danému kurzu seznam aktuálních úkolů/testů ke zpracování. Student může do data odevzdání odevzdat své řešení úkolu. Buď ve formě odpovědí na otázky s možnostmi A,B,C... nebo jako text, popřípadě je možnost odevzdat jako řešení libovolný soubor. Lektor má možnost úkoly vytvářet a upravovat, popřípadě mazat. Úkol je reprezentován formulářem, který obsahuje standardní formulářové prvky. Lektor dostává možnost otevřít úkol až v nějaký předem stanovený čas a nastavit jak dlouho bude „okno" pro odevzdávání otevřeno. Úkol může také fungovat jako test. Toho lze dosáhnout tím, že mu lektor nastaví maximální možnou dobu řešení. Jakmile student spustí řešení, začne běžet čas. Po uplynutí je formulář automaticky odeslán v aktuálním stavu a student již nemůže své řešení měnit. Tento modul také umožňuje lektorovi opravovat vložená řešení úkolů/testů a následně je i obodovat. Model nabízí výpis úkolů z databáze, přidávání a úpravy jejich struktury. Také do systému zanáší řešení jednotlivých studentů a ohodnocení získané od lektora. Pokud se jedná o úkol s autokorekturou, nabídne model i automatickou kontrolu řešení vzhledem k zadání a přidělení počtu bodů.

6.2.4 Results

Modul Results má za úkol zejména vypisovat tabulku výsledků všech studentů kurzu. Jedná se o výsledky jak klasických testů nebo úkolů, tak i testů typu online, které buď opraví lektor ručně nebo je opraví systém sám (autokorektura). Model umí vypisovat výsledky z databáze (a to jednak výsledků online úkolů, ale také klasických offline úkolů, jejichž výsledky zadá lektor ručně), počítat aritmetický/vážený průměr a součet bodů studenta. Dále dokáže vkládat do databáze výsledky offline úkolů/testů.

6.2.5 Events

V případě modulu Events se jedná v podstatě o malou verzi kalendáře. Každý kurz k sobě může mít napojeno neomezené množství událostí všeho druhu. Ať už se jedná o exkurze, testy nebo jiné speciální akce. Student o těch nejbližších bude informován na titulní stránce daného kurzu. Všechny události pak bude možné shlédnout na speciální stránce zanesené v přehledném kalendáři. Model poskytne výpis budoucích událostí, vkládání a editaci existujících událostí.

6.2.6 Resources

Tento modul nabídne lektorovi sdílení jakéhokoliv typu souboru. Jednak může připojit soubor ke konkrétní lekci, ale také může přidat obecné soubory, které jsou vázané jen na kurz. Model se stará o ukládání informací o souborech v databázi a ukládání a správu samotných souborů v souborovém systému. Samotné nahrávání souborů na server obstarává samostatná nezávislá Nette komponenta *Uploader*.

6.2.7 User

Modul User dokáže zobrazit profil daného uživatele - jeho jméno, kontaktní údaje atp. Modul se také stará o zobrazení průvodce vytvořením nového uživatele (i na základě pozvánky). Uživatel pomocí modulu může své údaje i upravovat. Uživatel má v každém kurzu jednu ze dvou rolí. Buď se jedná o lektora kurzu, který může kurz spravovat, nebo jde o studenta. Díky realizaci je možné, aby jeden uživatel mohl být v jednom kurzu v roli studenta a v jiném v roli lektora. Model nabízí metody k přidání uživatele, aktualizaci údajů, generování kontrolních registračních odkazů, kontrola bezpečnosti hesla. Také se stará o samotné přihlašování uživatelů - kontrolu údajů.

6.2.8 Forum

Modul Forum zajišťuje rychlou komunikaci mezi posluchači a lektorem pomocí populárního stylu diskusního fóra. Fórum nabídne možnost vytvořit vlákno (téma), na které mohou ostatní vzápětí odpovídat. Model poskytuje vkládání témat do databáze, následně také vkládání odpovědí a správné propojení s tématem, na kteroé odpovídá.

6.2.9 Messages

Modul Messages slouží k soukromé komunikaci mezi dvěma uživateli. Uživatel může odeslat zprávu druhému uživateli. Na zprávy lze jednoduše odpovídat. Model musí zvládnout zanést zprávu do databáze a zobrazit ji u příjemce ve složce doručených zpráv a odesílateli v odeslaných zprávách.

6.2.10 Settings

V modulu Settings je možné nastavovat uživatelské preference systému. Nastavit možnosti zasílání e-mailových upozornění v rámci jednotlivých modulů,

nastavení jazyka aplikace a další. Model poskytne potřebné metody pro aktualizaci nastavení. Dále nabízí výchozí profil nastavení, který bude přidělen každému novému účtu.

6.2.11 Mail

Modul Mail je používán kdykoliv, kdy je potřeba odeslat e-mail uživateli. Model se postará o vytvoření struktury mailu s potřebnou zprávou. Některá upozornění je potřeba odeslat bezprostředně po uživatelově interakci (např. vytvoření úkolu lektorem), jiná však potřebují být odeslána automaticky v určitou dobu bez spuštění skriptu ze strany uživatele. Tyto e-maily jsou proto odesílány pomocí démona cron na straně hostingu, který v určitý interval automaticky spouští skript (požadavek na určenou URL na serveru), mající za úkol podívat se do databáze, zkontrolovat, zda je potřeba někomu odeslat upozornění a popřípadě je všechna odeslat.

Kapitola 7

Implementace propojení s mobilní aplikací

Součástí implementace je Nette doplněk, který dokáže u již existující standardní Nette aplikace otevřít její API a Java knihovna, která se umí jako klient k takovému API jednoduše připojit. Knihovny na propojení byly vyvíjeny izolovaně, aby bylo možné je použít i jinde.

7.1 Na straně serveru

7.1.1 Popis rozšíření

Rozšíření pro Nette otevírá API aplikace tak, aby ho bylo možné použít pro již existující aplikaci, která s veřejným API od začátku nepočítala. Je třeba vyřešit, jak aplikace pozná, že na straně klienta je mobilní aplikace a také to, jak aplikaci poslat potřebná data. Na druhé straně klient musí data nějakým způsobem přečíst a zpracovat, aby je mohla dále prezentovat uživateli.

Nette aplikace pozná, že klientem je mobilní aplikace tak, že jedním z parametrů GET požadavku je parametr *mobile* s hodnotou *1*. Presenter se pak při vytváření odpovědi řídí hodnotou proměnné *mobile*. Ta musí být nastavena jako perzistentní (přenáší se mezi odkazy automaticky), aby se zachovala její hodnota i při případném přesměrování na straně serveru.

Dále je nutné vyřešit, jaká data se budou klientovi posílat. Implementace je navržena tak, aby mobilní aplikace nahradila pouze vrstvu View (z MVC), tedy aby měla za úkol pouze zobrazovat již připravená data. Cílem je tedy zachovat funkce modelů i presenterů a pouze nahradit funkci *latte* šablon. Toho lze nejsnáze docílit tak, že se v životním cyklu presenteru najde místo, které se nachází těsně před vykreslením stránky a místo samotného vykreslení pomocí

šablon se všechny proměnné nastavené šabloně (*$template->getParameters()*) vypíší ve zvoleném přenosovém formátu na výstup. V praxi je ale nutné vypsat proměnné všech šablon - šablony presenteru a k tomu ještě šablon všech vytvořených komponent. Ono hledané místo „těsně před vykreslením" není místem v presenteru, ale až v instanci třídy imlepemtující *IPresenterResponse* po jejím vytvoření, protože až tam proběhnou metody *render()* všech použitých komponent, ve kterých se šablony plní proměnnými. Z toho důvodu bylo v implementaci potřeba vytvořit vlastní objekt implementující *IPresenterResponse*, který všechny komponenty a presenter projde a proměnné posbírá.

Celým tímto procesem se otevře API aplikace, které posléze funguje podle pravidel architektury REST s výjimkou případného přenášení Cookies (záleží na implementaci webové aplikace).

7.1.2 Instalace rozšíření

Instalace na straně serveru je cíleně velmi jednoduchá. Je potřeba pouze aby všechny presentery aplikace byly potomky třídy *AndroidettePresenter*, tedy standardně stačí aby se potomkem stala použitá třída *BasePresenter*. Rozšíření umožňuje před odesláním strukturu dat profiltrovat, tedy odstranit konkrétní data, u kterých není potřeba, nebo je dokonce nežádoucí aby se přenášela (například hesla uživatelů, která sice do šablony v objektu uživatele odesíláme, ale při přenosu do mobilní aplikace je odesílání nežádoucí a nepotřebné). Toho lze docílit předefinováním metody *processAndroidVariables()*, která získá data v parametru a vrací ta profiltrovaná.

Druhým požadavkem je použití speciální Routy v routování aplikace. V souboru *bootstrap.php* se musí k definování routování přidat řádek:

```
$router[] = new AndroidetteRoute('courselist:homepage');
```

Routa musí být na takové pozici, aby byla vždy dosažitelná a zároveň nekolidovala s těmi již existujícími.

Nyní je integrace dokončena a aplikace již dokáže posílat klientovi ve strojově čitelné formě. Samozřejmě jen pokud je GET parametr *mobile* nastaven na hodnotu *1*. Jinak se aplikace chová stejně jako doposud.

V případě potřeby zabezpečení API je nutné nějakým způsobem autentizovat klienta. Například pomocí API klíče - presenteru se nastaví tajný řetězec (API klíč) a ten je pak požadován jako součást každého požadavku (klient musí řetězec znát a v každém dotazu ho posílat např. jako GET parametr).

7.2 Formát přenosu dat

Dalším krokem je výběr formátu přenosu dat. Pokud je webová aplikace naprogramovaná v jazyce PHP a mobilní aplikace v jazyce *Java*, výchozí knihovny umožňují jednoduší práci s formátem JSON.

Na straně PHP je implementace jednoduchá, pomocí standardních funkcí *json_encode()* a *json_decode()* lze snadno převést jakékoliv pole nebo objekt do formátu JSON a naopak. Příprava dat k odeslání je tedy vždy otázkou jednoho řádku kódu.

Na straně mobilního zařízení, při programování v jazyce *Java*, je situace složitější. Problém je způsoben odlišným charakterem programovacích jazyků PHP a Java. PHP jako skriptovací jazyk podporuje libovolnou strukturu pole - například asociativní pole. V poli tedy může být klíčem i řetězec a hodnotou může být v podstatě cokoliv. Při zpětné kompozici pole nebo objektu v Javě nastává problém, kterým je nutnost typování proměnných. Obsah pole se musí interpretovat ručně. Je nutné vědět jakého typu jsou hodnoty polí.

Zpracování JSON dat v Javě řeší standardní knihovna *org.json*. Ta nabízí sadu tříd pro kompletní práci s JSON daty. Nejdůležitější jsou třídy *JSONObject* a *JSONArray*. Používají se podle charakteru čtených dat. Pokud se jedná o klasické pole, použijeme objekt *JSONArray*. Pokud jde o asociativní pole nebo objekt, využijeme objekt *JSONObject*. Přes *JSONArray* lze iterovat a vybírat data, z konkrétní pozice v poli. Z objektu *JSONArray* lze vybírat data podle klíče ve formě řetězce. Data získáváme dostupnými metodami - můžeme získat základní primitivní typy (*int, String, boolean, double,* atd.) a nebo rekurzivně *JSONArray* nebo *JSONObject*.

Příklad použití:

```
...

// precteme vstupni data (String) jako JSONArray
JSONArray auta = new JSONArray(vstupni_data_json);

// precteme prvni zaznam jako JSONObject
JSONObject prvniAuto = auta.getJSONObject(1);

// vybereme vlastnost "motor" ve vlastnosti "parametry"
// jako JSONObject
JSONObject motor = prvniAuto.getJSONObject("parametry")
                                                .getJSONObject("motor");

// zjistime spotrebu auta (typ double)
```

```
double spotreba = motor.getDouble("spotreba");

...
```

7.3 Na straně klienta

Java knihovna *cz.kinst.jakub.androidette* má za cíl implementovat klientské připojení Android aplikace k Nette aplikaci s nainstalovaným výše zmíněným rozšířením. Snaží se vývojáři co nejvíce usnadnit práci a zároveň zachovat obecnost řešení, která je klíčová pro možnost aplikování na jakoukoliv Nette aplikaci.

Implementace se sestává z několika vrstev. Nejnižší vrstva je implementovaná třídou *HttpSmartClient*. Ta především pomocí standardní knihovny *org.apache.http* řeší vytváření HTTP požadavků (*HttpRequest*) na server a získávání (zatím nijak neinterpretovaných) textových dat z jeho odpovědí. Metodám vyšší vrstvy předávají URL cíle, GET parametry, POST parametry a případně také přiložené soubory (pro případ nahrávání souborů na server). Klient dokáže z přijatých odpovědí získávat HTTP Cookies z hlaviček *Set-Cookie*, uložit je do paměti a v dalších požadavcích Cookies odesílat na server v hlavičkách *Cookie*. Pokud je potřeba, umí klient získat ze serveru i libovolný soubor - tedy binární data. Třída *HttpSmartClient* implementuje i možnost připojení k zabezpečenému serveru pomocí technologie HTTPS.

Další vrstvu implementuje třída *AndroidetteJSONClient*. Ta má za úkol interpretaci dat získaných ze serveru. Přijatá textová data konvertuje do objektu *JSONObject*, který postupuje do vyšších vrstev. Kromě toho třída identifikuje speciální části získaných dat. Uloží tzv. Flash zprávy (informační zprávy pro uživatele používané v Nette aplikacích) a také speciální zprávy *___error* a *___dump*. Ty slouží k zaznamenávání chyb a ladících informací. Vývojář může z Nette aplikace naplnit tyto speciální zprávy, které poté *AndroidetteJSONClient* automaticky vypíše do standardní logovací konzole (Android LogCat).

Nejvyšší vrstva (třída *AndroidetteConnector*) už implementuje abstrahované připojení ke konkrétní Nette aplikaci. Je potřeba jí nastavit kořenovou URL aplikace na serveru a poté již lze využívat jejích veřejných metod. Tyto metody nabízejí získání dat z konkrétních akcí konkrétních presenterů, odeslání formuláře na jisté akci presenteru nebo odeslání signálu na presenteru. Metodám lze poskytnout dodatečné parametry (GET nebo POST), popřípadě soubory k odeslání na server. Metody vracejí odpověď serveru ve formátu *JSONObject*, ze kterého lze získávat konkrétní data proměnných šablon.

Práce s webovou aplikací potom může vypadat tak jednoduše jako následující příklad:

```
...

AndroidetteConnector mojeAplikace = new AndroidetteConnector("http://

ArrayList<NameValuePair> getArgs = new ArrayList<NameValuePair>();
getArgs.add(new BasicNameValuePair("id","123"));

// ziskame data akce "detail" presenteru "product" s parametrem id=12
JSONObject product = mojeAplikace.getAction("product","detail",getArg

double cena = product.getDouble("price");

...
```

Pokud je potřeba chování upravit, stačí třídu *AndroidettePresenter* podědit a upravit k potřebám konkrétní aplikace - například vyřešit přihlášení před prvním dotazem na server.

Kapitola 8

Architektura implementace mobilní aplikace

Mobilní aplikace je implementována standardně pomocí několika potomků třídy *Activity* (dále aktivity). Zpravidla každá aktivita (někdy více aktivit) implementuje jeden z modulů jak jsou popsané v Architektuře implementace webové aplikace. Aktivity tvoří určitou strukturu, podle které se mezi nimi řídí navigace.

Každá z aktivit je potomkem třídy *CMActivity* která slouží jako jakási šablona. Díky tomu mají všechny aktivity definovanou jednu hlavní úlohu, která proběhne vždy po zobrazení a také pokud uživatel v menu stiskne tlačítko Refresh (Obnovit). Typicky jde o stažení dat z webového protějšku a jejich prezentace (zobrazení na displeji). Třída *CMActivity* je potomkem třídy *TabbedActivity*, která jí umožňuje rozdělit na několik záložek, mezi kterými lze přepínat. Lze tedy zobrazit více informací (podstránek) v rámci jedné aktivity.

Dále si díky *CMActivity* aktivity udržují objekt, pomocí kterého získávají data z webové aplikace. Tento objekt je potomkem třídy *AndroidetteConnector*, která je popsána v Architektuře implementace propojení s mobilní aplikací. Potomkem je proto, že musí obstarávat ještě další procesy, které předchůdce neimplementuje. Jde například o zajištění přihlášení uživatele - před prvním požadavkem na server je potřeba ve webové aplikaci odeslat přihlašovací formulář se správnými udaji. Tím je uživatel přihlášen a díky použití Cookies ho není potřeba přihlašovat při dalších požadavcích. Poté již stačí v každé aktivitě stáhnout data z odpovídající akce na odpovídajícím presenteru a ty vhodným způsobem prezentovat - například naplnit daty *ListView* (seznam položek). Pokud je naopak potřeba nějaká data odeslat, v definici uživatelského rozhraní se vytvoří formulář a jeho data se potom pošlou na server pomocí POST požadavku - jako odeslání HTML formuláře.

Aplikace kromě připojení k internetu od uživatele nevyžaduje žádná zvláštní oprávnění.

TODO: Neni to moc malo ?

TODO: Implementace assignmentu

Kapitola 9

Možné rozšíření ??? Nebo do zaveru

Kapitola 10

Uživatelská dokumentace

10.1 Webová aplikace

10.1.1 Minimální požadavky

10.1.1.1 Provoz aplikace

Aplikace není vázaná na žádnou konkrétní platformu, jelikož potřebný software je dostupný pro většinu využívaných operačních systémů včetně *OS Linux*, *Microsoft Windows* a *Mac OS*.

Pro provoz aplikace je potřeba mít nainstalovaný následující software:

- webový server s podporou *OpenSSL (například Apache*[2]*)*
- *PHP*[9](verze alespoň 5.2)
- *MySQL*[7](verze alespoň 5.5)
- libovolný plánovač procesů (například program *cron*)

10.1.1.2 Použití aplikace

K použití uživateli stačí běžný webový prohlížeč. Aplikace je optimalizována pro prohlížeč *Google Chrome, Mozilla Firefox a Microsoft Internet Explorer 7* a vyšší.

10.1.2 Instalace aplikace

K nainstalování aplikace stačí následovat následující pokyny:

1. Zkopírovat obsah adresáře *CourseManager* do výchozí složky webového serveru (typicky složka *htdocs*)

2. Pokud jde o UNIXový operační systém, je potřeba nastavit oprávnění pro následující složky na hodnotu *777*

   - */temp*
   - */document_root/webtemp*
   - */log*
   - */uploads*

3. Vytvořit na serveru databázi s názvem *course-manager* určenou pro aplikaci

4. Nastavit přístupové údaje k databázi v souboru */app/config.neon*

5. Nastavit přístupové údaje k libovolnému SMTP serveru k odesílání oznámení e-mailem v souboru */app/config.neon*

6. Vytvořit základní databázové schéma v MySQL databázi spuštěním skriptu */db/CourseManager.sql*

7. Nakonfigurovat plánovač úloh aby spouštěl HTTP GET požadavky na následující URL

   - *URL/document_root/cron/sendemailsew5q3n825ml6bm2btz81* (každou minutu)
   - *URL/document_root/cron/deleteuncheckedusersew5q3n825ml6bm2btz81* (každý den)
   - *URL/document_root/cron/sendassignmentnotificationew5q3n825ml6bm2btz81* (každý den)
   - *URL/document_root/cron/deleteoldtempfilesew5q3n825ml6bm2btz81* (každý den)

   Řetězec *URL* je potřeba nahradit adresou URL provozované aplikace

10.1.3 Spuštění aplikace

Webovou aplikaci CourseManager lze spustit ve webovém prohlížeči přechodem na adresu aplikace. Po instalaci na lokální webový server se jedná typicky o adresu `http://localhost/slozka_aplikace/document_root`. Webová aplikace je ale také veřejně dostupná na adrese `http://rp-jakub-cz.dc.starver.net`.

10.1.4 Výchozí stránka, registrace uživatele a přihlášení do aplikace

Výchozí stránka nabízí základní informace o aplikaci a vyzývá uživatele k registraci, popřípadě k přihlášení. Anonymní uživatel má právo zobrazit pouze tuto stránku a stránku s uživatelským manuálem.

Kliknutím na tlačítko Přihlášení (Login) se rozbalí nabídka pro přihlášení společně s odkazem na stránku registrace uživatele.

Aplikace při registraci požaduje po uživateli jméno, příjmení, uživatelské jméno ve formě e-mailové adresy a heslo o minimální délce 5 znaků, které musí uživatel z bezpečnostních důvodů zadat dvakrát.

Po odeslání registračního formuláře je do několika minut uživateli odeslán potvrzující e-mail s odkazem, po jehož navštívení potvrdí vlastnictví zadané e-mailové adresy.

Od chvíle potvrzení adresy je uživateli umožněno přihlášení do aplikace prostřednictvím přihlašovacího formuláře. Po přihlášení je uživatel přesměrován na stránku zobrazující kurzy ve kterých je v roli učitele a kurzy, ve kterých je v roli studenta.

10.1.5 Zakládání nového kurzu a pozvání studentů

Nový kurz může přihlášený uživatel vytvořit kliknutím na tlačítko Přidat kurz (Add Course). Kurzu je potřeba přiřadit název a krátký popisek.

Uživatel, který kurz založil, automaticky získává v kurzu roli učitele. Po vytvoření je přesměrován na domácí stránku kurzu, která obsahuje seznam učitelů, seznam studentů, nadcházející události a především seznam lekcí kurzu.

Po založení kurzu na řadu přichází rozeslání pozvánek studentům. Kliknutím na tlačítko Pozvat studenty (Invite students) se učiteli zobrazí stránka, kde může postupně zadávat e-mailové adresy, které získal od studentů. Každý takto pozvaný student obdrží e-mail s pozvánkou do kurzu. Pokud je již v systému zaregistrován, stačí na domovské obrazovce akceptovat pozvánku. V opačném případě musí nejdříve projít procesem registrace(musí jako uživatelské jméno použít e-mailovou adresu, na kterou přišla pozvánka) a až poté je mu rovněž na domovské stránce zobrazena pozvánka.

10.1.6 Seznam lekcí a detail lekce

Domovská stránka kurzu mimo jiné obsahuje především seznam všech lekcí seřazený tak, aby nejnovější lekce byly vypsány jako první. Při otevření stránky jsou všechny lekce kromě nejnovější zobrazeny v úsporné podobě, lekci lze

“rozbalit” kliknutím na její titulek. Když je rozbalena, jsou zobrazeny základní informace jako titulek lekce, obsah lekce a odkaz na zobrazení detailu lekce.

Detail lekce zobrazuje kompletní data o lekci včetně přiložených souborů, komentářů studentů i učitelů a formuláře pro přidání nového komentáře.

Uživatel v roli učitele může navíc upravit atributy kurzu, přidávat a měnit obsah lekcí. Při přidávání lekce může učitel formátovat obsah pomocí systému Texy[11], který umožňuje přidávat nadpisy několika úrovní, obrázky, hypertextové odkazy, číslované i nečíslované seznamy, formátované zdrojové kódy nebo emotikony a další. Více informací o syntaxi viz. odkaz výše. Kdykoliv během vytváření obsahu lze kliknout na tlačítko Náhled (Preview) a nechat si zobrazit obsah zformátovaný tak, jak bude ve výsledku vypadat.

K jednotlivým lekcím může učitel také nahrávat soubory, které chce studentům poskytnout ke stažení, především studijní materiály. Pomocí tlačítek + a - nejdříve zvolí počet souborů a poté je vybere ze svého počítače. Po odeslání formuláře jsou soubory nahrány.

Poslední výsadou role učitele je možnost učinit kohokoliv ze studentů učitelem. Pokud je tedy v praxi potřeba více učitelů na jeden kurz, musí jeden z nich kurz vytvořit a ostatní nejdříve pozvat jako studenty a následně je povýšit do role učitele.

10.1.7 Výsledky (Results)

Stránka s výsledky zobrazuje studentům jejich průběžné výsledky. Sjednocuje výsledky klasických (offline) testů/úkolů, které učitel zadá ručně, dále online úkolů, které učitel vyvěsí v systému a nechá uživatele odevzdat svá řešení, která následně opraví a nakonec také online úkoly, které se dokáží podle zadaných správných odpovědí opravit samy.

Výsledky jsou rozděleny podle toho, zda je jejich ohodnocením počet bodů nebo známka na stupnici 1-5. Podle toho je napravo tabulky také vypsán buď součet bodů nebo aritmetický průměr známek.

Učitel zde může přidávat klasický(offline) úkol, kterému přiřadí jméno, typ(body/známky) a dále rovnou ohodnotí všechny studenty v kurzu.

10.1.8 Úkoly/Testy (Assignments)

Jednou ze stěžejních částí aplikace je možnost zadávat studentům online domácí úkoly nebo testy. Učitel může jednoduše vytvořit úkol, nastavit mu datum, kdy se úkol stane přístupným pro řešení, datum, do kdy bude přístupný pro řešení a volitelně může také nastavit, jak dlouho může student strávit vyplňováním úkolu (tím se z úkolu v podstatě stává test).

Po vyplnění údajů o úkolu nastává fáze tvoření formuláře pro zadávání odpovědí. Učitel má k dispozici několik možností, jak od studentů získat odpověď.

- Krátký text (Text input) - umožňuje zadat krátký text jako odpověď na otázku
- Dlouhý text (Text area) - umožňuje zadat komplexnější odpověď na otázku
- Jednoduchý výběr z možností (Radio list) - umožňuje vybrat jednu z předem nabídnutých odpovědí
- Výběr podmnožiny z možností (Multiselect list) - lze vybrat libovolnou podmnožinu z předem nabídnutých odpovědí
- Nahrání souboru (File upload) - lze nahrát jakýkoliv soubor s řešením

Úkol může představovat libovolnou kombinaci těchto prvků. Každému prvku je potřeba nastavit popisek - nejčastěji otázky, kterou má student zodpovědět. Učitel může strukturu úkolu kdykoliv změnit.

Pokud je již úkol „otevřený", můžou ho studenti začít kdykoliv řešit až do uzávěrky. V případě, že učitel nastavil maximální dobu řešení, je student s touto skutečností obeznámen a jakmile stiskne tlačítko Řešit (Solve), začíná tato lhůta běžet. Aplikace si zapamatuje, kdy s řešením začal (pro případ zavření okna prohlížeče, odhlášení atd.)a po daném časovém úseku již není možné úkol řešit. Pokud se stane, že uživatel řeší úkol v těsné blízkosti uzávěrky, zobrazí se mu na stránce odpočet času. Po uplynutí lhůty bude formulář automaticky odeslán v aktuálním stavu.

Učitel může kdykoliv začít opravovat řešení studentů, kteří ho již odevzdali. Při opravování se vytvoří tabulka, kde sloupce reprezentují otázky v úkolu a řádky odpovědi jednotlivých studentů. Na konci každého řádku je místo na vložení bodového ohodnocení. Výsledky se automaticky zapíší do tabulky Výsledky.

Pokud to úkol umožňuje, může ho učitel vytvořit jako samoopravný (Autocorrect). To znamená, že při vytváření struktury nastaví pro každou otázku i správnou odpověď, nastaví pro úkol maximální možný počet bodů a aplikace podle odpovědí studenta sama spočítá kolika procent, potažmo bodů získává.

Kvůli automatickému vyhodnocování úkolů není možné nabídnout při vytváření úkolu všechny komponenty jako v klasickém online úkolu. Do samoopravného úkolu nelze přidat komponentu pro dlouhý text a nahrávání souborů.

10.1.9 Zdroje/Dokumenty

Tato sekce sdružuje materiály poskytnuté studentům ke stažení. Nejčastěji se jedná o různé studijní materiály. Tyto dokumenty však nejsou vázány na konkrétní lekci, nýbrž pouze na kurz. Učitel soubory nahrává stejným způsobem jako při nahrávání souborů k lekci.

10.1.10 Fórum

Každý kurz má k dispozici jednoduché diskuzní fórum, které má za cíl usnadnit komunikaci mezi studenty, popřípadě lektory. Na rozdíl od soukromé komunikace mezi studenty je komunikace v diskuzním fóru dostupná všem ostatním studentům. Tudíž může například řešení nějakého problému posloužit i ostatním. Fórum se skládá z Témat (Topics), která jsou vypsána na výchozí stránce a uspořádána dle data poslední reakce na téma. Uživatel vytváří téma, které symbolizuje vlákno, ve kterém probíhá diskuze. Po kliknutí na téma se zobrazí odpovědi (replies) na téma seřazené dle data přidání. Zde mohou uživatelé kurzu prostřednictvím jednoduchého formuláře na přidávání odpovědí na dané téma diskutovat.

10.1.11 Události

Učitelé mohou ke kurzu přidat libovolnou událost, která je definovaná názvem, krátkým popisem a datem. Události naplánované na nadcházejících 10 dní jsou zobrazeny na hlavní stránce kurzu, ostatní události lze nalézt na stránce události. Zde jsou zobrazeny v kalendáři, který má několik možností zobrazení. Lze ho zobrazit jako denní kalendář, týdenní nebo měsíční. Kliknutí na událost v kalendáři vyvolá zobrazení detailu události.

Učitel může události jednoduše pomocí formuláře přidávat, popřípadě existující v jejich detailu měnit.

10.1.12 Zprávy

Aby bylo možné komunikovat s ostatními účastníky kurzu také soukromě, poskytuje aplikace systém zasílání soukromých zpráv. Hlavní obrazovky zpráv může uživatel zobrazit kliknutím na ikonu zprávy v horním fixním menu. Zobrazí se seznam příchozích zpráv.

Kliknutím na tlačítko Nová zpráva (New message) lze vytvořit a následně odeslat novou zprávu.

Každá zpráva je identifikována odesílatelem, příjemcem (oba ve formě uživatelského jména - e-mailové adresy), předmětem a obsahem zprávy - podobně

jako u e-mailové komunikace.

Na zprávy lze jednoduše odpovídat tak, že je původní zpráva v odpovědi citována.

10.1.13 Uživatelský profil

Uživatelský profil je osobní profilová stránka, která shrnuje data o uživateli - jeho jméno, příjmení, e-mailovou adresu nebo adresu webové stránky. Uživatel má právo svůj profil kdykoliv upravit.

10.1.14 Oznámení e-mailem

Aplikace umožňuje automatické rozesílání e-mailových oznámení v předem daný čas. Například po přidání online úkolu v modulu Úkoly (Assignments) jsou automaticky obesláni všichni studenti kurzu a jsou obeznámeni s novým úkolem. Dále si každý uživatel může v Nastavení zvolit jak dlouho před uzávěrkou úkolu má být na uzávěrku upozorněn.

10.1.15 Nastavení

Tato stránka umožňuje uživateli měnit své preference. Lze nastavit preferovaný jazyk aplikace, který bude aplikován po každém uživatelském přihlášení, dále to, zda bude pro ostatní uživatele viditelná e-mailová adresa a také lze nastavit kolik dní předem bude uživatel upozorněn na blížící se uzávěrku online úkolu.

10.2 Mobilní aplikace

10.2.1 Minimální požadavky

Aplikace je určena a vyvíjena pro mobilní zařízení se systémem Android[1]ve verzi 2.1 (Eclair), avšak pravděpodobně bude z velké části fungovat i na starších zařízeních se starší verzí systému. Testování proběhlo na mobilních telefonech se systémem verze 2.1, 2.2, 2.3 a 4.0. Aplikace vyžaduje připojení zařízení k internetu.

10.2.2 Instalace a spuštění aplikace

Aplikaci lze nainstalovat na zařízení standardním způsobem pomocí instalačního balíčku APK[1]. Soubor s příponou .apk je potřeba nakopírovat do zařízení a tam spustit. Proběhne standardní instalace aplikace a zástupce s titulkem *CourseManager* se objeví v seznamu nainstalovaných aplikací. Kliknutím na zástupce aplikace se aplikace spustí a zobrazí se výchozí obrazovka.

10.2.3 Výchozí obrazovka a přihlášení do aplikace

Po prvním spuštění je pro správné fungování potřeba aplikaci nakonfigurovat. Stisknutím standardního tlačítka MENU se zobrazí nabídka, ve které lze vyvolat obrazovku Nastavení. V nastavení je nutné nejdříve nastavit adresu běžící instance webové aplikace CourseManager - tedy například `http://rp-jakub-cz.dc.starver.net`. Dále musí uživatel vyplnit přihlašovací údaje - e-mailovou adresu a heslo.

Po návratu na výchozí obrazovku by už aplikace měla být úspěšně připojena na webový protějšek a zobrazovat seznam uživatelských kurzů - nejdříve kurzy, které učí a poté kurzy, které studuje. Po kliknutí na jeden z kurzů se zobrazí jeho detailní stránka s dalšími možnostmi navigace. V nabídce je v tuto chvíli kromě nastavení ještě přístup k soukromým zprávám uživatele.

10.2.4 Detail kurzu

Stránka kurzu nabízí stejně jako webová verze seznam jednotlivých lekcí, které jsou seřazeny podle jejich data. Po rozbalení lekce a kliknutí na tlačítko *Více* bude zobrazen detail této konkrétní lekce.

Důležitá je ale v tuto chvíli nabídka, která obsahuje odkazy na všechny moduly týkající se tohoto kurzu. Uživatel si může nechat zobrazit Diskuzní fórum kurzu, Události kurzu, Výsledky studentů, Úkoly (Assignments) a Zdroje (Resources).

10.2.5 Detail lekce

Detailní stránka konkrétní lekce je rozdělena pomocí záložek na dvě části. V první záložce se nacházejí komentáře k lekci. Uživatelé zde mohou lekci libovolně komentovat. Nový komentář se přidá pomocí položky v nabídce menu. Ve druhé záložce jsou vypsány soubory, které učitel k lekci přidal. Po kliknutí na položku je soubor stažen do zařízení a pokud je to možné, je otevřen v příslušné aplikaci. Stažené soubory se ukládají do složky `CourseManager_downloads`.

[1] APK, aplication package file

10.2.6 Diskuzní fórum

Mobilní aplikace nabízí kompletní možnost diskuze ve fóru - tedy zakládání nových témat diskuze, výpis odpovědí a přidávání nových odpovědí.

10.2.7 Události

Obrazovka událostí nabízí prostý výpis nejbližších událostí kurzu. Při dlouhém kliknutí na položku události se zobrazí kontextová nabídka s možností vložení události do kalendáře v zařízení. Otevře se dialogové okno s předvyplněnými detaily události a po potvrzení se událost uloží do výchozího kalendáře ve výchozí aplikaci Kalendář systému Android.

10.2.8 Výsledky

Na obrazovce Výsledky je k dispozici přehledná tabulka, kde sloupce představují jednotlivé úkoly/testy a řádky odpovídají konkrétním studentům kurzu. Tabulka je rozdělena na dvě části - úkoly/testy ohodnocené body a úkoly/testy ohodnocené známkou. Na konci tabulky je příslušnému studentovi vypočítán součet bodů, respektive aritmetický průměr známek.

10.2.9 Úkoly/Testy (Assignments)

Výchozí obrazovka Úkoly vypisuje všechny úkoly/testy kurzu pod sebou. Při kliknutí na položku se zobrazí detail, kde jsou možné dvě akce v závislosti na roli uživatele.

Pokud je uživatel v roli studenta, může úkol/test vyřešit přímo na mobilním zařízení. Aplikace podporuje klasické i samoopravné testy, s časovým omezením i bez něho. Řešení probíhá stejně jako ve webové verzi - pomocí standardních formulářových prvků zadává řešitel své odpovědi a následně řešení odešle příslušným tlačítkem. V horní části obrazovky je zobrazen zbývající čas pro řešení testu.

Pokud je uživatel v roli učitele, nemůže úkol/test řešit, ale může opravovat již odevzdaná řešení. Pomocí tabulky jsou mu zobrazeny jednotlivé odpovědi studentů a vzápětí pole pro vložení počtu získaných bodů.

10.2.10 Zdroje (Resources)

Tento modul nabízí výpis souborů, které nejsou přiřazeny ke konkrétní lekci, ale k celému kurzu. Stejně jako u souborů lekce stačí kliknout na položku a soubor se stáhne/otevře.

10.2.11 Zprávy

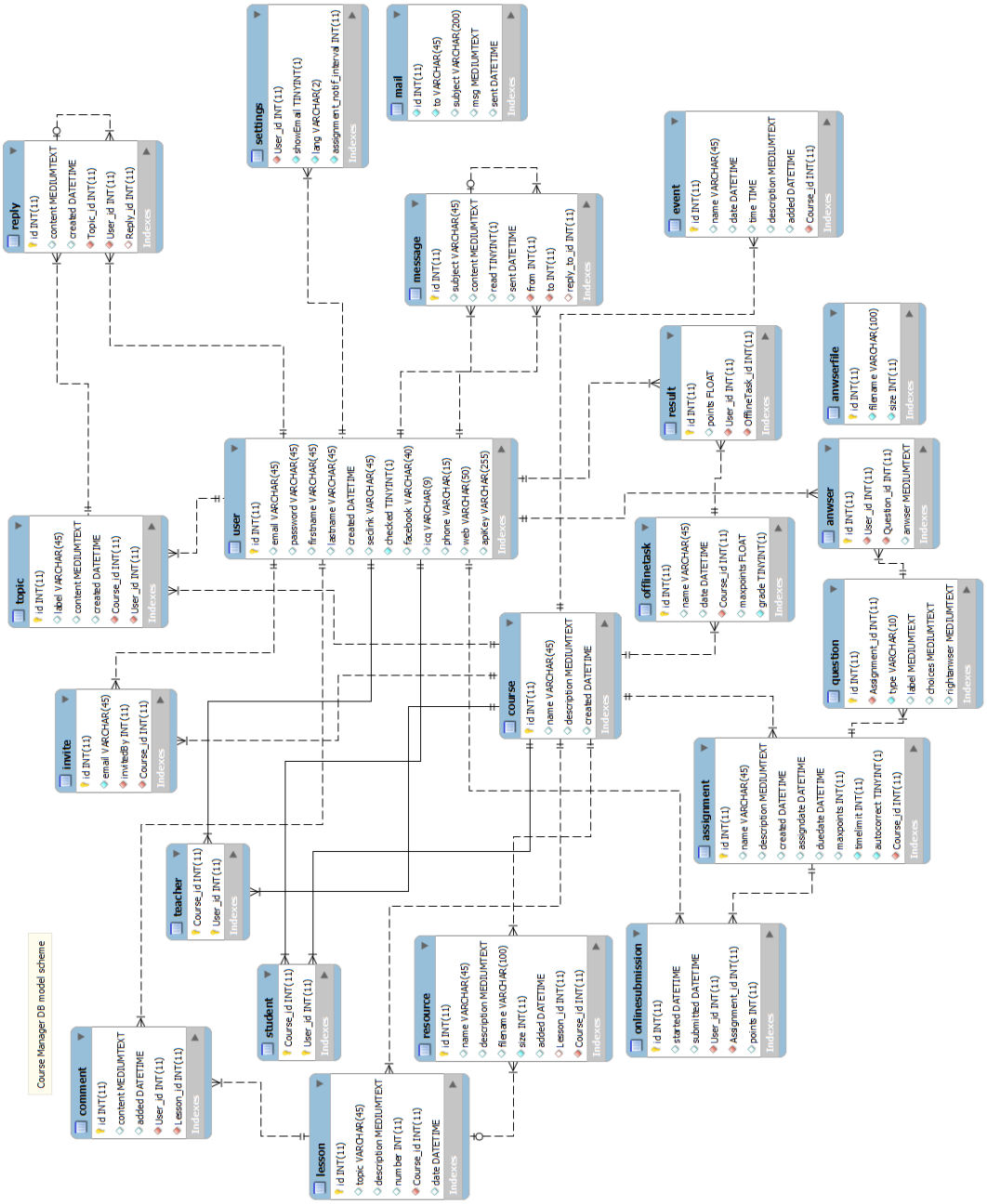
Mobilní verze, stejně jako ta webová, umožňuje kompletní komunikaci mezi uživateli aplikace pomocí systému soukromých zpráv. Modul Zprávy je rozdělen do dvou záložek - Doručené zprávy a Odeslané zprávy. Pomocí menu je možné vytvořit a následně odeslat zprávu novou. Samozřejmě lze i na příchozí zprávy jednoduše odpovídat.

Kapitola 11

Závěr

Kapitola 12

Přílohy



Obrázek 12.1: Schéma databázového modelu webové aplikace. Vygenerováno programem *MySQL Workbench*.

Kapitola 13

Seznam použitých zkratek

Literatura

[1] Android ™. http://android.com.

[2] Apache. http://apache.org.

[3] Dibi. http://dibiphp.com/.

[4] gettext. http://www.gnu.org/software/gettext/.

[5] jquery. http://jquery.com/.

[6] jquery ui. http://jqueryui.com/.

[7] Mysql. http://mysql.com.

[8] Nette framework. http://nette.org.

[9] Php. http://php.net.

[10] Soap. http://www.w3schools.com/soap/.

[11] Texy. http://texy.info/cs.

[12] Rfc 2109, http state management mechanism. [online], 1997. http://www.ietf.org/rfc/rfc2109.txt.

[13] Rfc 2818, http over tls. [online], 2000. http://tools.ietf.org/html/rfc2818.

[14] John Deacon. Model-view-controller (mvc) architecture. training, 1-6. [online], 2009. http://www.jdl.co.uk/briefings/MVC.pdf.

[15] Gail Rahn Frederick. Html5 and mobile web. [online], 2010. http://learnthemobileweb.com/2010/06/html5-and-mobile-web/.

[16] Sascha Schumann Chris Scollo Deepak Veliath Jesus Castagnetto, Harish Rawat. *Programujeme PHP profesionálně*. Computer Press, a.s., 2004.

[17] Martin Malý. Rest: architektura pro webové api. [online], 2009. http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/.