Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Webová PHP aplikace pro evidenci a organizaci individuálních třídních schůzek s možnostmi konfigurace a zabezpečeným přístupem

Karel Fiala

Vedoucí práce: RNDr. Helena Wallenfelsová

11. května 2014

Poděkování

Chtěl bych poděkovat vedoucí mé práce za odborný dohled.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 11. května 2014 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Fiala, Karel. *Webová PHP aplikace pro evidenci a organizaci individuálních třídních schůzek s možnostmi konfigurace a zabezpečeným přístupem.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstrakt

Tato bakalářská práce popisuje vývoj informačního systému, který si klade za cíl zefektivnit evidenci a organizaci individuálních třídních schůzek. Ve své práci popisuji požadavky na tento informační systém, mnou zvolené řešení, implementaci, současný stav aplikace a mé plány do budoucna.

Klíčová slova informační systém, software, databáze, zabezpečení, konfigurace, správa

Abstract

This thesis describes the development of an information system that aims to simplify the registration and the organization of individual class meetings. This thesis describes the requirements for the information system, the chosen solution, implementation, the current state of the application and my plans for the future.

Keywords information system, software, databases, security, configuration, administration

Obsah

Seznam obrázků

Úvod

Ve své bakalářské práci analyzuji potřeby aplikace pro evidenci a organizaci individuálních třídních schůzek. Aplikaci poté navrhnu a implementuji tak, aby byla konfigurovatelná a umožňovala zabezpečený přístup. Výslednou aplikaci otestuji a poté nasadím v reálném prostředí.

Aplikace bude distribuovatelná pomocí DEB balíčku, který bude kromě samotné aplikace také obsahovat manuálové stránky, počáteční konfiguraci a skripty pro jednoduchou správu z shellu pro případ nasazení aplikace na vlastním serveru.

Závěrem shrnu své poznatky z nasazení a nastíním plány o budoucím vývoji aplikace.

Kapitola 1

Rozbor zadání

- **Proveďte analýzu potřeb pro evidenci a organizaci individuálních třídních schůzek**
  Popíši důvody vzniku této aplikace a analyzuji potřeby, které by aplikace měla efektivně řešit, aby byla její existence přínosem.

- **Definujte potřebné konfigurace aplikace**
  Ukáži stěžejní body konfigurace a obecné požadavky na funkcionalitu. Dále se pokusím vžít do různých rolí a definovat tak konkrétní požadavky uživatelů aplikace.

- **Navrhněte strukturu webové aplikace a MySQL databáze pro ukládání informací**
  Zde odůvodním vybranou architekturu, jednotlivé aplikační vrstvy a také výběr a strukturu databázové technologie.

- **Vytvořte databázi a aplikaci**
  V této části krátce popíši implementaci architektury MVC a zmíním problémy na které jsem narazil během implementace databáze i aplikace a také jejich řešení.

- **Vytvořte instalační balíček pro Debian-based OS, který bude obsahovat aplikaci, manuálové stránky, počáteční konfiguraci a skripty pro základní práci s daty (import, export)**
  Nastíním instalaci aplikace a počáteční konfiguraci, etický důvod proč je třeba mít manuálové stránky a popíši skripty pro správu aplikace z shellu.

- **Navrhněte a realizujte zabezpečení aplikace**
  Popíši výhody a nevýhody použití HTTPS, obranu proti SQL injection a XSS, přístup do administrace aplikace a omezení přístupu do adresářů aplikace z webového prohlížeče.

- **Navrhněte uživatelské testy, proveďte testování a výsledky testování vyhodnoťte**
  Zde se dočtete o průběhu testování aplikace, jeho vyhodnocení a následné opravě chyb.

- **Vyhodnocení výsledků**
  Závěrem shrnu přínos této aplikace a nastíním její budoucí vývoj.

Kapitola 2

Analýza potřeb pro evidenci a organizaci schůzek

2.1 Rostoucí trend individualizace

S rostoucím trendem individuálního přístupu ke vzdělání vznikla i potřeba individuální komunikace mezi učitelem, žákem a rodičem. Smyslem tohoto přístupu je sblížit tuto trojici, pochopit a tak zefektivnit potřeby pro rozvoj vzdělání konkrétního žáka. Standardní třídní schůzky neposkytují čas ani soukromí pro řešení důležitých problémů. Většina času se věnuje pouze obecným informacím a poté se řeší spíše problémoví žáci. Dochází tím k nevědomému zazdívání potenciálních schopností schopnějších žáků. Rodič ví, že jeho dítě má výborný prospěch, ale již se téměř nedozví, v čem přesně jsou jeho slabé či naopak silné stránky. Učitel je s žáky několik hodin denně v kontaktu ve snaze rozvíjet jejich vzdělání a tak tyto informace zajisté má.

Individuální třídní schůzky se tedy snaží tuto problematiku řešit individuálním přístupem, kterým vnáší spoustu nových možností. Pro prospěchově slabší žáky je tak dostatek času, aby si rodič a učitel vysvětlili, kde přesně spočívá problém a to vše za přítomnosti žáka – dítěte. Naopak prospěchově silnějším žákům se díky individuálním třídním schůzkám dostává více pozornosti a jejich rodiče mohou s učitelem probrat konkrétní body a směr dalšího rozvoje vzdělání.

2.2 Průběh ITS a současné řešení situace

Z pravidla dvakrát ročně vedení školy rozhodne o konání individuálních třídních schůzek. S ohledem na svátky a školní akce vybere čtrnáctidenní období, kdy se tyto třídní schůzky budou konat. Sestrojí se univerzální tabulka, která se spolu s informacemi předá třídním učitelům.

Třídní učitelé si upraví tabulku dle svých potřeb, zvolí si délku jednoho

termínu a zanesou do tabulky termíny. Tabulky poté odevzdají vedení školy, které je zkontroluje a rozhodne, zda je pokrytí termíny vyhovující. Následně vedení školy tabulky opět upraví tak, aby měly jednotnou formu.

Tabulky se následně poskytnou rodičům k přihlašování na nástěnku školy. Rodiče se chodí postupně zapisovat na volné termíny do tabulky své třídy. V době vybraného termínu se rodič i se svým dítětem dostaví do školy a proběhne tak individuální třídní schůzka.

2.3 Problémy současného řešení

Toto řešení přináší řadu organizačních problémů. Mezi ty nejzávažnější pak patří, že rodič musí osobně do školy pro přihlášení na konkrétní termín a to tedy i v případě změny termínu. Osobní informace rodičů visí na nástěnce školy a tak jsou dostupné všem návštěvníkům školy. Tabulky je třeba kontrolovat, aby se rodiče vzájemně nezapisovali na cizí termíny, a proto se v tabulce nesmí škrtat.

Tabulku je třeba opakovaně přepisovat s každou provedenou změnou. Je také problém tabulky a jednotlivé termíny měnit během otevřených zápisů. Tyto problémy celou myšlenku znesnadňují a rodiče tak odrazují.

2.4 Klíčové funkcionality

Aplikace by měla poskytovat větší pohodlí, efektivní práci a měla by řešit minimálně výše zmíněné problémy. Rodič nebude muset osobně do školy, aby se zaregistroval na termín, ale bude moci provést registraci z pohodlí svého domova. Nikde nebudou zobrazeny osobní informace a vše proběhne zcela anonymně. Vedení školy by měla odpadnout nutnost opakovaně přepisovat tabulky a řešit případné kolize. Změny tabulek bude možné provést při otevřených registracích a jednotlivé tabulky půjdou tisknout.

2.5 Potřeby do budoucna

Je třeba, aby výsledná aplikace byla transparentní svým chováním a umožnila tak jednoduchý vývoj. Aplikace musí být schopna fungovat na obyčejném hostingu, ale také na vlastním serveru, který je do budoucna v plánu. Je také pravděpodobné, že obsluhu aplikace dříve či později přebere někdo jiný, a proto musí existovat podklady podle kterých bude nový správce schopen s aplikací dále pracovat a spravovat ji.

Kapitola 3

Potřebné konfigurace aplikace

3.1 Obecné požadavky na funkcionalitu

Aplikace by jako celek měla splňovat následující obecné požadavky funkčnosti.

- Aplikace bude dostupná přes internet pomocí webového prohlížeče.
- Aplikace umožní nasazení na vlastním serveru i na obyčejném hostingu.
- Část aplikace pro registrace by měla být velmi jednoduchá na ovládání.
- Veřejně nebudou dostupné žádné osobní informace a v případě nasazení na vlastním serveru půjde využít zabezpečené komunikace pomocí protokolu HTTPS.
- Aplikace bude veřejná, otevřená všem, protože není v možnostech školy všem rodičům rozdávat hesla, aby se mohli přihlásit.
- V případě nasazení na vlastním serveru bude možné využít jednoduchou správu aplikace z shellu.

3.1.1 Požadavky z pohledu rodiče

Rodič bude přistupovat k aplikaci přes internet pomocí webového prohlížeče a měl by mít následující možnosti.

- Zobrazit si tabulku s termíny své třídy, kde uvidí volné, obsazené, prozatím rezervované a nedostupné termíny.
- Vybrat si volný termín a podat si na něj přihlášku.

3.1.2 Požadavky z pohledu školy

Škola bude v aplikaci spravovat třídy, termíny a přihlášky. K aplikaci bude přistupovat také přes internet pomocí webového prohlížeče a měla by mít následující možnosti.

- Zobrazit, přidat, přejmenovat a smazat *(skrýt)* třídy.
- Zobrazit, přidat a smazat *(skrýt)* termíny jednotlivých tříd.
- Zobrazit, schválit a smazat *(skrýt)* přihlášky na vypsané termíny.
- Zobrazit a vytisknout přehledy přihlášek ve formě tabulky termínů každé třídy.
- Zobrazit statistiku obsazených/volných termínů.

3.1.3 Požadavky z pohledu správce aplikace

Správce aplikace může, v případě nasazení aplikace na vlastním serveru, přistoupit k aplikaci pomocí shellu. Měl by mít následující možnosti.

- Zobrazit manuálové stránky aplikace.
- Smazat celou databázi.
- Vytvořit novou, čistou, databázi.
- Exportovat data z databáze do CSV souborů – *jeden pro každou tabulku databáze.*
- Importovat data z CSV souborů do databáze – *jeden pro každou tabulku databáze.*
- Pročistit databázi – smazat skryté *(z administrace smazané)* záznamy.
- Zabalit aktuální verzi aplikace do nového DEB balíčku – *vhodné pro další vývoj.*

Kapitola 4

Návrh struktury a technologií

4.1 Struktura aplikace

4.1.1 Architektura MVC

Pro potřeby flexibility budoucího vývoje aplikace se mi jako nejlepší jeví použití jednoduché třívrstvé architektury Model-View-Controller. Architektura MVC je jednoduchá na implementaci a přitom zcela oddělí důležité části kódu. Toto chování mi do budoucna umožní změnit vzhled aplikace bez přepisování ostatních částí aplikace, čehož pravděpodobně využiji pro budoucí mobilní verzi. Stejně tak je možné měnit jednotlivé business procesy a přitom zcela zachovat vzhled a řízení aplikace. Grafické znázornění architektury MVC je vidět na obrázku 4.1.

4.1.1.1 Model

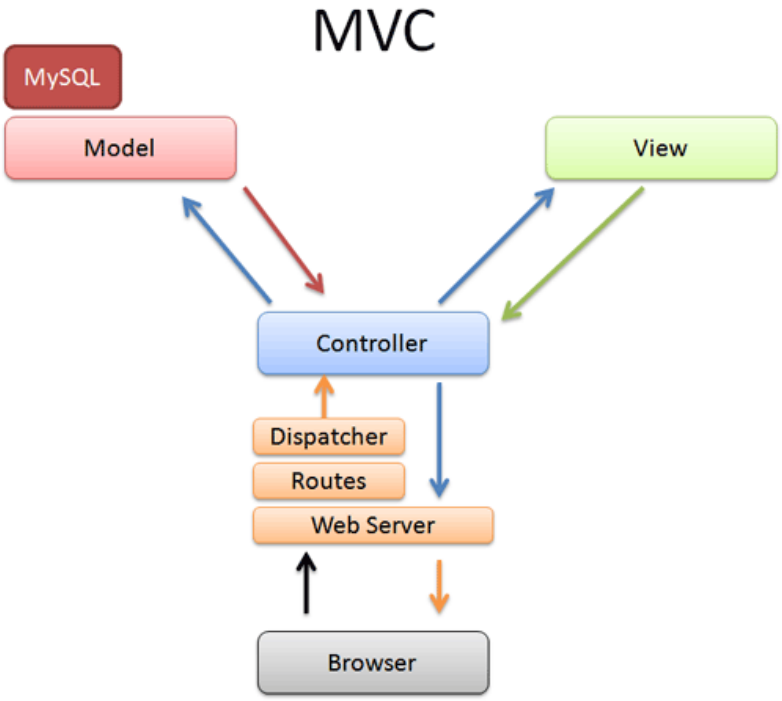
Modelem rozumíme takovou část aplikace, která implementuje pouze business logiku celé aplikace. Model by tak měl být zcela nezávislý na tom, jaký bude grafický výstup aplikace a neměl by se starat o zpracování požadavků. Model často zaštiťuje konkrétní datovou strukturu a provádí nad ní potřebné operace.

4.1.1.2 View

View se naopak stará pouze o vykreslení dat do určité šablony. Často bývá implementován jako knihovna a šablony vzhledu, které knihovna zpracuje. Každá šablona očekává určitá data v podobě proměnných a různá nastavení, podle kterých se řídí samotné vykreslení.

4.1.1.3 Controller

Kontrolér je poté pojítko mezi **view** a jednotlivými **model**y. Je tak hlavním organizátorem všech akcí. Veškeré požadavky přijdou na kontrolér a ten roz-



Obrázek 4.1: Schéma architektury MVC[1]

hodne, jaké metody modelů se zavolají a jaké šablony bude **view** vykreslovat. Po zpracování celého požadavku předá kontrolér výsledek – zde HTML kód – zpět klientovi, který se na něj dotázal.

4.1.2 Framework

Dále jsem se rozhodl, že v aplikaci nepoužiji žádný existující framework a to z následujících důvodů.

- Naučit se efektivně použít framework by pro mě bylo časově náročné a do kódu bych pravděpodobně zanášel zbytečné použití jiných funkcí.
- Výsledná aplikace by byla „složitější“, rozsáhlejší a zcela závislá právě na použitém frameworku.
- Chtěl jsem mít celou aplikaci plně pod kontrolou. V případě použití frameworku bych musel procházet jeho vlastní kód, abych zjistil, co přesně se vykonává v místě volání.
- Skromně si myslím, že pokud vývoj či správu aplikace převezme někdo jiný, tak skutečnost, že je vše „čistě“ napsané bez použití frameworku

pro něj bude příjemnější, než když se dozví, že by se musel rychle přiučit frameworku, který nepoužívá nebo dokonce považuje za nevhodný.

4.2 Databázová technologie

V databázi budou uloženy aplikační data. Je třeba zaznamenávat informace o třídách, termínech pro každou třídu a jednotlivé přihlášky na konkrétní termíny. Záznamy by se z databáze neměly mazat, ale jen skrývat, aby bylo možné je v případě potřeby vrátit nebo například vynášet statistiku toho, jaké termíny jsou nejžádanější.

4.2.1 Databázová vrstva

Kromě třívrstvé architektury MVC by aplikace měla obsahovat další oddělenou databázovou vrstvu pro přístup k samotné databázi. Tato vrstva by měla implementovat takový interface, aby bylo možné všechny vrstvy implementující tento interface zaměnit a jednoduše tak vyměnit použitou databázovou technologii. Toto chování umožní do budoucna přechod aplikace z databázové technologie MySQL na jinou databázovou technologii.

4.2.2 Zjednodušené schéma

Pro potřeby aplikace jsem použil zjednodušené databázové schéma na obrázku 4.2, které jsem modeloval v nástroji **MySQL Workbench**. Schéma ukazuje sloupce jednotlivých tabulek a vzájemnou provázanost tabulek cizími klíči.

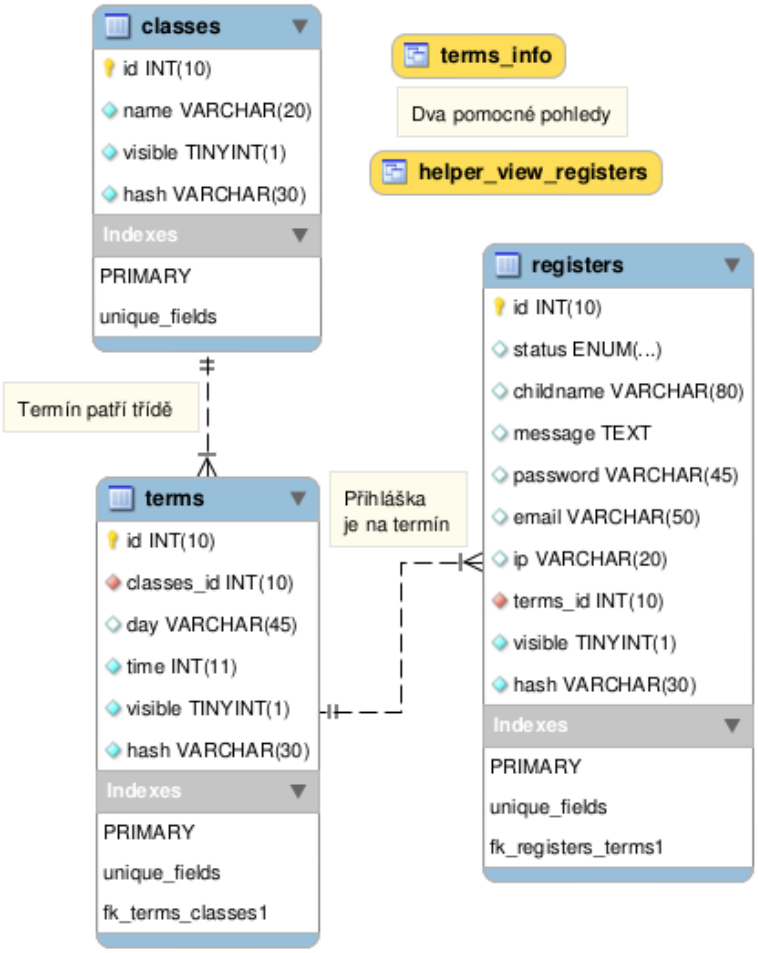
Žluté „tabulky“ jsou pohledy, kde *helper_view_registers* je pomocný pohled se selekcí pro *terms_info*. Tento pohled je triviální ale nutný, protože MySQL nedovoluje použít poddotaz v klauzuli FROM při vytváření rozsáhlejšího pohledu.

4.2.3 Integritní omezení

Databáze by měla obsahovat dostatek integritních omezení, aby byla zaručena konzistence uložených dat. Proto jsem zvolil tyto integritní omezení.

- **Na jeden termín může existovat pouze jedna viditelná *(nesmazaná – neskrytá)* přihláška.**
  Toto integritní omezení zaručuje, že nedojde k situaci ve které by dva rodiče měli rezervovaný jeden stejný termín.

- **Nemůže existovat více zcela stejných *(třída, den, hodina)* a viditelných termínů.**
  Toto integritní omezení zaručuje, že se pod jedním termínem tabulky



Obrázek 4.2: Použité databázové schéma

nebudou skrývat dva termíny, což by mohlo způsobit další logickou nekonzistenci v tabulce přihlášek. Mohly by totiž existovat dvě přihlášky na dva různé, ale přesto zcela identické termíny.

- **Nemůže existovat více tříd se stejným názvem.**
  Toto integritní omezení je ochrana před duplicitním pojmenováním tříd, které by pak mohlo způsobit nechtěné vložení termínů do jiné třídy.

4.2.4 MySQL

MySQL je velmi rozšířená databázová technologie, kterou nalezneme téměř na každém hostingu. Poskytuje všechny potřebné náležitosti pro běh aplikace a samotné použití této technologie by tak přineslo možnost nasazení celé aplikace na školním hostingu.

4.2.5 PostgreSQL

PostgreSQL je sice o něco méně rozšířené než MySQL, ale aplikaci by poskytlo „profesionálnější“ zázemí pro vlastní data. Umožňuje použít PL/SQL, kterým je možné rozšířit databázi o vlastní procedury a funkce. Také je možné napsat vlastní triggery, které by mohly vhodně rozšířit možnosti databázové vrstvy aplikace.

4.3 Použité technologie

Po důkladné analýze jsem dospěl k závěru, že následující vybrané technologie mi umožní napsat efektivní aplikaci, která poslouží svému účelu a bude vhodně rozšířitelná do budoucna.

4.3.1 Aplikace

Protože dle požadavků je třeba, aby byla výsledná aplikace dostupná přes internet z webového prohlížeče a to zcela bez instalace, tak jsem jako vhodnou technologií pro samotnou aplikaci zvolil PHP, které bude generovat HTML stránky s obsahem. O grafickou úpravu aplikace se postará CSS a pro rozšířenou interaktivitu s uživatelem použiji jazyk JavaScript.

4.3.2 Databáze

Jako vhodnou databázovou technologii jsem zvolil velmi rozšířené MySQL, na kterém aplikace poběží na školním hostingu. MySQL je tedy výchozí databázová technologie, které jsem celou aplikaci přizpůsobil.

Nakonec jsem se však také rozhodl samotnou aplikaci již nyní portovat i pro použití databáze PostgreSQL, která by do budoucna umožnila přesunout některou business logiku aplikace přímo do databázového stroje. Cílem je rozšířit konfigurovatelnost aplikace a připravit aplikaci k dalšímu vývoji.

Kapitola 5

Implementace

5.1 Použité prostředí

Pro vývoj aplikace jsem použil Linuxovou distribuci vycházející z distribuce Ubuntu, která si také nese mnoho prvků z distribuce Debian. Tato distribuce mi tak poskytla zázemí pro všechny potřebné nástroje jako je například Apache2, PHP5, MySQL, PostgreSQL, NetBeans.

Implementaci aplikace jsem prováděl ve vývojovém prostředí NetBeans, který kromě jazyku Java také podporuje vývoj pro jazyk PHP. Během psaní kódu nabízí případnou nápovědu a díky tomu byla implementace rychlejší a následná oprava chyb snazší.

5.2 Implementace databáze

5.2.1 Databáze MySQL

Návrh databáze jsem vytvořil v MySQL workbench a exportoval jako SQL skript, který jsem importoval do databázového stroje. Pro snazší práci s databází jsem si chtěl vytvořit pohledy, ale byl jsem nemile překvapen, že MySQL neumí vytvořit pohled, který by obsahoval v klauzuli FROM poddotaz. Proto ve finální verzi existuje ještě druhý pohled, který supluje funkci poddotazu v hlavním pohledu.

Při implementaci aplikace jsem udělal několik změn v databázi, které jsem poté zanesl zpět do schématu pomocí funkce reverzního převodu databáze do schématu v programu MySQL WorkBench. Mezi tyto změny patřily právě vytvořené pohledy a úprava unikátní klíčů kvůli integritním omezením.

5.2.2 Databáze PostgreSQL

Databázi pro PostgreSQL jsem chtěl vytvořit exportem MySQL databáze s příznakem pro kompatibilitu a dodržení standardu. Bohužel tato cesta byla

zcela beznadějná i přesto, že při importu do PostgreSQL jsem použil příznak pro import ze souboru exportovaného z databázového stroje MySQL.

Po několika marných pokusech různých kombinací příznaků importu a exportu a použití skriptů pro převod mezi databázemi, jsem nakonec soubor ručně procházel, zaměňoval jednotlivé datové typy, které jsou mezi databázemi odlišné a opravoval nekompatibilitu dvojitých uvozovek u databáze PostgreSQL.

Pro další vývoj aplikace by bylo vhodné napsat vlastní skript, který by dokázal řešit převod databáze mezi těmito databázovými technologiemi zcela automaticky.

5.3 Implementace aplikace

5.3.1 Připojení k databázi

PHP umožňuje připojení k databázi pouze vestavěnými funkcemi nad kterými bylo třeba napsat vlastní databázovou vrstu. Tyto vrstvy – pro MySQL i PostgreSQL – implementují jeden interface o který se opírají jednotlivé modely aplikace. Díky tomu by například bylo možné napsat další vrstvu pro databázový stroj Oracle a pak pouhou záměnou této vrstvy s vrstvou pro databázi MySQL by celá aplikace mohla dále využívat databáze Oracle.

5.3.2 SQL dotazy na databázi

Problémem změny databázové vrstvy by mohly být samotné SQL dotazy na databázi. Při použití jiné databázové vrstvy, která například neimplementuje použití funkce *MD5*[1] by bylo třeba přepsat tyto dotazy v souboru *sql.php*. Každá databázová vrstva proto implementuje metodu, ve které vrátí svůj *identifikátor*, podle kterého se třída v souboru *sql.php* může rozhodnout, zda provede v dotazu změnu pro danou databázovou technologii.

Jiným, určitě hezčím, ale také výpočetně náročnějším řešením by bylo použít třídu, která by dotazy sestavovala dynamicky na základě použité databázové technologie. Takové třídě se říká *query builder*. Použití této třídy by v aplikaci bylo možné tak, že třída v souboru *sql.php* by místo jednotlivých dotazů obsahovala ekvivalentní volaní třídy *query builder*, která by již podle použité databázové technologie vracela SQL dotazy. Toto řešení jsem však zavrhl, protože v současné době neplánuji dále vyvíjet aplikaci pro jiné databázové technologie než je MySQL a PostgreSQL.

5.3.3 Základ pro MVC

Dalším krokem byla implementace základu pro zvolenou architekturu MVC. Při implementaci jsem narazil na mnoho komplikací a byl jsem nucen kód ně-

[1] hašovací funkce

kolikrát přepisovat, než jsem se dostal do takové podoby, abych byl spokojený.

5.3.3.1 Magická funkce autoload()

Pro načítání dosud nenačtených tříd jsem použil magickou funkci *autoload()*, která se automaticky zavolá pokud se aplikace pokusí použít dosud nenačtenou třídu. Zavolaná funkce si načte jednotlivé cesty, kde jsou uloženy třídy a pokusí se v nich vyhledat a načíst tu, která ještě není načtená.

5.3.3.2 Implementace třídy Loader

Každý požadavek, který na aplikaci přijde je přepsán pomocí souboru *.htaccess* tak, aby se vždy spustil soubor *index.php* s parametrem **q**. Ten zavede potřebné konstanty, nastaví prostředí, vytvoří instanci objektu Loader a zavolá jeho metodu *start()*. Nyní již převezme řízení aplikace Loader, který rozparsuje vstupní argumenty a na jejich základě rozhodne, který kontrolér a jeho metodu zavolá.

5.3.4 Implementace kontrolérů

Implementoval jsem dva kontroléry pro řízení požadavků – jeden pro část s přihlašováním a druhý pak pro celou administraci. Každý kontrolér dědí abstraktní třídu *Controller*, která implementuje základní chování každého kontroléru a poskytuje tak sdílené metody jako základ.

Každý kontrolér si pak hlídá ty parametry u kterých reaguje změnou svého chování a tak parametry určené pro zavolaný *model* zpravidla nekontroluje.

5.3.4.1 Kontrolér pro rezervace

Kontrolér pro obsluhu veřejně přístupné části aplikace má na starost zobrazení volných termínů a obsluhu nových rezervací, které rodiče provedou. I přesto, že kontrolér je tak malý, tak jeho existence vnáší větší přehlednost do samotného kódu a jeho členění.

5.3.4.2 Kontrolér pro administraci

Tento kontrolér je značně obsáhlejší než jeho bratříček, protože administrace obsahuje více možností než samotné rezervace na termíny. Kontrolér také pro každou akci požaduje potvrzení přihlášení do administrace pomocí **HTTP/1.0 Basic Auth**[2]. Tak je pro každou administrativní akci zajištěná jednoduchá autentifikace uživatele.

[2]označení pro jednoduchou autentizaci přístupu

5.3.5 Implementace modelů

Modely obsahují business logiku celé aplikace a starají se o práci s daty v databázi ke které přistupují pomocí databázové vrstvy. Každý model si hlídá vstupní parametry a v případě chyby tak vrací hodnotu *false*.

Každý model dědí univerzální a společné metody z abstraktní třídy *Model*, kterou dále rozšiřuje svými metodami. V aplikaci používám tři modely pro řízení všech akcí tříd, termínů a registrací. Tyto tři modely mi tak zastřešují jednotlivé databázové tabulky.

5.3.6 Implementace view

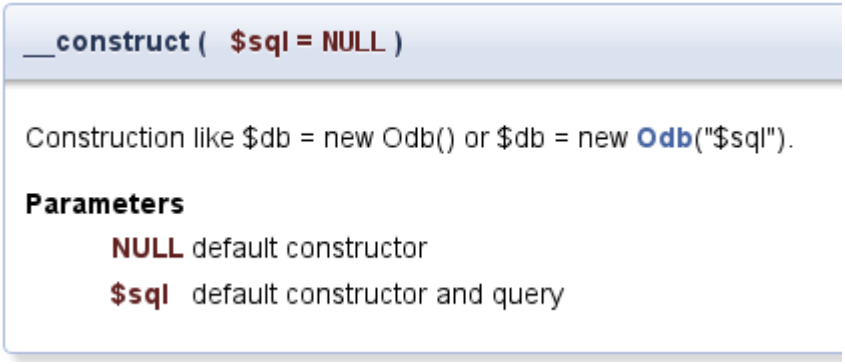
View jsou v podstatě jen vzhledové šablony, které vypisují proměnné do kterých se při zpracování uloží data. Šablony načítá a zpracovává knihovna pro vykreslení, která je zavolána kontrolerem ve chvíli kdy je již jasné, že jsou všechna potřebná data k vykreslení zpracována.

Šablona může vložit do svého těla jinou šablonu a je tak možné stejné části kódu vyčlenit do jednoho společného souboru. Na počátku jsem implementoval speciální značky, které zpracovávala šablonovací knihovna tak, že je nahradila příslušným kódem z jiné šablony. Nicméně od toho to nápadu jsem brzy upustil, protože výsledné vykreslování bylo zbytečně náročné a často neposkytovalo dostatečnou flexibilitu. Místo toho samotné šablony obsahují části PHP kódu pro řídící struktury a jsou tak schopny samy do sebe vložit jinou šablonu. Příkladem vkládání může být identická hlavička stránky a řídící strukturou například zpracování informace o provedení akce jako je uložení termínu.

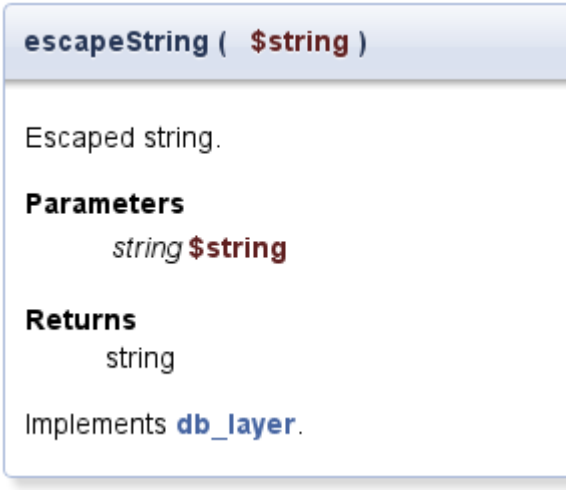
5.3.7 Dokumentace kódu

Zdrojový kód je dokumentován, protože tato dokumentace poté usnadňuje další vývoj. Navíc vývojové prostředí jako je NetBeans dokáže číst tuto dokumentaci a při psaní dalšího kódu je tak možné ze své vlastní dokumentace číst poznámky.

Nad celým zdrojovým kódem je poté vytvořena podrobná dokumentace pomocí programu **Doxygen**, který přiložil i grafy volání, hierarchie a také jednotlivé zdrojové soubory. Ukázka dokumentace je na obrázcích 5.1 a 5.2. Zde jsou také dobře vidět drobné nedostatky. Některé části dokumentace nebyly kvůli špatné syntaxy Doxygenem rozpoznány nebo nemusejí být zcela jasné. Dále tedy navrhuji projít a zdokumentovat některé části kódu lépe ještě dříve, než naroste jeho množství.



Obrázek 5.1: Příklad dokumentace vygenerované programem Doxygen



Obrázek 5.2: Druhý příklad dokumentace vygenerované programem Doxygen

Kapitola 6

Instalace, manuálové stránky a skripty

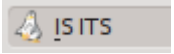
6.1 Vlastní instalace aplikace

Instalace aplikace se provádí v shellu z DEB balíčku pomocí příkazu **sudo dpkg -i <jmeno-balicku.deb>**. Instalace si ohlídá nutné závislosti a v případě chybějící závislosti bude instalace včas ukončena. Při instalaci z DEB balíčku není totiž možné automaticky stahovat nutné závislosti z repozitářů a tak je lze opravit až příkazem **sudo apt-get upgrade -f**. Po nainstalování chybějících závislostí již bude instalace aplikace pokračovat. Ke konci instalace se spustí poinstalační skript, který provede základní nastavení aplikace a vytvoří novou databázi.

Po úspěšné instalaci aplikace máme v shellu dostupné dva nové příkazy a to **itsadm** a **itsadm-createdeb** pokud jsme instalovali vývojovou verzi. V případě použití grafické nadstavby se nám v menu spuštění pod záložkou *Internet* vytvořil také odkaz **IS ITS** s ikonkou tučňáka, který spouští firefox s lokální adresou pro přístup k aplikaci. Ikona spouštěče je vidět na obrázku 6.1.

6.2 Odinstalace aplikace

Aplikaci můžeme odinstalovat pomocí příkazu **sudo dpkg -r its**. Budou odebrány všechny nainstalované soubory, nastavení a změny, které se během instalace provedly.



Obrázek 6.1: Ikona spouštěče aplikace

6.3 Manuálová stránka

Instalací z DEB balíčku se nainstaluje také manuálová stránka aplikace. Ta vysvětluje základní funkčnost aplikace, nástroje určené pro správu a několik možných scénářů. Lze podle ní nahrát čistou databázi či vytvořit nový DEB balíček.

Myslím si, že manuálová stránka a dokumentace je pro každou aplikaci nutnost, která by měla být zajištěna etikou vývojáře aplikace. Díky této manuálové stránce může kompletní správu aplikace převzít i člověk, který aplikaci nevyvíjí a přesto potřebuje udělat určité změny. Bez manuálové stránky by byla aplikace předem odsouzena k záhubě ihned po změně správce. Bohužel někteří vývojáři a některé firmy toto nedodržují a nutí tak své zákazníky k placené správě.

Instalace aplikace předpokládá defaultní nastavení všech závislých balíků a proto se například pro přihlášení k databázi MySQL využívá účet **root** bez hesla. Tyto parametry může být třeba změnit a právě v případě nasazení na vlastní server je změna tohoto chování doporučena. Tyto informace jsou také součástí manuálových stránek.

6.4 Skripty pro vytvoření DEB balíčku

Po nainstalování vývojové verze aplikace je dostupný nový příkaz **itsadm-createdeb**, který je schopen současnou podobu aplikace kdykoliv znovu zabalit a vytvořit z ní nový DEB balíček pro další distribuci aplikace. Skript vytvoří dočasný adresář, kam nakopíruje současnou podobu aplikace, manuálové stránky, skript pro práci s aplikací, konfiguraci webového serveru Apache2 a další potřebné části. Adresář poté zpřístupní k zápisu pro provedení změn – například kvůli změně verze, popisu, přihrání dalších souborů. Až jsou změny hotové, stačí jen pokračovat libovolnou klávesou a skript tak vytvoří nový DEB balíček. Ukázka je na obrázku 6.2.

Takto vytvořený DEB balíček lze opět zcela stejně nainstalovat včetně všech jeho přidaných částí, které budou opět zahrnuty do dalšího vytvořeného balíčku, pokud se nevymykají svým umístěním v balíčku. V případě přidání zcela nové části je třeba tuto změnu také zanést do skriptu pro vytváření nového DEB balíčku.

Skript **itsadm-createdeb** implementuje jeden nepovinný parametr **no-dev**, který umožní výslednou aplikaci zabalit jako nevývojovou verzi, tedy nebude obsahovat právě skript pro vytvoření nového balíčku a ostatní k tomu nutné části. Tento postup je vhodný pokud chceme aplikaci předat někomu, kdo ji bude jen používat a nebude ji dále vyvíjet.

```
$ itsadm-createdeb

Připravuji soubory pro vytvoření balíčku ...[ok]

Soubory jsou připraveny, nyní můžete provést ruční
změny v adresáři: /tmp/tmp.WETZJfdeYT

Pro pokračování stiskněte libovolnou klávesu ...
dpkg-deb: building package 'its' in 'its-fialaka1.deb'.
Balíček byl vytvořen.
```

Obrázek 6.2: Tvorba nového DEB balíčku

6.5 Skripty pro práci s daty

Aplikace obsahuje skript **itsadm**, který umožňuje.

- Zobrazit všechny tabulky z vybrané databáze
  – pro kontrolu, zda existuje databáze a zda obsahuje správné tabulky
- Zobrazit všechny záznamy z tabulek vybrané databáze
  – pro kontrolu, zda tabulky obsahují data
- Exportovat celou vybranou databázi do CSV souborů
  – proběhne export všech tabulek do jednotlivých CSV souborů
- Smazat celou vybranou databázi
  – nenávratně odstraní celou databázi
- Vytvořit celou databázi
  – vytvoří zcela novou databázi včetně všech tabulek
- Importovat data z CSV souborů
  – vyzve k importu dat ze souborů ve stejném CSV formátu jako probíhá export
- Pročistit databázi
  – odstraní smazané *(skryté)* záznamy z databáze a uvolní tak místo

Tento skript nečte nastavení databáze v aplikaci, ale vždy vyzve uživatele o zadání údajů pro přístup do správné databáze.

Kapitola 7

Zabezpečení aplikace

7.1 Zabezpečená komunikace – HTTPS

Standardní HTTP protokol neposkytuje žádné zabezpečení komunikace, která se přenáší mezi mezi serverem a klientem. Může tak snadno dojít k odposlechnutí této komunikace a zneužití osobních informací v ní obsažené. Taková skutečnost by byla pro školu velmi nepříjemná. Je proto třeba přenášené informace vhodně chránit. K tomuto účelu existuje protokol HTTPS, což je vlastně standardní HTTP protokol, který se přenáší po zabezpečené vrstvě pomocí SSL. Taková to komunikace je dostatečně zabezpečena proti zneužití osobních údajů v ní obsažené. Ukázka komunikace pomocí HTTPS je na obrázku 7.1.

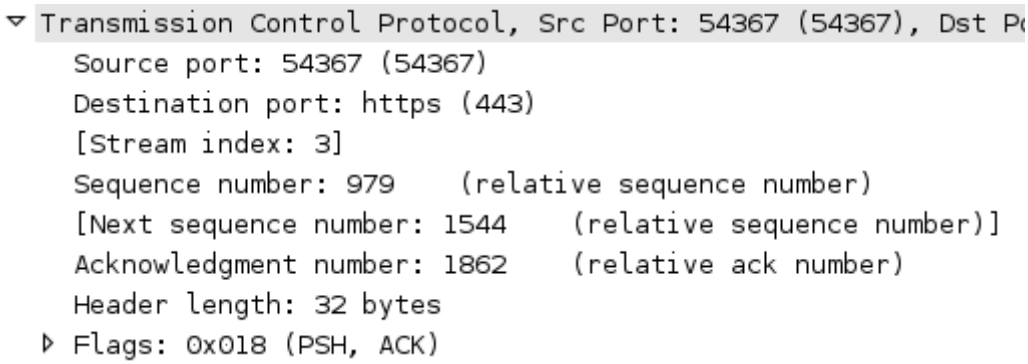
7.1.1 Výhody HTTPS

Kromě zvýšení bezpečnosti komunikace z hlediska přenosu osobních údajů, nám použití HTTPS přináší i ochranu proti náhodnému podvržení paketů, protože každá HTTPS komunikace má kontrolní mechanismy pro řízení samotné komunikace a prosté opakování zachyceného paketu bude vyhodnoceno jako chyba komunikace.

Další drobnou výhodou HTTPS je, že aplikace pak působí profesionálnějším dojmem a škole to tak dělá zase o něco lepší jméno. Myslím si, že málo škol nabízí rodičům individuální třídní schůzky, mají na to vlastní rezervační aplikaci a komunikaci zabezpečují školním certifikátem.

7.1.2 Nevýhody HTTPS

Velkou nevýhodou HTTPS je větší datová náročnost oproti samotnému HTTP protokolu. Tato datová náročnost je způsobena počáteční výměnou šifer a samotným šifrováním během komunikace. Pokud se na stránku aplikace opakovaně podívám pomocí HTTP protokolu, tak se přenese přibližně 4,5 kB



Obrázek 7.1: HTTPS komunikace v programu Wireshark

dat. Pokud na stejnou stránku opakovaně přistoupím pomocí HTTPS, tak se přenese přibližně 8,6 kB dat. Důležitým faktem také je, že se během komunikace pomocí HTTPS přeneslo mnoho malých paketů navíc oproti klasickému HTTP.

Komunikace pomocí HTTPS s sebou také přináší ztížené možnosti kešování obsahu stránky pomocí proxy serveru mezi klientem a serverem. Toto další zpomalení spolu s datovou náročností přispěje tak k tomu, že se celkově aplikace využívající HTTPS bude jevit pomalejší.

Další nevýhodou je fakt, že certifikát musí být podepsaný uznávanou certifikační autoritou, jinak by se klientům tvářil jako neověřený a jeho použití by mělo spíše odrazující účinek. Za takový certifikát je třeba zaplatit nemalou částku.

7.2 Cross-site scripting

7.2.1 Definice XSS

Cross-site scripting (XSS) je metoda narušení WWW stránek využitím bezpečnostních chyb ve skriptech (především neošetřené vstupy). Útočník díky těmto chybám v zabezpečení webové aplikace dokáže do stránek podstrčit svůj vlastní javascriptový kód, což může využít buď pouze k poškození vzhledu stránky, jejímu znefunkčnění anebo dokonce k získávání citlivých údajů návštěvníků stránek, obcházení bezpečnostních prvků aplikace a phishingu. [2]

7.2.2 Obrana proti XSS

Proti XSS se lze v PHP bránit tak, že každý výstup vypisovaný do stránky, jehož vstup byl dříve neošetřený, ošetříme pomocí funkce **htmlspecialchars()**, která zajistí nahrazení HTML značek za jejich textové vyjádření tak, aby je prohlížeč neinterpretoval jako HTML, ale pouze jako text k zobrazení.

Například tag `<script>` se po použití funkce *htmlspecialchars()* změní na `&lt;script&gt;`. Tím tak zaručíme, že žádný výstup neovlivní vlastní chování aplikace.

7.3 SQL injection

7.3.1 Definice SQL injection

SQL injection je technika napadení databázové vrstvy programu vsunutím (odtud „injection“) kódu přes neošetřený vstup a vykonání vlastního, samozřejmě pozměněného, SQL dotazu. Toto nechtěné chování vzniká při propojení aplikační vrstvy s databázovou vrstvou (téměř vždy se totiž jedná o dva různé programy) a zabraňuje se mu pomocí jednoduchého escapování potencionálně nebezpečných znaků. [3]

Útočník například vnutí svůj kód do parametru, který má udávat číslo třídy. Pokud tento parametr nebude ošetřen, tak se například místo jednoho SELECT dotazu nad databází může vykonat celá sekvence příkazů, která je uložená v proměnné, kde původně mělo být pouze číslo třídy.

7.3.2 Obrana proti SQL injection

Obranou proti SQL injection je provést tzv. *escape string*, tedy znaky v proměnné, které by mohly způsobit změnu původního dotazu, ještě před vložením do dotazu změníme tak, aby se již dále nemohly interpretovat jako řídíci znaky dotazu, ale pouze jako jeho data, parametry.

Pokud bychom tedy místo čísla třídy vložili do dotazu jiný dotaz *(za předpokladu, že číslo třídy neprochází validací, zda se opravdu jedná o číslo a ne řetězec)* a při vkládání použili funkci pro *escape string*, tak by nám databáze buď nevrátila žádný řádek nebo rovnou chybu dotazu. Toto chování je již akceptovatelné, protože nezpůsobí žádné další škody. Útočník se dozví například jen to, že tato třída neexistuje.

7.3.2.1 Řešení MySQL

MySQL pro *escape string* používá funkci/metodu **real_escape_string()**. Funkce/metoda nám vrátí řetězec, který je bezpečný pro použití v SQL dotazu. Avšak je třeba si dát pozor, protože to tak funguje pouze v případě, že si předtím dohodneme správné kódování, které použijeme při komunikaci s databází. Kódování je vhodné nastavit hned po připojení k databázi pomocí **set_charset('utf8')**.

7.3.2.2 Řešení PostgreSQL

PostgreSQL má stejné řešení, ale není třeba nic předem nastavovat. Stačí každý nebezpečný parametr, který posíláme v SQL dotazu zabezpečit násle-

dující funkcí **pg_escape_string()**, kterou nám PHP poskytuje.

7.4 Zabezpečení dat aplikace

7.4.1 Webserver Apache2

V konfiguraci webserveru je aplikaci přidělen přístup pouze na svůj datový adresář a je povoleno užití souboru .htaccess, který se stará o přepis parametrů v URL a zabezpečení adresáře aplikace.

HTTPS komunikace je pak zabezpečena tzv. self-signed certifikátem samotného webserveru. Je to tak z důvodu, že certifikát může být pouze jeden jediný pro celý webserver, protože SSL pracuje s IP adresami a nerozezná jednotlivé virtuální hosty na webserveru.

Chyby a přístupy k aplikaci se pak logují do svého vlastního logu, aby bylo možné snadno řešit případné problémy.

7.4.2 Soubor .htaccess

Soubor .htaccess používá aplikace k přepisu URL adres a k zabezpečení jednotlivých adresářů. K těmto adresářům pak nelze přistupovat skrz URL prohlížeče. Jediný adresář ze kterého lze číst obsah je adresář **www**, kde jsou uloženy soubory potřebné pro vzhled a JavaScriptové knihovny.

Soubory .htaccess jsem použil proto, že jejich podpora na hostingu je celkem rozšířená a je tak možné beze změny použít stejně se chovající prostředí na lokálním serveru i na hostingu. To by při nastavení přímo v konfiguraci Apache2 nebylo možné.

7.4.3 HTTP Basic Auth

7.4.3.1 Definice HTTP Basic Auth

Basic access authentication (v překladu jednoduché ověření přístupu) je v informatice označení pro jednoduchou autentizaci při přístupu na webové stránky. Webový server vyzve pomocí protokolu HTTP přistupujícího klienta (typicky webový prohlížeč), aby poslal v rámci požadavku na stránku také autentizační informace (tj. jméno a heslo).[4]

7.4.3.2 Použití

Autentizace HTTP/1.0 Basic Auth je použitá pro přístup do administrace. Odpadlo tak řešení tabulky uživatelů, přihlašování, práva a podobné. Toto zjednodušení jsem si dovolil protože informační gramotnost učitelů na škole není dostatečná k tomu, aby si spravovali své termíny. Vedení školy tak chtělo striktně dodržet jediný přístup do administrace pro kompletní správu.

Je důležité poznamenat, že použití HTTP Basic Auth neposkytuje žádné zabezpečení dat při přenosu. Odposlechnutí komunikace tak znamená vyzrazení přihlašovacích údajů. Jde v podstatě o stejný problém, který by se vyskytl v případě přihlašování pomocí přihlašovacího formulář bez zabezpečené komunikace. Formulář sice neodesílá přihlašovací údaje s každou stránkou, ale pro udržení přihlášení je třeba posílat například identifikátor session, který je uložen v cookies prohlížeče. V obou případech lze tuto bezpečnostní mezeru vyřešit použitím HTTPS.

7.4.4 Připojení k databázi

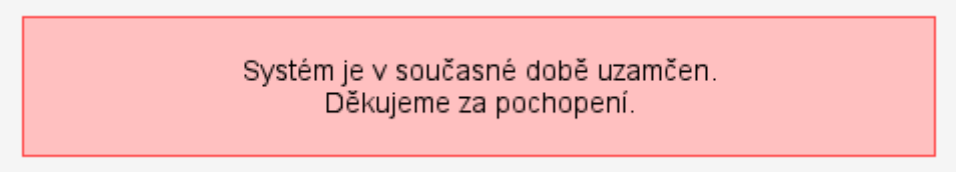
Ve výchozí instalaci aplikace je přístup do databáze nastaven jako uživatel root s prázdným heslem. Toto nastavení je zvoleno kvůli výchozímu nastavení jednotlivých závislých služeb a je čistě k demonstrativní ukázce, že aplikace funguje a není vhodné aplikaci pod tímto přihlášením do databáze dále provozovat. Předpokládám, že správce, který se rozhodne aplikaci nasadit na vlastním serveru má znalost základních nastavení a bude tak chtít aplikaci zařadit do již existující komunity jiných aplikací, a proto tuto konfiguraci nechávám zcela otevřenou.

V případě nasazení na hostingu není potřeba řešit toto zabezpečení. Pouze se vyplní správné přihlašovací údaje k databázi, které poskytuje hostingová společnost, do souboru **db-login.php** v adresáři **app/cfg/**.

7.4.5 Aplikační zámky

Pro potřeby řídit přístupnost aplikace jsem zavedl tzv. aplikační zámky. Umožňují zablokovat konkrétní část aplikace a zobrazit informací, která se do nich napíše. Konfigurovat lze tyto čtyři zámky.

- **zámek na celou aplikaci**
  – tento zámek zablokuje přístup do celé aplikace. Vhodné pokud neprobíhají individuální třídní schůzky.
- **zámek na administraci aplikace**
  – tento zámek zablokuje pouze administraci aplikace. Vhodné pokud chceme mít jistotu, že se do administrace nikdo nemůže dostat.
- **zámek na veřejnou část aplikace**
  – tento zámek naopak zablokuje celou veřejnou část aplikace a administrace bude přístupná. Vhodné pro přípravu nových termínů.
- **zámek na nové rezervace**
  – tento zámek zablokuje pouze podání nové přihlášky. Vhodné pokud chceme ukázat volné termíny, ale oficiální spuštění registrací chceme ještě pozdržet.



Obrázek 7.2: Ukázka aplikačního zámku

7.4.6 Drobná ale důležitá zabezpečení

Protože každý útok začíná důkladnou analýzou, je vhodné tuto analýzu útočníkovi ztížit. K tomu poslouží následující direktivy, které nám PHP nabízí.

- **header('Server: ');**
  header('X-Powered-By: ');
  Pomocí těchto dvou direktiv skryjeme informaci o tom, na jakém webovém serveru aplikace běží. Útočník tak nemá tušení ani o verzi webového serveru a bez této informace je těžké zvolit vhodnou strategii třeba ze znalosti, na jaké chyby trpí použitý webový server.

- **header('X-Frame-Options: deny');**
  Direktiva slouží k ochraně proti tzv. click-jackingu, tedy nechtěnému klikání například pomocí rámů. Při použití není možné aplikaci zobrazit v rámu jiné stránky.

- **header('X-XSS-Protection: 1; mode=block');**
  Tato direktiva není příliš podporována, ale slouží k zapnutí XSS filtru u některých prohlížečů. Největším přínosem asi je, že ji podporuje právě prohlížeč, který používá velké procento uživatelů, Internet Explorer „již“ od osmé verze.

- **ini_set('session.use_cookies', 1);**
  ini_set('session.use_only_cookies', 1);
  Povolí a následně vynutí, aby se session ID ukládalo jen do cookie prohlížeče a nemohlo tak dojít „k opsání“ či nechtěnému poslání URL adresy s session ID.

- **ini_set('session.cookie_httponly', 1);**
  Cookie aplikace budou přístupné pouze pomocí HTTP. Bohužel tuto direktivu nepodporuje mnoho prohlížečů.

- **ini_set('session.use_trans_sid', 0);**
  Zakáže použití session ID v URL i přesto, že klient neakceptuje cookie s session ID.

- **ini_set('display_errors', 'Off');**
  Když vznikne v aplikaci chyba, tak ji nezobrazíme, protože podle ní by mohl útočník jednat dále.

Zabezpečení session jsem použil i přesto, že aplikace v současné době session nepoužívá.

7.5 Automatické testování zabezpečení

K automatickému testování zabezpečení aplikace jsem využil opensource projekt Wapiti. Jedná se o skript v jazyce Python, který otestuje zadanou URL adresu, včetně odkazovaných adres, zadanou sadou známých útoků.

```
root@pentest$ wapiti http://local.bp/
Wapiti-2.2.1 (wapiti.sourceforge.net)
.
 Notice
========
This scan has been saved in the file /root/scans/local.bp.xml
You can use it to perform attacks without scanning again the web
site with the "-k" parameter
[*] Loading modules :
        mod_crlf, mod_exec, mod_file, mod_sql, mod_xss,
        mod_backup, mod_htaccess, mod_blindsql, mod_permanentxss,
        mod_nikto

[+] Launching module crlf

[+] Launching module exec

[+] Launching module file

[+] Launching module sql

[+] Launching module xss

[+] Launching module blindsql

[+] Launching module permanentxss

Report
------
A report has been generated in the file generated_report
Open generated_report/index.html with a browser to see this report
```

Během vteřin tak zjistíme, kde přesně a proč je slabé místo naší aplikace. Testování určitě není dokonalé, ale je například vhodné k rychlému ověření, že nově přidané pole ve formuláři je správně ošetřené proti běžným chybám.

Kapitola 8

Testování a vyhodnocení

8.1 Testeři

První testování spočívalo v kontrole, zda aplikace funguje dle předpokladů, splňuje zadané požadavky a hlavně zda odolá běžným pokusům v aplikaci způsobit chybu.

8.1.1 Průběh testování

Testování probíhalo lokálně mnou se znalostí zdrojového kódu a také s pomocí mých kolegů, kteří mi aplikaci připomínkovali na testovací subdoméně hostingu.

Během testování zda aplikace splňuje předepsané funkcionality z pohledu rodiče a vedení školy, nás nejvíc zajímalo přihlašování rodičů na termín a práci s termíny v administraci aplikace.

Při testování zabezpečení aplikace jsme se snažili v aplikaci způsobit takovou chybu, která by mohla ukázat bezpečnostní mezeru celé aplikace. Tento testovací scénář se zaměřoval hlavně na veřejně přístupnou část – přihlašování rodičů na termín. Stěžejními body byla kontrola SQL injection a přístupnost datových souborů aplikace.

8.1.2 Vyhodnocení

Během testování se podařilo odhalit drobné chyby, které vznikly jako důsledek urychleného vývoje. Také mi bylo doporučeno mírně přepracovat uživatelské rozhraní.

Při testování zabezpečení se ukázalo nedostatečné zabezpečení adresářů aplikace, na které se podařilo přistoupit pomocí URL v prohlížeči. Jako řešení jsem tak zvolil zabezpečit adresáře proti přístupu pomocí souboru *.htaccess*. Toto řešení problém vyřešilo.

8.2 Zkušební nasazení

Zkušební nasazení probíhalo na vlastním serveru základní školy. Hlavním cílem bylo, aby se vedení školy vyjádřilo k funkčnosti a ovládání aplikace. Zároveň jsem měl tak možnost vyzkoušet si instalaci, správu a po mírných změnách i vytvoření DEB balíčku s novou verzí.

8.2.1 Průběh testování

Ve zkušebním nasazení se testovací scénář až na výjimky blížil reálnému provozu aplikace. Testování probíhalo celý týden a postupně tak byly vyzkoušeny všechny funkce. Během testování se totiž připravovala data se kterými se pak chystalo ostré spuštění aplikace.

8.2.2 Vyhodnocení

V průběhu testování se postupně dělaly drobné změny ve slovosledu a názvech odkazů či tlačítek. K dosavadní funkčnosti aplikace nebylo nic vytknuto, ale vznikly další požadavky jako je použití HTTPS a aplikační zámky.

Poté, co jsem tyto funkce doplnil, byla aplikace shledána jako vhodná k ostrému nasazení.

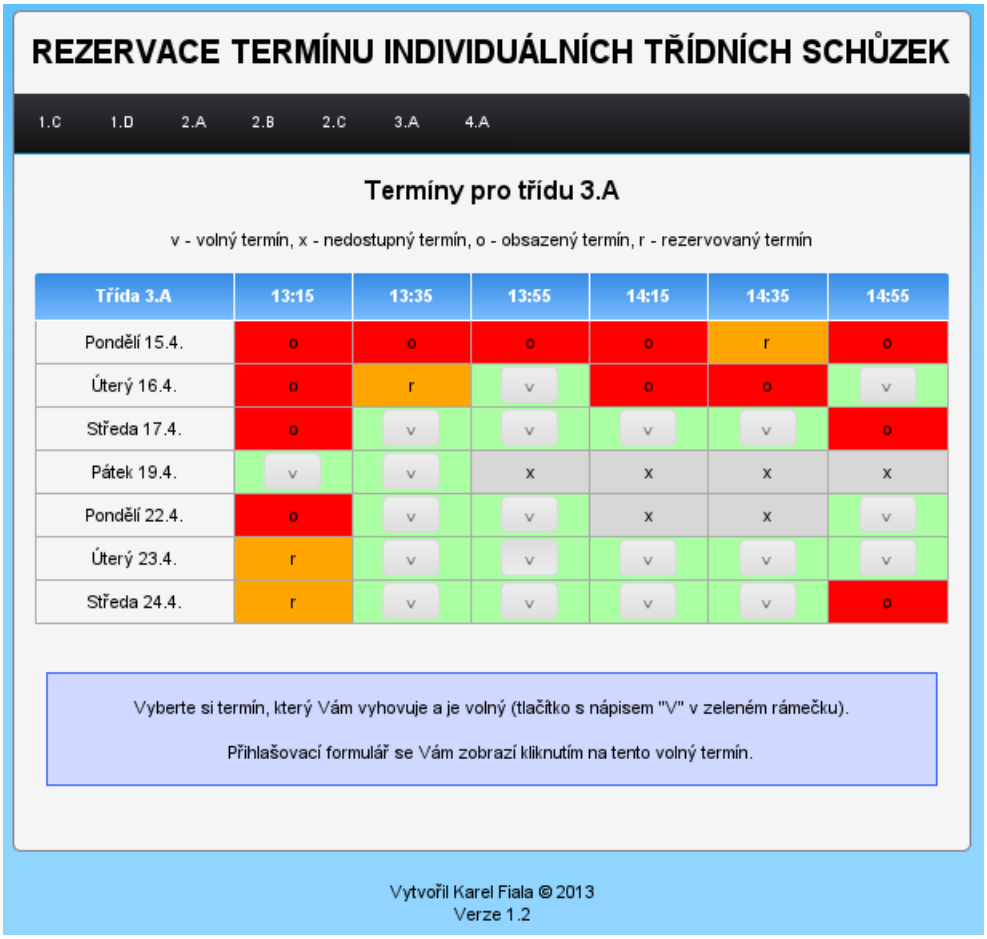
8.3 Ostré nasazení

První ostré nasazení aplikace nakonec probíhalo na hostingu, který si škola platí. Importovala se data vytvořené během zkušebního nasazení a aplikace se postupně odemykala pro přístup veřejnosti k přihlašování.

8.3.1 Vyhodnocení

Aplikace splnila svůj účel a během nasazení nevznikl žádný problém. Přesto však nasazení přineslo mnoho informací a tipů k dalšímu rozvoji aplikace. Mezi ty nejcennější informace určitě patří následující.

- Rodiče by rádi dostali upozornění včetně informací o svém termínu po uskutečnění úspěšné registrace.
- Také by rodiče rádi dostali upomínku, že následující den mají dohodnutý termín na konkrétní hodinu.
- Zároveň se ukázalo, že celá aplikace by zasloužila ještě pár nápovědných popisků navíc, aby se i méně zdatní rodiče lépe zorientovali.
  – Tento požadavek jsem tedy rovnou zapracoval a pokusil se každou stránku doplnit krátkým doprovodným textem.



Obrázek 8.1: Ukázka výsledné aplikace

- Vedení školy by chtělo každý den dostat jeden souhrnný email s informací kolik je nových přihlášek a jaké jsou stavy termínů ve třídách.
- Vedení školy také požaduje, aby aplikace umožňovala kliknutím odeslat emaily jednotlivým učitelům s tabulkou přihlášených žáků.
- Jednomu rodiči se stalo, že po kliknutí na tlačítko **Odeslat přihlášku** se „nic nedělo“ – vznikla totiž prodleva s odpovědí serveru a pak při druhém kliknutí dostal informaci, že je již termín obsazen.

Kapitola 9

Vývoj do budoucna

Vedení školy si pochvaluje ulehčení práce a přehlednost celého průběhu individuálních třídních schůzek. Rodiče jsou rádi, že se jim dostane individuálnějšího přístupu při řešení dalšího rozvoje jejich dítěte a zároveň nemusí vynaložit téměř žádné úsilí navíc. Z těchto důvodů jsem byl požádán o další vývoj aplikace.

9.1 Možné změny

Mezi první změny, které se chystám udělat patří emailové notifikace pro rodiče, pravidelné statistické zprávy pro vedení školy a pro správce bych chtěl realizovat notifikace na případné chyby. Toto rozšíření by mohlo zajistit větší komfort při použití aplikace.

Dalším vývojem, který je ale závislý na informační gramotnosti jednotlivých učitelů, by mohlo být zavedení uživatelských účtů a přístupových práv. Takové rozšíření by opět mohlo ulehčit práci vedení školy s organizací a poskytlo by větší kontrolu s organizací individuálních třídních schůzek jednotlivým třídním učitelům. Bohužel zde také narážíme na skutečnost, že někteří učitelé příliš nefandí informačním technologiím a pokud se individuální třídní schůzky nestanou určitým standardem vyžadovaným rodiči tak hrozí, že by nedostatečná snaha učitelů vedla spíše k návratu ke klasickým třídním schůzkám.

9.1.1 Z třídních schůzek budou konzultace

Nedávno nás také na poradě napadla myšlenka, že bych aplikaci přepracoval tak, aby sloužila k evidenci a organizaci konzultací jednotlivých učitelů. Funkčnost by byla zachována, jen místo tříd by byli jednotliví učitelé, kteří by si mohli sami vypisovat své vlastní konzultační termíny po celý školní rok.

Tato myšlenka by se dala dále rozvíjet například o informaci, ve které místnosti bude termín probíhat. Během třídních schůzek by bylo vypsáno hodně termínů, které by tak posloužily, jako jsou nyní termíny pro ITS s tím rozdí-

lem, že by rodičům umožňovali přihlásit se i na konzultační termín jiného než třídního učitele.

9.2 Mobilní verze aplikace

Tablety různých velikostí a mobilní telefony nás obklopují stále častěji. Současná aplikace ale není optimalizovaná pro tato zařízení a pokusí se zobrazit zcela identicky jako je tomu na klasickém počítači. Takové chování není vhodné a je třeba zvážit optimalizaci aplikace pro menší rozlišení a menší obrazovky než mají klasické počítače.

Kapitola 10

Současný stav

10.1 Aplikace pomáhá s organizací zápisu do prvních tříd

Každoroční zápis do prvních tříd bývá symbolem mírného zmatku. Budoucí prvňáčci jsou často nesmělí a rodiče s nimi přichází ve vlnách. Dochází tak k situacím, kdy jsou některá stanoviště zápisu prázdná a naopak na jiných se tvoří fronta. Což samozřejmě nikomu na klidu nepřidá.

Vedení školy tedy rozhodlo, že se pokusí tuto situaci řešit vypsáním termínů, na které mohou rodiče s prňáčky přijít a dostanou tak přednost před nepřihlášenými. Nejdříve byly vypsány termíny pro 50% očekávaných rodičů. Avšak vzhledem ke skutečnosti, že všechny termíny byly zabranné během pár dní, jsme dospěli k závěru, že bude vhodné vypsat další termíny. Celkem jsme tedy vypsanými termíny pokryli 80% očekávaných rodičů a zápis tak probíhal po šesti dětech současně. Zbývajících 20% rodičů vyplnilo případné odchylky vzniklé z časové rezervy každého termínu.

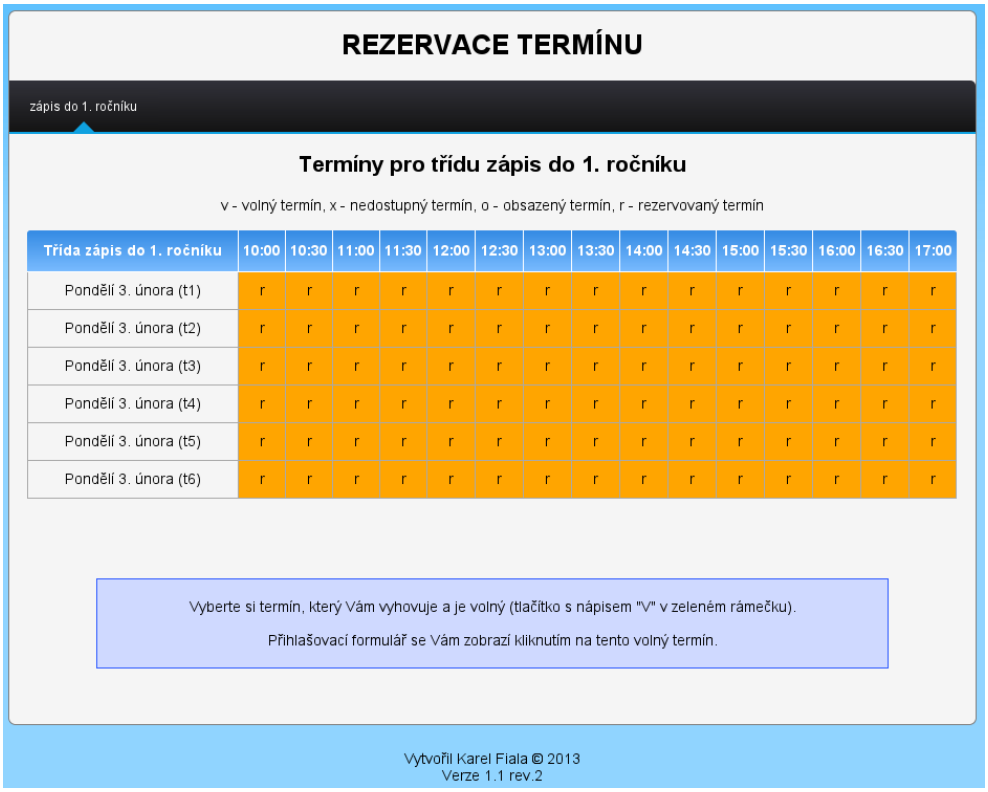
Celý zápis tak proběhl velmi organizovaně a klidně.

10.2 Zájem projevila další škola

Vzhledem k pozitivním ohlasům od rodičů i vedení školy na přínos aplikace, se nám ozvala spřátelená škola, která by měla zájem o využití této aplikace pro své vlastní záležitosti. V současné době se vše nachází ve fázi jednání a je na vedeních jednotlivých škol, zda se případně dohodnou.

10.3 Logopedická ordinace

Aplikace se také zalíbila jedné mamince, která vlastní logopedickou ordinaci a ráda by umožnila svým klientům přihlašování na vypsané termíny logopedických cvičení. Bohužel zde jsou požadavky na aplikaci poměrně diametrálně



Obrázek 10.1: Ukázka použití aplikace pro zápis

odlišné. Kromě odlišné funkčnosti by se také slušelo rovnou uvažovat o použití novějších technologií jako je HTML5, CSS3 a určitě také mobilní verze aplikace.

Vzhledem k těmto skutečnostem je zatím vše ve fázi vyjednávání.

Závěr

Ve své práci jsem analyzoval potřeby pro evidenci a organizaci individuálních třídních schůzek, definoval jsem potřebné konfigurace aplikace, navrhl strukturu aplikace a aplikaci poté implementoval. Dále jsem aplikaci zabezpečil, rozšířil její možnosti konfigurace a připravil ji pro další vývoj. Celou aplikaci jsem řádně otestoval, nasadil v reálném prostředí a nastínil její další vývoj.

Velice mě těší skutečnost, že vývoj aplikace neskončí současně s touto prací. Budu v něm pokračovat dále tak, jak jsem již uvedl v 9. kapitole.

Literatura

[1] Kolektiv autorů. New features in ASP.NET MVC 4. Dostupné z WWW: <http://dreamztech.com/blog/new-features-in-asp-net-mvc-4/>, 2013.

[2] Kolektiv autorů. Cross-site scripting. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Cross-site_scripting>, 2013.

[3] Kolektiv autorů. SQL injection. Dostupné z WWW: <http://cs.wikipedia.org/wiki/SQL_injection>, 2013.

[4] Kolektiv autorů. Basic access authentication. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Basic_access_authentication>, 2013.

Příloha **A**

Seznam použitých zkratek

ITS Individuální třídní schůzky

DEB DEB balíček – je instalační soubor pro operační systémy založené na Distribuci Debian

HTTP Hypertext Transfer Protocol – internetový protokol pro výměnu hypertextových dokumentů

HTTPS Hypertext Transfer Protocol Secure – zabezpečený HTTP protokol, který přenáší šifrovaná data pomocí SSL/TLS

XSS Cross-site scripting – metoda narušení webových stránek pomocí neošetřených vstupů

SQL Structured Query Language – strukturovaný dotazovací jazyk pro komunikaci s databází

MVC Model-View-Controller – třívrstvá softwarová architektura aplikace

PL/SQL Procedural Language/Structured Query Language – programovatelná nadstavba databázových systémů

PHP Personal Home Page – skriptovací programovací jazyk

HTML HyperText Markup Language – značkovací jazyk pro hypertextové dokumenty

CSS Cascading Style Sheets – jazyk pro popis grafického zobrazení internetových stránek

JS JavaScript – skriptovací jazyk pro vykonávání akcí na straně klienta

MD5 Message-Digest Algorithm – kryptografická hašovací funkce

URL Uniform Resource Locator – slouží k určení umístění zdrojů na internetu

Příloha **B**

Seznam použitého softwaru

shell – příkazový řádek pro *nixové operační systémy, např. bash

MySQL – databázový systém

PostgreSQL – databázový systém

Ubuntu – jedna z Linuxových distribucí

Debian – jedna z Linuxových distribucí

MySQL WorkBench – program pro návrh grafický databáze pro databázový systém MySQL

NetBeans – vývojový nástroj pro jazyky Java, PHP, ...

Apache2 – webový server s podporou PHP, MySQL, PostgreSQL

Wapiti – skript pro penetrační testy

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `chroot` ..... rozbalený DEB balíček
  - `docs` ..... dokumentace zdrojového kódu
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF