VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

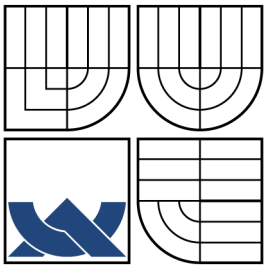
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MODERNÍ E-LEARNING SYSTÉM NA PLATFORMĚ JAVA

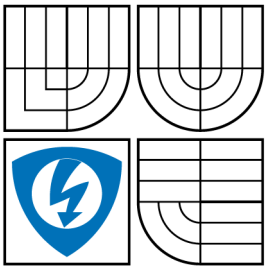
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. TOMÁŠ HÁK
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MODERNÍ E-LEARNING SYSTÉM NA PLATFORMĚ JAVA

MODERN E-LEARNING SYSTEM ON JAVA PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. TOMÁŠ HÁK
AUTHOR

VEDOUCÍ PRÁCE — ING. VÁCLAV PFEIFER
SUPERVISOR

BRNO 2008

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMLOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMLOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Tato diplomová práce se zabývá e-learningem – elektronickou formou výuky. Rozebírá detailněji klady i zápory této nové formy přístupu ke vzdělávání. Zároveň jsou popsány možnosti některých vybraných e-learningových systémů a jejich základní struktury. V druhé části této práce jsou uvedeny základní principy a možnosti programovacího jazyka Java. V tomto jazyce je následně navrhnut vlastní koncept systému, který by umožňoval základní správu a ovládání e-learningu. Je popsán průběh komunikace mezi serverem a klientem, způsob odesílání SQL dotazu a přenosu výsledku po síti zpět ke klientovi. V poslední části je rozebrána výsledná aplikace jako celek, je vysvětleno základní ovládání, možnosti administrace a způsob zadávání dat. Vlastní aplikace je testována v rámci lokálního testovacího serveru.

KLÍČOVÁ SLOVA

e-learning,jazyk Java,3 vrstvý model,internetová výuka,dvojitě vázaný seznam, abstraktní datový typ, serializace, databáze SQL

ABSTRACT

This dissertation is studying e-learning – electronic form of education. At once it is analysing in details the advantages and disadvantages of this new form of education approach and is describing possibilities of some chosen e-learning systems and their basic structures. In the second part of this dissertation are said basic principles and possibilities of programming language Java in which language is designed it's own concept of system, that should make possible for users to manage and control the e-learning. In this part is described progress of communication between server and client, method of sending off the SQL (query) and receiving result too. In last part is viewed resulting application as a whole, is explained basic controlling, possibilities of administration and methods of entering data. Application itself is tested on local testing server.

KEYWORDS

e-learning,language Java, 3 layers model, internet education, double linked list,abstract data type, serialization, database SQL

HÁK, T. *Moderní e-learning systém na platformě JAVA*. Brno : Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. 68 s. + 1 CD. Diplomová práce. Vedoucí práce Ing. Václav Pfeifer.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Moderní e-learning systém na platformě JAVA“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Chtěl bych poděkovat Ing. Václavu Pfeiferovi, vedoucímu mé diplomové práce, za odborné vedení, cenné rady a připomínky, které mi při přípravě a vypracování diplomové práce ochotně poskytoval.

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Informační a komunikační technologie přináší nové široké možnosti pro využití ve spoustě oborů a nejinak je tomu i ve vzdělávání. Především v distanční formě vzdělávání mají informační a komunikační technologie své zásadní využití při tvorbě a používání multimediálních výukových opor, při komunikaci mezi studenty a lektory, při organizování a administraci distančního studia atd.. Tyto technologie se však mohou uplatnit i při kombinované výuce či jako doplněk pro prezenční formu studia.

Tato diplomová práce seznamuje čtenáře s pojmem e-learningu, rozebírá detailněji klady i zápory této formy přístupu ke vzdělávání. Zároveň jsou popsány možnosti některých vybraných současných e-learningových systémů a jejich základní struktury.

V druhé části této práce jsou uvedeny základní principy a možnosti programovacího jazyka Java. V tomto jazyce je následně navrhnut a zrealizován vlastní koncept e-learningového systému, který je testován v rámci lokálního serveru.

1 ZÁKLADNÍ POJMY

1.1 E-learning

1.1.1 Obecný popis

Pojem e-learning vyjadřuje výuku prostřednictvím výpočetní techniky. Předmětem e-learningu nemusí být samotná výuka, ale také předávání informací nebo ukázka práce s nějakým (např. softwarovým) systémem.

E-learning podle [6] a [1] představuje informační systém zajišťující řízení, monitoring, transfer a vyhodnocování informací poskytovaných studentovi prostřednictvím počítačových on-line nebo off-line kurzů. Kurzy jsou zpravidla sestavovány tak, aby vyžadovaly určitou interakci studenta za účelem provokace vedoucí k vlastní účasti na studiu. Takový přístup má pozitivní vliv na paměť studenta. Vzdělávací materiály v textové podobě jsou doplněny o různorodé multimediální prvky (simulace, animace, video apod.), aby výuku ještě více zpestřily.

Smyslem e-learningu je umožnit další vzdělávání pracovně vytíženým zájemcům, kteří mají zájem o rozšiřování odborných znalostí za účelem zvyšování své ceny na trhu práce. Vzhledem k neustálým dynamickým změnám ve společnosti dochází k růstu nároků firem i neziskových organizací na odbornost svých zaměstnanců. Jednou z možností, jak se podílet na zvyšování vlastního potenciálu z pohodlí vlastního domova, je e-learning.

Studium je značně časově flexibilní, tzn. že doba studia závisí pouze na studentovi. Pro zastánce tradičních metod výuky umožňuje e-learning také spuštění videa nebo zvukových záznamů z konkrétní prezentace lektorem, čímž je nahrazena nejčastější forma tzv. face-to-face studia. Dotazy ke studiu může student pokládat svému lektorovi prostřednictvím e-mailu, fóra nebo on-line chatů. Chatování nemusí probíhat striktně pouze s lektorem, ale také s ostatními studenty kurzů, což umožňuje transfer osobních znalostí a zkušeností nebo společné řešení vlastních nebo lektorem předložených problémů.

Výuka je nejefektivnější tehdy, pokud její předmět odpovídá zájmům studentů, jenž jsou zároveň i jejich potřebami. Elektronická komunikace s lektorem umožňuje méně protřelým studentům klást odborné otázky lektorovi nepřímo. Studium prostřednictvím e-learningu umožňuje studentovi si v klidu nastudovat odbornou problematiku a položit smysluplné otázky. Studentská komunita mu poskytuje vědomí sounáležitosti a členství v ní sdílení zkušeností, možnost spolupráce i vzájemné podpory.

1.1.2 Výhody e-learningu

Přínosy by se daly rozdělit podle [1] do dvou perspektiv, a to z hlediska společnosti a z hlediska uživatelů e-learningu, tedy studentů. Toto hledisko je vždy uváděno v závorce.

- Vyšší efektivnost výuky (student) - nejvíce citovaným přínosem e-learningu je jeho flexibilita. Ta je umožněna díky tomu, že veškeré informace jsou uspořádány do malých přehledných modulů, ze kterých se skládají jednotlivé kurzy. Tím mohou vznikat kurzy přesně podle potřeb a požadavků uživatelů. Další výhodou je pak to, že uživatelé mohou přistupovat k těmto zdrojům odkudkoli (s přihlédnutím na potřebu komunikačního média) a kdykoli. Sami si tak můžou plánovat čas strávený studiem.

- Dostupnost kdykoliv (student) [1] - e-learningový systém (či portál) je přístupný na rozdíl od tradičních vzdělávacích kurzů kdykoli a poskytuje informace v čase, který není určován vzdělávacími centry nebo školicím plánem podniku a instituce, nýbrž samotným uživatelem. Informace jsou tak získány ve správný čas a přesně podle potřeb uživatele, který si může vybrat, na kterou část probíráné látky se chce zrovna zaměřit. E-learning tak umožňuje uživatelům studovat jak doma, tak v práci nebo ve speciálním školicím centru.

- Individuální přístup k uživateli (student) - uživatel přestává být v e-learningu pasivním účastníkem, jako je tomu po většinu času při klasickém kurzu. Je naopak interaktivním systémem donucen informace vyhledávat a nacházet v nich potřebné znalosti. Volba probírané látky již nezáleží na možnostech lektora, ale na požadavcích studentů. Při tradičním školení získají všichni účastníci shodné materiály ve stejné obsahové formě a zaměřené na řešení stejných problémů. Moderní e-learningový systém umožňuje vytvořit pro každého uživatele jeho tzv. profil, nebo-li jistý soubor informací o samotném uživateli, jeho studijním stylu, oblíbené formě materiálů apod. Dále je možné ukládat do těchto profilů také informace o předešlých zkušenostech a absolvovaných kurzech každého uživatele. Systém pak případně může „za běhu“ definovat směr vzdělávání, kterým se každý jednotlivý student ubírá a navrhnout mu další postup v podobě navazujících znalostních modulů.

- Menší náklady na vzdělávání (společnost) - dalším velmi diskutovaným přínosem e-learningu je jeho efektivnost po finanční stránce. Mnoho e-learningových poskytovatelů právě na cenové výhodě pro spotřebitele staví svoji marketingovou strategii. Jejich názor je takový, že u e-learningového řešení vzdělávání

[1] v literatuře označováno jako „just-in-time“

odpadají náklady na tisk a distribuci materiálů a především na dopravu studentů (popřípadě i jejich ubytování a stravování) na místo konání kurzu či školení. Dále je zde i eliminován zisk z ušlé příležitosti, neboli čas, který je bez náhrady stráven na tradičním školení. Toto platí v případě pracujících uživatelů. Přestože se většina odborníků shoduje, že jisté finanční výhody e-learningové řešení určitě přináší, ne vždy se jejich názory zcela překrývají. Například Evropská komise uvádí číslo 32,4 procent jako procentuální podíl nákladů, které lze ušetřit oproti tradičnímu vzdělávání. Na druhou stranu metody kalkulace nákladů se významně liší u Spojených států amerických, kde se tento ukazatel pohybuje okolo hodnoty 75 procent. Hlavní rozdíl ve výsledcích pravděpodobně spočívá v rozdílné velikosti evropského a amerického trhu a v rozdílné připravenosti firem na zavedení e-learningového řešení. Pravdou ovšem je, že počáteční náklady na zavedení e-learningu jsou vysoké, s počtem účastníků však klesají, a od určitého množství uživatelů jsou pak náklady na každého dalšího studenta-uživatele téměř nulové.

- Modularita (student i společnost) - znalosti jsou studentům poskytovány v tzv. modulech, což je činí jednak rychleji vstřebatelnými, ale především daleko přehlednějšími. Moduly lze charakterizovat jako malé části logicky spjatého obsahu na určité téma. V jednotlivých kurzech lze takto charakterizovat například jednotlivé podkapitoly. Další výhoda modulů je jejich snadná aktualizovatelnost. Pokud totiž nějaký z modulů zastará, například vlivem příchodu nové verze aplikace, či novými objevy v daném oboru, není žádný problém ho odstranit a následně nahradit novým modulem, který je doplněn i novými funkcemi.

- Větší aktuálnost informací (student i společnost) - síťové technologie umožňují neustálou synchronizaci dostupných zdrojů a prezentaci nejaktuálnějších materiálů. Změny v obsahu e-kurzů (ať už jsou dané třeba zastaráním nějaké informace, změnou legislativních opatření, změnou v požadavcích uživatele nebo změnou související se samotnými produkty) lze provádět díky propojitelnosti a pravidelné akutalizaci serverů ihned, z jednoho místa a velmi snadno. Obsah tak nezůstává statický, jako je tomu v případě tištěných materiálů, jejichž aktualizace je poměrně složitá a finančně nákladná.

- Rychlejší vstřebávání informací (student) - dalším přínosem je rychlejší vstřebávání informací uživatelem, než při tradičním školení. Tento názor vychází z předpokladu, že uživatel dostává pouze informace, které on sám bezprostředně vyžaduje, a navíc v interaktivní formě, díky které postupuje ve vzdělávání rychlejším tempem na rozdíl od klasického skupinového školení. Výuka je tak

uživatelům ušita přímo na míru a neztrácejí čas informacemi, které pro ně nemají význam.

- Lépe zapamatovatelná forma informací (student) - moduly, tedy určité balíky přesně požadovaných informací, jsou předány studentovi ve formě, která je bezesporu daleko lépe vybavitelná v pozdější aplikaci znalostí při řešení reálných problémů. Informace jsou předávány po malých částech a znalosti jsou velice koncentrované, což znamená, že tedy učební materiály nemusí obsahovat žádné zbytečné texty. U synchronních kurzů navíc existuje stálá možnost vznést dotaz na tutora [2], který je odborníkem v daném oboru, při jakékoliv nejasnosti či nepochopení látky. Další výhodou je možnost konzultace dané problematiky s ostatními studenty, např. prostřednictvím fóra k danému kurzu.

- Větší možnosti testování znalostí (student) - studenti mají v e-learningových systémech možnost bezpečného otestování svých znalostí po absolvování jednotlivých vzdělávacích kroků. Stejný úkol je možné řešit vícekrát bez jakékoliv obavy z chyby a díky anonymitě uživatelů mají jejich odpovědi vyšší vypovídací hodnotu. To dovoluje prosazení i stydlivějším, nejistým nebo velmi těžko se vyjadřujícím osobám, které by během tradičních kurzů aktivitou příliš neoplývaly.

  Dále je zde bezesporu eliminována možnost ovlivnit výsledky různými negativními vlivy souvisejícími s osobním kontaktem, protože testy jsou opraveny předem vytvořenou softwarovou aplikací a lidský faktor je zcela odstraněn.

- Shodný obsah pro všechny studenty (student i společnost) - při vzdělávání prostřednictvím e-learningového systému dostává každý uživatel stejné informace (pokud nebereme v potaz změny modulů díky aktuálnímu vývoji znalostí databáze v daném oboru). Toho není možné u tradičních kurzů dosáhnout. Mění se lektoři, používají se různé informační zdroje a materiály. Tento nesoulad v získaných znalostech může ve firmě způsobit nemalé problémy, jsou-li přijaté informace interpretovány několika způsoby. Největší problém může nastat u oborů znalostí, které by měly být standardizovány, například instrukce pro vojenské operace nebo právní zásahy. Problém může ovšem nastat i při docela jednoduchých kurzech, na kterých závisí další práce zaměstnance, například kurz na seznámení s vnitropodnikovou kulturou a pravidly.

- Vyšší míra interaktivity (student) - e-learning je interaktivním prostředím. Neomezuje se na poskytování informací pouze v textové podobě, ale přináší do

[2] Tutor je nejbližší spolupracovník studujícího, vyškolený pracovník pověřený vzdělávací institucí pro řízení studujících v určitém výukovém modulu.

výuky mnoho multimediálních prvků, které zvyšují dynamičnost celého kurzu. Hojně zaváděným a oblíbeným prostředkem interaktivity se stávají v poslední době simulace, ve kterých uživatel má možnost vyzkoušet reálné situace a navrhnout určitý druh řešení. Poté jsou mu vráceny výsledky v podobě další simulace následků jeho rozhodnutí. Tento prvek ovšem vyžaduje poměrně složité a nákladné řešení simulačních doplňků tak, aby mohly být vhodně implementovány do komplexního e-learningového systému.

- Snadná administrace (společnost) - e-learningový systém zahrnuje jako jednu z hlavních složek také administrativní stránku vzdělávacího procesu. Veškeré nutné administrativní práce byly a v mnoha případech stále jsou prováděny především v papírové podobě, která prochází jednotlivými odděleními firmy, kde jsou její dílčí části zpracovány. Tento způsob však značně zvyšuje nároky na lidskou práci s administrací spojenou. Kvalitní e-learningový systém, který zahrnuje veškeré pomocné funkce pro registraci uživatelů, plateb, monitorování vzdělávacího procesu, testování uživatelů a zpracování jejich výsledků, poskytuje značné finanční úspory. Ty jsou dosaženy především snížením režijních nákladů na vyřizování formalit a snížením mzdových nákladů na administrativní práce. (Platí zejména tam, kde je pro oblast vzdělávání používán HRIS [3], neboť tak lze zajistit integraci nebo propojení dat o vzdělávacích akcích. Tím se značně zjednoduší vzdělávací procesy a sníží náročnost administrativy.)

  Jinou výhodou komplexně administrativně vytvořeného systému je možnost provádění certifikace uživatelů, kterou využívá čím dál více firem. Problém ovšem spočívá v nejednotnosti požadavků na výsledky testování znalostí, nejednotné jsou v různých firmách i testové otázky. Každá firma poskytující certifikaci se snaží o prosazení svého certifikátu a zvýšení jeho důležitosti v tržních podmínkách tak, aby uživatelům jméno certifikační firmy přinášelo co největší konkurenční výhodu na trhu pracovních sil.

- Zvyšování obecných znalostí z informačních technologií (student i společnost) - e-learningový trend je podporován a zároveň podporuje mnoho informačních a komunikačních technologií, které je nutné k efektivnímu použití systému umět ovládat. Díky snadným navigacím a hlavně potřebě vzdělání v informačních technologiích pro okamžité využití lze říci, že používání e-learningu pomáhá zvyšovat znalosti z tohoto oboru daleko rychleji než tradiční vzdělávání.

[3] HRIS - Human Resource Information System, v překladu: informační systém pro řízení lidských zdrojů

1.1.3 Nevýhody e-learningu

Jako nic na světě, ani e-learning není dokonalý. Existuje i několik potencionálních bariér, které částečně brání efektivnímu využití tohoto relativně nového vzdělávacího trendu. V této kapitole je uveden jejich výčet. Každý, kdo o e-learningu uvažuje (ať už jako firma či jako jednotlivec), by je měl vzít v potaz a určitě o nich uvažovat.

- Nekompatibilita komponent - na toto téma se vede mnoho debat. Problém, který se řeší, je jak dosáhnout úplné kompatibility kurzů s různými systémy. Kvůli nedodržování standardů a nejednotnosti LMS [4] i postupů při tvorbě kurzů totiž dochází k tomu, že ne každý kurz je použitelný ve všech systémech. Zatím chybí jasná platforma, která by byla všeobecně uznávaná a používána všemi – něco jako formát PDF pro dokumenty. Vytvoření standardů by dovolilo popsat obsah e-learningových kurzů podle stejných charakteristik, takže by s ním pak mohly pracovat všechny existující e-learningové platformy. V současné době probíhá mnoho pokusů o vytvoření standardů ve Spojených státech amerických (AICC – letecký průmysl, IEEE – inženýrství, ADL – obranné systémy, IMS – univerzity) i v Evropě (ARIADNE – vzdělávání), ovšem zatím nebyly žádné všeobecně přijaté standardy vytvořeny. Jako možné řešení se nabízí používání formátu XML [5], pomocí něhož by mohl být uložen vlastní obsah kurzů a všechny důležité údaje. Formát XML je otevřený a lze jej proto implementovat na všech platformách.

- Závislost na technologiích - uživatelé e-learningu potřebují mít zajištěn trvalý přístup k určitému hardwaru i softwaru, aby mohli plně a efektivně využít funkce e-learningového systému. Tento požadavek se však může stát kritickým, protože ne každá firma takovým vybavením disponuje a jeho pořízení může být pro mnoho firem příliš nákladné. Pokud však není vybavení dostatečné, mohou vznikat problémy – následkem může být například limitovaný objem obsahu, nižší přenosová rychlost nebo nemožnost využití multimediálních doplňků. Některé názory dokonce hovoří o významných demotivujících aspektech e-learningu, které mají původ právě v technologických bariérách.

  Tato bariéra by snad měla být brzy odstraněna. Vývoj technologií je velmi rychlý a jejich ceny klesají. Internetové připojení má dnes již skoro každý a jedinci i firmy vynakládají na technické vybavení značné prostředky. Jinou

[4] Learning Management System je řídicí výukový systém (systém pro řízení výuky), tedy aplikace řešící administrativu a organizaci výuky v rámci e-learningu.

[5] XML (eXtensible Markup Language, v překladu rozšiřitelný značkovací jazyk) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat.

otázkou je, zda všichni lidé budou mít k těmto technologiím přístup. Většina odborníků z oblasti vzdělávání se obává, že e-learning ještě více prohloubí propastné rozdíly mezi kvalifikovanou a nekvalifikovanou pracovní silou, která ve většině případů možnost využívat tyto technologie nemá.

- Nevhodnost pro určité typy studentů - přestože možnosti e-learningu přizpůsobit se přesným potřebám, přáním i požadavkům uživatelů jsou velké, nedá se říci, že by byl e-learning opravdu pro každého.

  Většina e-learningových aplikací komunikuje s uživateli pomocí textových zpráv. Z tohoto důvodu je nutné, aby uživatelé dokázali převádět své myšlenky a řešení praktických úkolů do slovní podoby. Pokud to je pro někoho obtížné, jsou jeho možnosti omezené. E-learning se také nehodí pro sluchové a pohybové typy studujících. Pro ty je nejvhodnější klasické školení. Rovněž i starší lidé e-learning odmítají buď z principu (odpor ke všemu novému), nebo se již nechtějí či nejsou schopni naučit nové technologie ovládat. V neposlední řadě také mnoha lidem schází spolupráce s ostatními studujícími a při studiu se cítí osamoceni

- Princip dobrovolnosti - vzdělávání pomocí e-learingu je jedna z činností, která není ve většině případů na zaměstnancích či ostatních typech studentů přímo vyžadována. Záleží tedy zejména na motivaci a sebekázni každého jednotlivce, na tom, zda se opravdu chce něco naučit a obětovat svůj čas i úsilí. Největším problémem je určitě čas, který je nutný studiu věnovat. Převážná většina prosperujících firem dosáhla své pozice i tím, že produktivita jejich zaměstnanců dosáhla poměrně vysoké úrovně. Zaměstnanci často nepracují zákonných osm hodin denně, ale deset, dvanáct, někdy dokonce i více hodin podle aktuální potřeby. Potom je samozřejmě velmi obtížné najít dostatek volného času pro vzdělávání, a pokud se snad nějaký čas nalezne, často zapůsobí faktor únavy či demotivace z další činnosti související s prací.

- Vysoké počáteční náklady - přestože nákladová složka je citována především jako výhoda pro firmy, které zavádí e-learning do svých infrastruktur, často se mluví o nákladech na e-learning i v záporném slova smyslu. Variabilní náklady, tedy náklady na distribuci a řízení e-learningových modulů, jsou skutečně daleko menší ve srovnání s tradičním modelem vzdělávání. Ovšem do celkové kalkulace je nutné zavést i náklady, které souvisí s vývojem znalostních databází a s technologickou a multimediální podporou systému. U tradiční formy vzdělání jsou tyto náklady převážně zahrnuty v ceně školení, zvláště pokud školení probíhá u poskytovatele ve školicím centru.

- Závislost na lidské podpoře - většina firem používajících e-learning jako jednu z forem firemního školení používá pro své e-learningové aktivity interní i externí podporu. Interní podpora často zahrnuje kvalifikované zaměstnance z řad IT oddělení firmy, externí pak lektory, vývojáře obsahu e-learningových kurzů a konzultanty pro samotný e-learningový systém. Až po nějakém čase, kdy je systém v provozu a funguje bez komplikací, dochází k přesunu veškeré podpory na podporu on-line – veškeré dotazy a odpovědi jsou řešeny v podobě zaměstnance personálního oddělení řídícího vlastní školení a manažerů, kteří motivují pracovníky k využívání e-learningu pro zdokonalování svých znalostí.

1.2 Současné systémy elektronického vyučování

Tato kapitola se věnuje popisu používaných současných systémů elektronického vyučování. V této diplomové práci je vytvořena aplikace e-learningu orientována na akademické prostředí. Pro popis byly vybrány systémy používané na brněnských univerzitách, konkrétně na Masarykově univerzitě, Mendelově zemědělské a lesnické univerzitě a samozřejmě také Vysokém učení technickém.

1.2.1 E-learning na Masarykově univerzitě

Informační systém Masarykovy univerzity, jehož součástí je i vlastní e-learning, je samostatně vyvíjený univerzitní informační systém.

Podle článků v [13] se přístupy do tohoto systému pohybují zhruba kolem 20 000 uživatelů denně. Studenti zde provádějí potřebnou administrativu svého studia, obsluhují univerzitní elektronickou poštu a diskusní fóra, dozvídají se informace a dění na univerzitě a mají možnost pracovat s e-learningovými aplikacemi pro jednotlivé kurzy. Dále také systém umožňuje zájemcům podání přihlášky do studia, ukládání dokumentů na dokumentovém serveru nebo fulltextové vyhledávání.

Pro své pedagogy vytváří tento systém mimořádné podmínky, aby mohli rozvíjet jak komplexní e-learningové kurzy, tak i tzv. rapid e-learning. Rapid e-learning je elektronická podpora výuky, která se snaží umožnit rychle ke studentům dostat studijní objekty, dělit kurz spíše na více menších aktivit, sledovat, co na studenty funguje, rychle objekty modifikovat, vylepšovat, aktualizovat a reagovat na zpětnou vazbu. Kurzy jsou na základě interakce se studenty kontinuálně vylepšovány. Rapid e-learning vyžaduje analýzu toho, co se pro konkrétní kurz projeví jako užitečná elektronická podpora výuky, na co se zaměřit, čím se v poměru cena/výkon zabývat. Učitelé na Masarykově univerzitě mají k dispozici fakultní pracovníky pro

uživatelskou podporu (tzv. e-techniky), celouniverzitní pracoviště pro tvorbu multimediálních a interaktivních objektů a e-learningový zdroj obsahující metodické postupy, návody, zkušenosti učitelů s e-learningem, tipy a ukázky pro inspiraci.

Informační systém MU obsahuje mnoho e-learningových agend a širokou nabídku funkcí a služeb úměrnou množství fakult a rozdílnosti učitelů i oborů. Mezi základní e-learningové agendy patří: Studijní materiály, Odevzdávárny úloh, Diskusní fórum, Interaktivní osnovy, Odpovědníky, Skenování písemek, Hledání podobných souborů (služba pro odhalování plagiátů), prostor pro vlastní webovou prezentaci a svépomocné agendy pro studenty. Na obrázku 1.1 je vidět základní struktura e-learningové agendy. Každá z těchto částí může obsahovat libovolné množství dalších složek a souborů.

Obr. 1.1: Zobrazení struktury vzorového kurzu v e-learningu MU (výřez části obrazovky internetového prohlížeče

Systém je neustále vyvíjen již od roku 1999 a postupně nasazován i na jiných vysokých školách či fakultách, například na Fakultě humanitních studií Univerzity Karlovy. Jeho provoz je zajišťován asi 40 procesory. Na většině jeho počítačů je nainstalován operační systém Linux s webovým serverem Apache a databázovým serverem Oracle. Velká část aplikací byla vytvořena v programovacím jazyce Perl [6].

V posledních letech tomuto systému bylo uděleno několik prestižních ocenění. V roce 2005 to byla cena EUNIS Elite Award 2005, která se uděluje každoročně od roku 1999 za nejlepší implementaci informačního systému na některé z vysokých

[6]Perl - je interpretovaný programovací jazyk vytvořený Larry Wallem v roce 1987. Licencován je pod GNU General Public License, což znamená že, zdrojové kódy software mohou být svobodně upravovány a používány, šířeny však musí být opět pod GPL

škol v Evropě. Roku 2007 získala Univerzita za svůj elektronický archív závěrečných prací se systémem na odhalování plagiátů cenu Inforum 2007.

1.2.2 E-learning na Mendelově zemědělské a lesnické univerzitě

Univerzitní informační systém je programový systém (webový informační systém) vytvořený domácím vývojem na MZLU [7] v Brně za účelem automatizace její studijní a vědecko-výzkumné agendy.

Podle dokumentace na [14] pokrývá v současné době tento informační systém všechny hlavní činnosti univerzity a postupně automatizuje další provozní a administrativní agendy. Programový soubor je také postupně zaváděn na dalších univerzitách v České a Slovenské republice. Jedná se o Slovenskou technickou univerzitu v Bratislavě, Technickou univerzitu ve Zvolenu, Vysokou Školu Škoda Auto a.s., Vysokou školu ekonomickou v Praze [8].

Informační systém je rozvíjen postupně od roku 2000 a získal již v roce 2002 3. místo v soutěži EUNIS Elite Award of Excellence, v roce 2007 byl pak v téže soutěži doporučen jako High Recommended a získal 2. místo.

Do konce roku 2005 byl e-learning vyvíjen jako samostatný informační systém ELIS, později došlo k akvizici a nadále je rozvíjen jako modul e-learning v rámci UIS. Modul eAgenda je v současné době vyvíjen a postupně nasazován pouze na domácí univerzitě.

Informační systém je podle [6] členěn do následujících modulů, viz. obr. 1.2 . Všechny tyto moduly jsou přístupné z jednoho místa a je i všude zachováno stejné ovládání a vzhled, což velmi usnadňuje ovládání zvláště začínajícím uživatelům systému. Bohužel chybí navigační menu, které by umožňovalo alespoň zakladní možnosti ovládání.

Co se týká technického zázemí, tak je informační systém vyvíjen nad relačním databázovým systémem Oracle Database 10g s využitím možností jazyků SQL a PL/SQL [9]. Aplikační vrstva je tvořena pomocí původně vlastního programového jádra a soustavy modulů a aplikačních skriptů v programovacím jazyce Perl za využití dalších externích aplikací – zejména sázecího systému LaTeX. Na různých instalacích je provozován informační systém nad operačními systémy Linux (Red Hat Enterprise Linux, Fedora Core, CentOS) a Solaris. Nejmenší instalace informačního systému (pro testování) zahrnuje jeden počítač s procesorem Opteron, nejrozsáhlejší

[7] MZLU - zkratka pro Mendelovu zemědělskou a lesnickou univerzitu

[8] zde ještě implementace probíhá

[9] PL/SQL (Procedural Language/Structured Query Language) je procedurální nadstavba jazyka SQL od firmy Oracle založená na programovacím jazyku Ada.

Obr. 1.2: Struktura Univerzitního informačního systému MZLU

instalace (produkční instalace na MZLU v Brně) je provozována na 32 procesorech Opteron v šestnáctiuzlovém clusteru (aplikační strana) a 16 jádrech Sun Sparc ve dvouuzlovém clusteru (databázová strana).

1.2.3 E-learning na Vysokém učení technickém v Brně

E-learningový systém VUT je vytvářen na softwarovém balíčku Moodle a je implementován společně s celouniverzitním informačním systémem. Moodle patří do kategorie LMS, tj. řídicí výukový systém (systém pro řízení výuky), tedy aplikace řešící administrativu a organizaci výuky v rámci e-learningu.

LMS jsou podle [6] aplikace, které v sobě integrují zpravidla nejrůznější on-line nástroje pro komunikaci a řízení studia (nástěnka, diskusní fórum, chat, tabule, evidence atd.) a zároveň zpřístupňují studentům učební materiály či výukový obsah on-line nebo i off-line. LMS aplikací je řada - od těch jednoduchých přes nejrůznější LMS z akademické sféry až po rozsáhlé a složité komerční aplikace. Řada LMS je šířených i jako free nebo open source software.

V České a Slovenské republice jsou používané například tyto aplikace:

- Enterprise Knowledge Platform

- eDoceo
- Microsoft Class Server
- Moodle
- WebCT
- EDEN
- LMS UNIFOR

Aplikace Moodle, nad kterou celý e-learning VUT běží, je podle [12] softwarový balíček pro tvorbu výukových systémů a elektronických kurzů na internetu. Jedná se o neustále se vyvíjející projekt navržený na základě sociálně konstruktivistického přístupu k vzdělávání.

Moodle je poskytován zdarma jako Open Source software spadající pod obecnou veřejnou licenci GNU. To v zásadě znamená, že je chráněn autorskými právy, ale poskytuje přitom uživatelům značnou svobodu. Moodle můžete kopírovat, používat i upravovat, pokud souhlasíte s tím, že: budete tento zdroj poskytovat ostatním; nebudete měnit ani odstraňovat původní údaje o licencích a autorských právech, a uplatníte stejné licenční podmínky i u jakýchkoliv odvozených produktů.

Moodle je tedy volně šiřitelný software s otevřeným kódem. Běží na Unix, Linux, Windows, Mac OS X, Netware a na jakémkoliv dalším systému, který podporuje PHP. Data jsou ukládána v jediné databázi (největší podpora pro MySQL a PostgreSQL, nicméně lze použít i Oracle, Access, Interbase, ODBC atd.). Multiplatformnost v rámci operačního systému i systému databázového dělá z Moodle silný nástroj pro tvorbu e-learningu. Mezi další vlastnosti aplikace patří:

- Jednoduché, efektivní, široce kompatibilní, technicky nenáročné a intuitivní uživatelské rozhraní
- Snadná instalace na téměř všechny platformy, které podporují PHP. Vyžaduje pouze jednu databázi (a tu může sdílet)
- Velký důraz na zabezpečení: data ze všech formulářů jsou kontrolována, cookies jsou šifrovány atd.
- K stávajícím instalacím Moodlu lze přidávat doplňkové moduly činností
- Doplňkové moduly jazyků umožňují plnou lokalizaci do jakéhokoliv jazyka. Jazykové balíčky lze upravovat pomocí vestavěného webového editoru. V současné době existují balíčky pro více než 34 jazyků

- Lze použít různé metody ověřovacích mechanismů při zakládání účtu (Standardní e-mailová, LDAP, IMAP, POP3 nebo pomocí externí databáze
- Pro různé účely lze uživateli přiřadit různá práva
- Široká nabídka možných činností v kurzu: fóra, deníky, testy, materiály, hlasování, dotazníky, úkoly, chat, workshop

Výčet možností tohoto systému má široké spektrum, které bychom mohli popisovat, ale není to účelem této diplomové práce.

1.3 Programovací jazyk JAVA

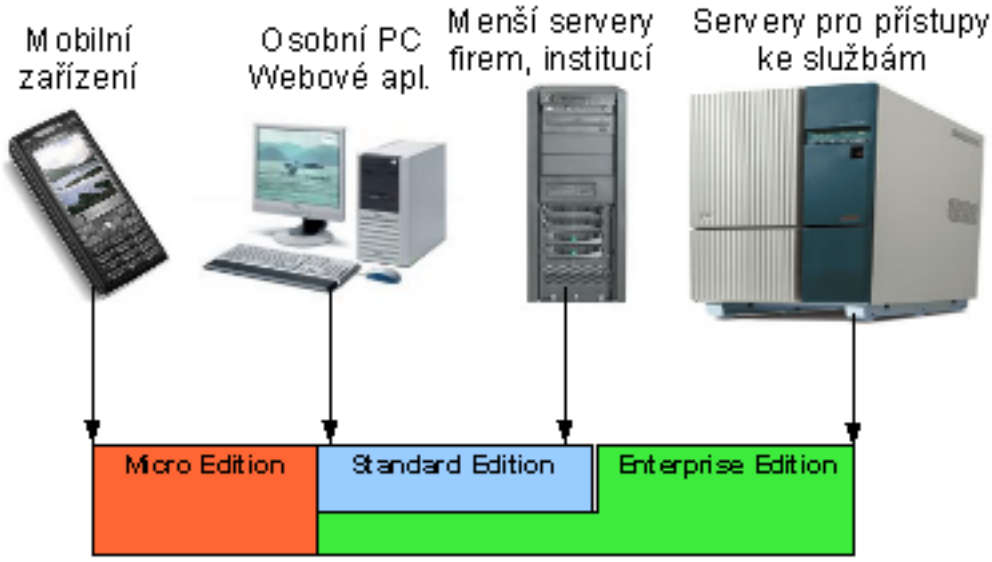
Java je objektově orientovaný programovací jazyk, který podle [4] a [5] vyvinula firma Sun Microsystems a představila ho 23. května 1995. Jedná se o jazyk vycházející z C++, ke kterému má také syntakticky nejblíž.

Java je jedním z nejpoužívanějších programovacích jazyků na světě. Díky své přenositelnosti je používána pro programy, které mají pracovat na různých systémech počínaje čipovými kartami (platforma JavaCard), přes mobilní telefony a různá zabudovaná zařízení (platforma Java ME), aplikace pro desktopové počítače (platforma Java SE) až po rozsáhlé distribuované systémy pracující na řadě spolupracujících počítačů rozprostřené po celém světě (platforma Java EE). Tyto technologie se jako celek nazývají platforma Java. Dne 8. května 2007 Sun uvolnil zdrojové kódy Javy (cca 2,5 miliónů řádků kódu) a Java bude moci být dále vyvíjena jako open source, což ještě více pomůže jejímu rozšiřování.

1.3.1 Základní vlastnosti jazyka JAVA

Podle [10] a [6] lze vlastnosti rozdělit do následujících bodů.

- jednoduchý – jeho syntaxe je zjednodušenou (a drobně upravenou) verzí syntaxe jazyka C a C++. Odpadla většina konstrukcí, které způsobovaly programátorům problémy a na druhou stranu přibyla řada užitečných rozšíření. Java také neobsahuje preprocesor (tj. žádná makra ani direktivy), není povoleno přetěžování operátorů, je zaveden jednotný zápis pro přístup k objektům i knihovnám.
- objektově orientovaný – s výjimkou osmi primitivních datových typů jsou všechny ostatní datové typy objektové.



Obr. 1.3: Přehled použití Java Platform pro jednotlivá zařízení

- distribuovaný – je navržen pro podporu aplikací v síti (podporuje různé úrovně síťového spojení, práce se vzdálenými soubory, umožňuje vytvářet distribuované klientské aplikace a servery).

- interpretovaný – místo skutečného strojového kódu se vytváří pouze tzv. mezikód (bajtkód). Tento formát je nezávislý na architektuře počítače nebo zařízení. Program pak může pracovat na libovolném počítači nebo zařízení, který má k dispozici interpret Javy, tzv. virtuální stroj Javy - Java Vitrual Machine (JVM). V pozdějších verzích Javy nebyl mezikód přímo interpretován, ale před prvním svým provedením byl program dynamicky zkompilován do strojového kódu daného počítače (tzv. just in time compilation - JIT). Tato vlastnost zásadním způsobem zrychlila provádění programů v Javě, ale také výrazně zpomalila start programů. V současnosti se převážně používají technologie zvané HotSpot compiler, které mezikód zpočátku interpretují a na základě statistik získaných z této interpretace později provedou překlad často používaných částí do strojového kódu včetně dalších dynamických optimalizací (jako je např. inlining krátkých metod atp.).

- robustní – je určen pro psaní vysoce spolehlivého softwaru – z tohoto důvodu neumožňuje některé programátorské konstrukce, které bývají častou příčinou chyb (např. správa paměti, příkaz goto, používání ukazatelů). V Javě neexistuje přímý a nekontrolovaný přístup do paměti - ukazatele jsou nahrazeny referenčními proměnnými, datové typy jsou pevně definovány [10] , provádí se

[10] shodně pro všechny hardwarové platformy

kontrola při indexaci polí, zabraňuje se přístupu ke zrušeným objektům. Java používá tzv. silnou typovou kontrolu – veškeré používané proměnné musí mít definovaný svůj datový typ. Správa paměti je realizována pomocí Garbage collectoru, který automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití. To bylo v prvních verzích opět příčinou pomalejšího běhu programů. V posledních verzích běhových prostředí je díky novým algoritmům pro garbage collection a tzv. generační správě paměti [11] tento problém ze značné části eliminován.

- bezpečný – má vlastnosti, které chrání počítač v síťovém prostředí, na kterém je program zpracováván, před nebezpečnými operacemi nebo napadením vlastního operačního systému nepřátelským kódem. Bezpečnost aplikací lze hlídat programovou kontrolou přístupu k objektům, vlastní kód programu je před spuštěním verifikován, podporovány jsou šifrovací standardy atd. Prostřednictvím výjimek lze rovněž zachytit chyby a neočekávané stavy vyskytující se za běhu programu a umožnit bezpečné zotavení aplikace.

- nezávislý na architektuře – vytvořená aplikace běží na libovolném operačním systému nebo libovolné architektuře. Ke spuštění programu je potřeba pouze to, aby byl na dané platformě instalován správný virtuální stroj. Podle konkrétní platformy se může přizpůsobit vzhled a chování aplikace.

- přenositelný – vedle zmíněné nezávislosti na architektuře je jazyk nezávislý i co se týká vlastností základních datových typů (je například explicitně určena vlastnost a velikost každého z primitivních datových typů). Přenositelností se však myslí pouze přenášení v rámci jedné platformy Javy (např. J2SE). Při přenášení mezi platformami Javy je třeba dát pozor na to, že platforma určená pro jednodušší zařízení nemusí podporovat všechny funkce dostupné na platformě pro složitější zařízení a kromě toho může definovat některé vlastní třídy doplňující nějakou speciální funkčnost nebo nahrazující třídy vyšší platformy, které jsou pro nižší platformu příliš komplikované.

  Aritmetika v Javě je definována podle normy IEEE, což zaručuje stejný výsledek na rozdílných systémech. Abecedy neanglických jazyků jsou podporovány pomocí šestnáctibitového znakového kódu Unicode.

- výkonný – přestože se jedná o jazyk interpretovaný, není ztráta výkonu významná, neboť překladače pracují v režimu, kdy se ze strojového kódu překládá jen ten kód, který je opravdu zapotřebí.

[11] paměť je rozdělena na více částí, v každé se používá jiný algoritmus pro garbage collection a objekty jsou mezi těmito částmi přesunovány podle délky svého života

- víceúlohový – Jazyk zahrnuje prostředky pro paralelní běh částí programu (multithreading) a jejich synchronizaci.

- dynamický – Java byla navržena pro nasazení ve vyvíjejícím se prostředí. Knihovna může být dynamicky za chodu rozšiřována o nové třídy a funkce, a to jak z externích zdrojů, tak vlastním programem. Java podporuje vytváření dynamických a distribuovaných aplikací i jejich automatické šíření po síti. I za běhu aplikace lze zajistit upgrade softwaru.

- elegantní – velice příjemně se v tomto jazyku pracuje, je snadno čitelný (např. i pro publikaci algoritmů), přímo vyžaduje ošetření výjimek a typovou kontrolu.

1.3.2 Nevýhody jazyka JAVA

Proti programovacím jazykům, které provádějí tzv. statickou kompilaci (např. C++), je start programů psaných v Javě pomalejší, protože prostředí musí program nejprve přeložit a potom teprve spustit. Je však možnost využít mechanismů JIT [12] a HotSpot, kdy se často prováděné nebo neefektivní části kódu přeloží do strojového kódu a program se zrychlí. Na zrychlení se také podílí nové přístupy ke správě paměti, viz výše popsaná generační správa paměti.

Další nevýhodou projevující se hlavně u jednodušších programů je větší paměťová náročnost při běhu způsobená nutností mít v paměti celé běhové prostředí.

V návrhu Javy je vidět snaha znemožnit programátorovi psát problematické konstrukce známé především z jazyka C. Součástí jazyka proto nejsou například bezznaménková čísla, příkaz goto nebo preprocesor, ačkoli se v oddůvodněných příkladech jedná o užitečné nástroje a ani rozšířené možnosti Javy je plně nenahrazují.

[12] JIT je anglický akronym Just In Time a jedná se o speciální metodu překladu aplikací urychlující běh interpretovaných programů. Program, který je spuštěn a prováděn může být interpretem v době provádění přeložen přímo do nativního kodu stroje, na kterém běží.

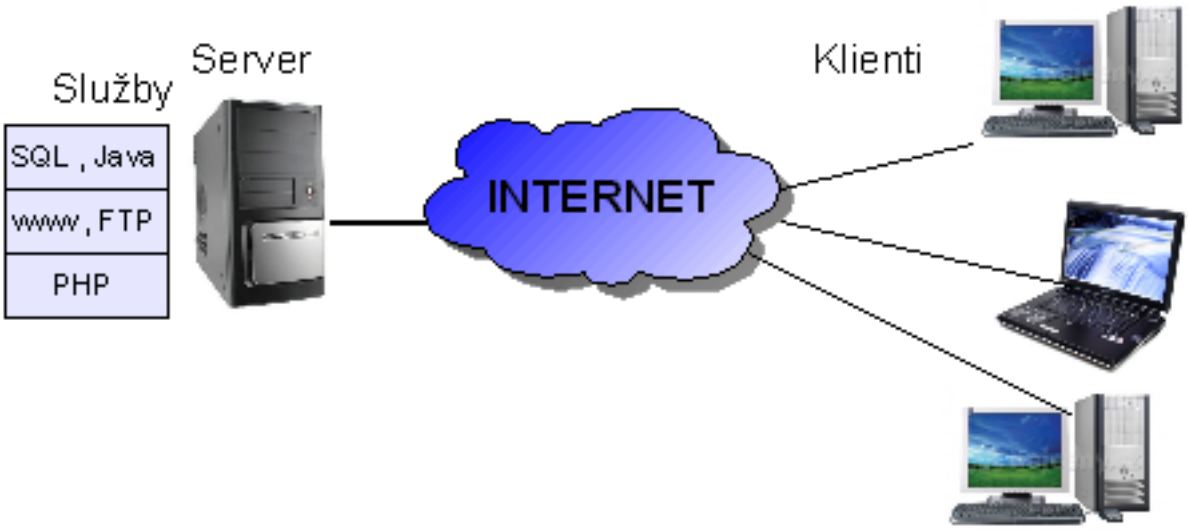
2 IMPLEMENTACE VLASTNÍHO E-LEARNING SYSTÉMU

2.1 Návrh základní koncepce

Na základě doposud získaných znalostí byla navrhnuta základní koncepce aplikace v jazyce JAVA, která bude spravovat e-learningový systém a bude umožňovat uživatelům vykonávat základní úkony spojené s údržbou obsahu jednotlivých kurzů.

Z hlediska bezpečnosti je použití pouze jedné aplikace pro přístup k databázovému serveru velice nebezpečné a umožňovalo by případnému útočníkovi celkem snadno získat všechny citlivé údaje. V klientské aplikaci by totiž někde muselo být uloženo přístupové jméno a heslo k databázi. Sice by mohlo být šifrovaně, ale díky jedné z nevýhod Javy, kterou je možnost zpětné dekompilace JAR souboru[1], by bylo možné tyto údaje dešifrovat.

Z tohoto důvodu se jeví vhodněji výslednou aplikaci rozdělit na dvě vzájemně spolupracující části. Jedná se o serverovou a klientskou část. Graficky je toto rozložení zobrazeno na obrázku 2.1



Obr. 2.1: Zobrazení struktury serverové a klientské části aplikace

Serverová část je spuštěna přímo na počítači, kde je i umístěna vlastní databáze. Zde je předpoklad bezpečné komunikace v rámci jedné smyčky localhostu. V serverové části jsou bezpečně uloženy přístupové údaje k databázi. Tato část vykonává

[1] Jedná se o zpustitelný soubor aplikace, který v sobě uchovává všechny použité třídy a knihovny

všechny SQL dotazy, které přijdou jako požadavek od klientské části aplikace. Popis tohoto pseudoprotokolu je uveden v následující kapitole.

Klientská aplikace, která se může připojovat odkudkoli z internetu, v sobě žádné údaje o připojení k vlastní databázi neuchovává. Při požadavku na získání dat se připojí na určený port serverové části aplikace, se kterou komunikuje, a veškerá data z databáze prostřednictvím ní získává.

Mezi hlavní výhody tohoto rozdělení lze zařadit:

- Nezávislost na SQL serveru - pokud potřebujeme změnit SQL server (např. z MySQL na Oracle), tak se změní pouze serverová aplikace, kdy se pouze pomocí JDBC [2] nahraje jiný potřebný ovladač a všechny ostatní funkce zůstávají neměnné.

- Jednoduchost získávání dat - v klientské aplikaci se všechna data získávají přes metody jednoho objektu, který následně iniciuje TCP spojení a stará se o vlastní komunikaci. Při programování dalších rozšířených funkcí již není nutné se starat o formu získání dat. Lze tak např. i vytvořit velmi jednoduchou aplikaci pro mobilní zařízení, která by přes tento objekt měla přístup k celému e-learning systému.

- Správné kódování řetězců - řetězce, přenášené v rámci komunikace, jsou stejně jako v celé Javě kódovány šestnáctibitovou tabulkou znaků UNICODE. Kódování ukládaných řetězců do databáze se nastavuje v serverové aplikaci, při vytváření spojení s SQL.

- Bezpečnost - SQL databáze většinou umožňují připojování a zasílání dotazů pouze z lokálního počítače. Komunikace probíhá v nešifrované podobě a případný útočník by lehce mohl odposlechnout přístupové jméno a heslo. Takto klient nekomunikuje přímo s SQL databází, ale zprostředkovaně, přes serverovou část. Nemusí tedy znát přístupové údaje pro zadávání dotazů v databázi. Nemusí být také připojen lokálně. Volitelně lze i komunikaci mezi serverem a klientem šifrovat. Vhodná by byla symetrická šifra s využitím např. bezpečného ustavení klíče Diffie-Hellmanovým protokolem.

- Jednoduchá rozšiřitelnost - Při tvorbě klientské aplikace byl kladen hlavní důraz na univerzálnost. Jádrem aplikace je několik objektů, které umožňují správu přihlašování, řízení komunikace se serverem, nastavení důležitých údajů a jejich ukládání do INI souboru, logování důležitých zpráv, apod. Pokud se

[2] JDBC (Java database connectivity) - definuje jednotné rozhraní pro přístup k relačním databázím. JDBC je součástí Javy SE od JDK 1.1. Pro přístup ke konkrétnímu databázovému serveru je potřeba JDBC ovladač, který poskytuje tvůrce databázového serveru.

tedy vytváří nějaká nová funkčnost nebo nový modul, není potřeba již tyto věci ošetřovat a lze se soustředit už jen např. na vlastní interpretaci získaných dat apod. Blíže je tato problematika rozebrána v kapitole 3. Rovněž je možné na jednom serveru provozovat více e-learning systémů, rozlišených od sebe číslem portu. Klient by se tak mohl s jednou stejnou klientskou aplikací připojovat k různým systémům, například učitel, vyučující zároveň na více různých školách apod.

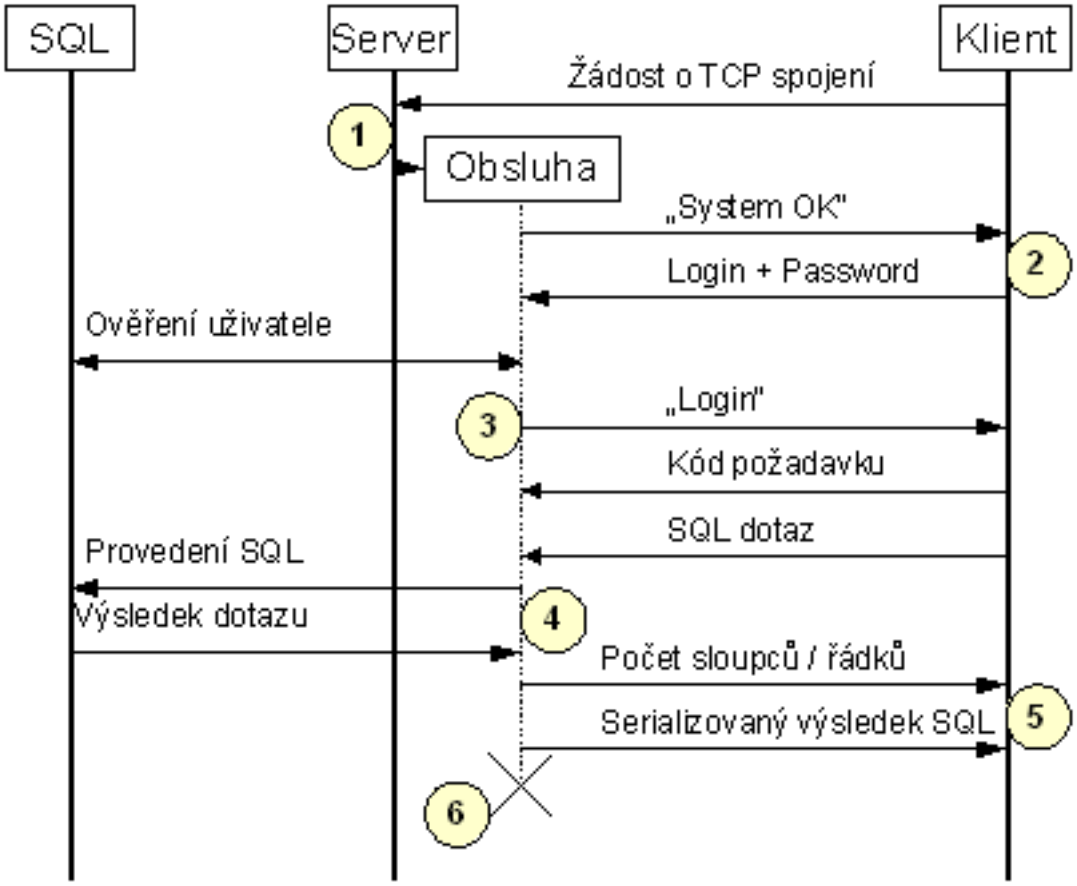
2.2 Způsob komunikace

Jak již bylo naznačeno, z důvodu bezpečnosti a zabránění případnému útočníkovi k přímému přístupu do databáze je aplikace rozdělena na klientskou a serverovou část. Kvůli komunikaci a vzájemné výměně dat obou částí bylo nutné vytvořit určitý pseudoprotokol, který by klientovi umožnil se u serveru identifikovat, zaslat požadavek (např. výpis nějaké SQL tabulky) a přijmout výsledek v nějakém dohodnutém tvaru a tento následně zpracovat. Graficky je průběh této komunikace mezi oběma částmi aplikace zobrazen na následujícím obrázku 2.2. Důležité body komunikace jsou označeny číslem ve světle žlutém kruhu a jsou v dalším textu podrobněji rozvedeny.

2.2.1 Iniciace spojení

Serverová část aplikace využívá jednu z dalších výhod programovacího jazyka Java. Jedná se konkrétně o to, že jazyk zahrnuje prostředky pro paralelní běh částí programu (multithreading) a jejich synchronizaci. Jedno (případně i další) vlákno obsluhuje události GUI aplikace. Na pozadí pak běží další vlákno, které poslouchá na definovaném portu. Pokud tedy klient na tomto portu iniciuje TCP spojení (viz. obrázek 2.2,bod 1), pak vlákno „Server“ tuto žádost přijme. Následně toto vlákno vytvoří objekt „Obsluha“, kterému toto nové spojení předá jako parametr. Objekt „Obsluha“ představuje nové vlákno, které běží na pozadí a komunikuje s klientskou stranou. Výhodou tohoto řešení je to, že aktuální komunikace s klientem neblokuje Serverové vlákno, které může nadále naslouchat na daném portu a v případě potřeby vytvářet nová vlákna pro obsluhu dalších klientů.

Nově vytvořené vlákno „Obsluha“ v této fázi převezme spojení s klientem. Následně si ověří funkčnost SQL databáze. V případě chyby zašle pouze klientovi chybovou hlášku a běh vlákna se ukončí. Pokud je vše v pořádku, zašle ke klientovi zprávu `System OK` . Jakmile klient tuto zprávu obdrží (viz. obrázek 2.2,bod 2), může pokračovat zasláním přihlašovacích údajů uživatele, který chce se systémem pracovat. Protistrana tyto údaje přijme a jednoduchým SQL dotazem ověří, zda je



Obr. 2.2: Grafické vyjádření průběhu komunikace mezi serverovou a klientskou částí

přihlášení správné. Pokud ano, zašle klientovi zprávu `Login` . V opačném případě odešle klientovi zprávu `Login Failure` - přihlášení se nezdařilo a ukončí se spojení.

2.2.2 Odeslání SQL dotazu

Na obrázku 2.2 v bodě 2 již jsme ve stavu, že klient ví, že přihlášení proběhlo v pořádku. Nyní vlákno obsluhy očekává od klienta zaslání požadavku. Protože ale požadavek od klienta může mít více forem, bylo přistoupeno k dodatečnému rozlišení pomocí signálu pojmenovanému jako „Kód požadavku“. Popis hodnot signálu a jejich význam je vysvětlen v následující tabulce 2.1.

2.2.3 Provedení SQL dotazu na straně serveru

V případě, že tedy klient zvolí kód požadavku 01 nebo 02, následuje odeslaní SQL dotazu v prosté textové formě. Server tento dotaz převezme a postará se o jeho provedení v SQL databázi. V Javě se pro spojení s databází využívá ovladač JDBC,

Tab. 2.1: Kód požadavku a jeho význam

Kód	Význam
01	Požadavek bude obsahovat SQL dotaz typu SELECT, tzn. že výsledkem bude např. vypsání obsahu nějaké tabulky. Princip přenosu tohoto výsledku bude popsán v následující kapitole. Zpět ke klientovi pak následně bude server posílat signály „Počet sloupců“ „Serializovaný výsledek SQL“
02	V rámci požadavku bude odeslán SQL dotaz typu INSERT,UPDATE,DELETE. Jako výsledek bude očekáván pouze počet řádků, které dotaz nějak ovlivnil (např. počet modifikovaných řádků). Tzn. že zpět ke klientovi se již nebude přenášet signál „Serializovaný výsledek SQL“.
03	Klient pouze ověřuje správnost přihlašovacích údajů. Ta již proběhla, proto po přijetí tohoto kódu spojení končí.
04	Kód rezervován pro budoucí rozšiřování, např. pro nové typy požadavků.

který umožňuje připojit se k různým druhům databáze. Programátor se tak tedy nemusí starat o způsob práce s jednotlivými druhy databázových serverů a může se soustředit pouze na vlastní práci se získanými daty a jejich reprezentaci např. v GUI.

V následujícím zdrojovém kódu je uveden příklad vytvoření třídy, díky které lze přistupovat do databáze a získat z ní data. Jednoduchý popis tohoto kódu je uveden následně.

```
import java.sql.*;

public class mojeSQL {

    private Connection conn = null;
    private Statement  s    = null;
    private ResultSet  rs   = null;

    /** Creates a new instance of mojeSQL */
    public mojeSQL() {
        //konstruktor
    }

    public void Pripoj () {

           try
```

```
            {
                Class.forName("org.gjt.mm.mysql.Driver").newInstance();
                this.conn = DriverManager.getConnection
                 ("jdbc:mysql://localhost/jmenodatabaze?useUnicode=true&
                 characterEncoding=UTF-8&user=uzivatel&password=heslo);
            }
            catch (Exception e)
            {
                //osetreni chybneho pripojeni k databázi
            }
            //nyní je již vytvořeno připojení k databázi

            s = conn.createStatement();
            this.s.executeQuery("SELECT * FROM 'KURZ'");
            this.rs = s.getResultSet();
            ResultSetMetaData rsmd = rs.getMetaData();
            int PocetSloupcu = rsmd.getColumnCount();
            //vysledek SQL dotazu je reprezentovan objektem rs
        }

        public int Odpoj () {
           if (this.conn != null)
                  {
                      try
                      {
                          this.conn.close ();
                      }
                      catch (Exception e) { /* ošetření chybného ukončení */ }
                  }
           return 0;
        }
}
```

Pro vlastní připojení jsou důležité 3 objekty definované jako privátní atributy třídy. Jedná se o JDBC API, což je specifická skupina abstraktních Java rozhraní. Patří mezi ně :

- Connection - reprezentuje spojení s konkrétní databází.

- Statement - představuje kontejner pro spouštění SQL příkazu na daném spojení.
- ResultSet - kontroluje přístup k řádkům vráceného výsledku.

Po zavolání metody `Pripoj()` je tedy nejprve nahrán příslušný JDBC ovladač. Následně je do objektu `Connection` přiřazeno nově vytvořené spojení s databází definované pomocí metody `getConnection`. Té je jako parametr předán speciální řetězec, ve kterém je definováno, k jakému typu SQL databáze se má připojit, na jaké IP adrese se server nachází (zde použit localhost) a nad kterou databází se bude pracovat.

Mezi další náležitosti, předávané jako parametr za znak „?“, patří uvedení, v jakém kódování bude zasílán text a přihlašovací údaje pro práci s databází. Celý tento proces připojení je nutné v kódu uvést v bloku `try-catch`. Je to hlavně z důvodu následného ošetření selhání spojení. Lze tak odpovídajícím způsobem zareagovat. Poté se na základě tohoto spojení vytvoří objekt, který umožňuje spouštět SQL dotazy.

Zadání dotazů se provádí metodou `executeQuery("SELECT * FROM 'KURZ'");`. Např. zde se jedná o jednoduchý výpis tabulky KURZ, vypíší se všechny řádky a všechny sloupce. Pokud nechceme použít dotaz SELECT na výpis tabulky, ale např. INSERT na vložení nového řádku, napíšeme metodu `executeUpdate("...");`, která jako návratovou hodnotu má celočíselný `Integer`, udávající počet řádků, kterých se tento dotaz týkal.

Vraťme se ještě k tomu složitějšímu, a to k provedení SQL dotazu `SELECT`. Jelikož výsledkem tohoto dotazu je složitý datový typ, tak funkce `executeQuery(...)` nemá žádnou primitivní návratovou hodnotu, ale pro získání výsledku je nutné naplnit objekt `ResultSet`. Ten umožňuje vlastní práci s výsledkem dotazu. Zpracování probíhá po řádcích sekvenčně, tzn. že na začátku je aktuální nultý řádek a voláním metody `Next()` se pomyslný ukazatel přesouvá na další řádky. V každém z řádků můžeme vyčíst hodnoty příslušející jednotlivým sloupcům mnoha funkcemi, které se liší v typu hodnot, které vracejí.

Například metoda `getString(sloupec)` vrací hodnotu daného sloupce jako String (řetězec). Sloupec lze této metodě zadat jako jeho název, tedy třeba „id“ nebo „poznamka“. V případě, že dopředu neznáme jména sloupců, lze jako parametr uvést i pořadové číslo sloupce. Pokud víme, že sloupec obsahuje datový typ Integer nebo Date, lze použít podobně pracující metody vracející příslušné typy.

2.2.4 Odeslání výsledku dotazu klientovi

Pokud bychom v tuto chvíli nějakým způsobem potřebovali pracovat s výsledkem SQL dotazu, jednalo by se jen o jednoduché programovací smyčky, které by postupně vyčítaly např. hodnoty z každého řádku. My ale potřebujeme tento výsledek nějak přenést zpátky na přijímací stranu ke klientovi. A zde nástává jeden z větších problémů. Doposud si klientská a serverová strana zasílaly pouze jednoduché datové typy (ať už celočíselný typ Integer, či řetězec String). Při komunikaci mezi oběma stranama je nutné vše převést na proud bajtů, tzn. provést serializaci.

Pokud posíláme již zmíněné jednoduché typy, tak Java sama provádí serializaci a výsledek předává do objektu `ObjectInputStream`, resp. `ObjectOutputStream`. Tyto objekty představují oba směry toku dat ve vytvořeném TCP spojení klienta a serveru.

Pro čtení příchozích dat se používá metoda `readObject()` , která vrací referenci na nově takto přijatý objekt. Jelikož se jedná o obecný objekt, je nutné ho ještě správně přetypovat, jak je naznačeno v následujícím řádku kódu.

```
prijem = (String) ObjectInputStream.readObject();
```

Přijatý proud dat je správně převeden na původní objekt, včetně přetypování, zde přetypováno na objekt reprezentující textový řetězec. Překladač v tomto případě ještě vyžaduje tento příkaz uzavřít do bloku `try-catch`. Nebo musí mít funkce, ve které je příkaz uveden, ošetřeno špatné dekódovaní přijatého objektu. Toto ošetření se provádí přidáním klíčových slov `throws ClassNotFoundException`. Tato klíčová slova se připíší do hlavičky funkce.

Pro zápis odchozích dat, která chceme odeslat, se používá metoda `writeObject()`, které jako parametr předáme vytvořený objekt určený k transportu. Nelze však takto odesílat všechny objekty. Nutnou podmínkou je, aby implementovaly rozhraní `Serializable` (pokud se pokusíme serializovat nevyhovující objekt, dočkáme se výjimky NotSerializableException). Protože se instance serializuje i se všemi odkazovanými objekty, musí být i tyto serializovatelné, anebo musí být jejich referenční proměnné označené modifikátorem `transient` (tedy že nebudou uloženy).

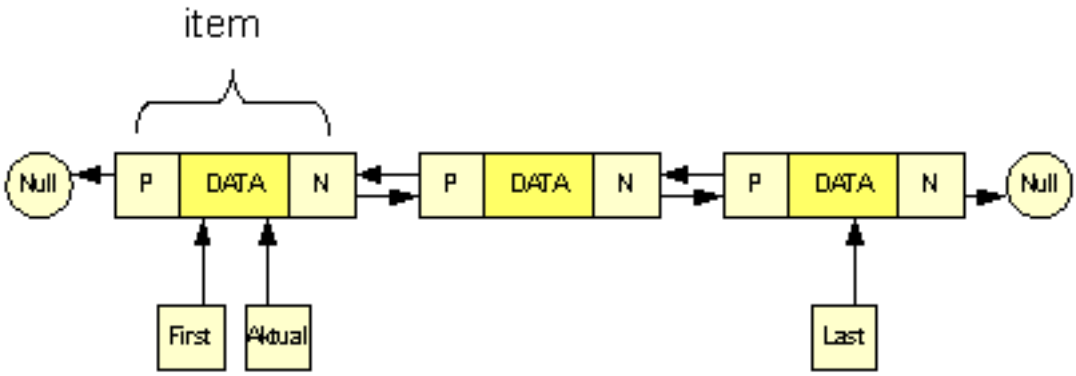
Pokud se tedy vrátíme zpátky k původnímu problému, je třeba výsledek SQL dotazu nějakým způsobem serializovat, aby mohl být odeslán a na straně klienta převeden do původní podoby. Jak již bylo řečeno, k vlastnímu výsledku SQL dotazu lze přistoupit přes objekt `ResultSet`. Při pohledu do dokumentace ale zjistíme, že tento objekt neimplementuje rozhraní `Serializable`, tedy že ho nelze převést na proud bytů. Další překážkou je i to, že objekt `ResultSet` je podle [8] vázán na objekt, reprezentující datové spojení s SQL databází. Jakmile toto spojení skončí, je objekt `ResultSet` současně zrušen. Z tohoto důvodu bylo nutné najít alternativu, která

by umožňovala nějakým způsobem uchovávat výsledek dotazu a zároveň by byla serializovatelná. Nabízelo se mnoho řešení, ale jako nejschůdnější se jevila vlastní implementace dvojitě vázaného seznamu[3].

Dvojitě vázaný seznam paří do skupiny lineárních seznamů, což je dynamická datová struktura, vzdáleně podobná poli (umožňuje uchovat velké množství hodnot ale jiným způsobem), obsahující jednu a více datových položek (struktur) stejného typu, které jsou navzájem lineárně provázany vzájemnými odkazy pomocí ukazatelů nebo referencí. Aby byl seznam lineární, nesmí existovat cykly ve vzájemných odkazech.

Lineární seznamy mohou existovat jednosměrné a obousměrné (dvojitě vázané). V jednosměrném seznamu odkazuje každá položka na položku následující a v obousměrném seznamu odkazuje položka na následující i předcházející položky. Zavádí se také ukazatel nebo reference na aktuální (vybraný) prvek seznamu. Na konci (a začátku) seznamu musí být definována zarážka označující konec seznamu. Pokud vytvoříme cyklus tak, že konec seznamu navážeme na jeho počátek, jedná se o kruhový seznam.

Datová struktura dvojitě vázaného seznamu je zobrazena na obrázku 2.3



Obr. 2.3: Reprezentace objektu dvojitě vázaného seznamu v paměti

Nejprve tedy bylo nutné vytvořit třídu, reprezentující jednu položku seznamu. V obrázku 2.3 je označena jako „item“. Atributy třídy jsou dvě reference na předchozí a následující položku. Samozřejmě i datový atribut, zde zvolený jako String (textový řetězec). Z hlediska objektového programování je výhodnější tyto atributy mít privátní [4], tzn. navíc byly ještě vytvořeny funkce, umožňující číst i nastavovat data a reference na předchozí a následující položku. Zdrojový kód je pro ilustraci uveden v příloze B.1

[3] v literatuře ozačováno jako Double Linked List

[4] atributy jsou viditelné pouze pro metody ve vlastní třídě, z jiných objektů k nim nelze přistupovat

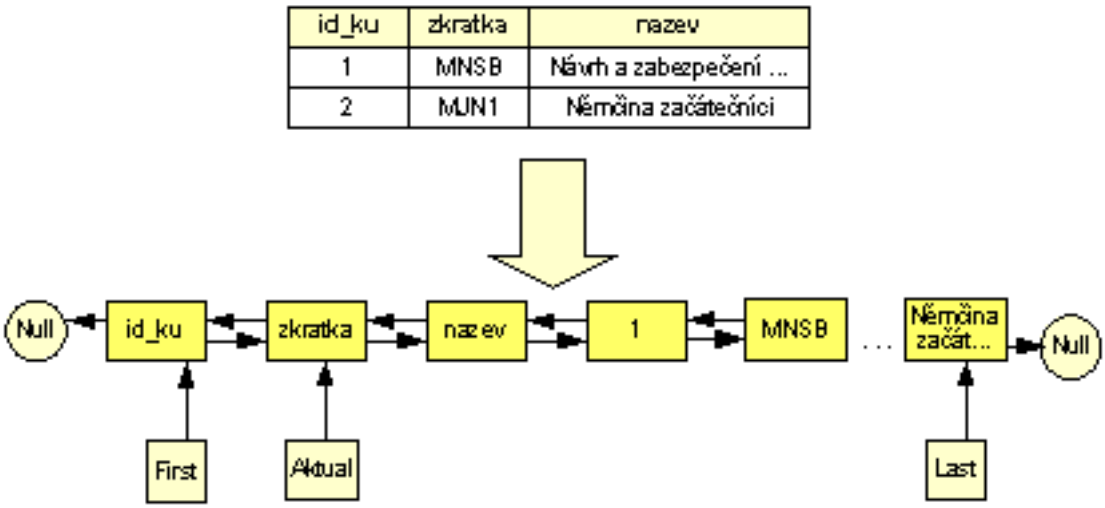
Tyto položky lze tedy řadit za sebe podobně, jako je to naznačeno na obrázku 2.3. Práce s těmito položkami by ale v tomto stavu byla velmi složitá, proto nyní definujeme objekt dvojitě vázaného seznamu, který by měl umožňovat jednoduchou a hlavně spolehlivou práci s položkami. Základem je nějakým způsobem označit začátek a konec takovéhoto provázaného seznamu (v obrázku 2.3 označeno jako First a Last). Podobně potřebujeme i nějaký „pracovní“ ukazatel, který by sloužil k pohybu po seznamu (v obrázku označeno Aktual).

Na základě těchto vlastností a dle definic [8] byly dále implementovány funkce umožňující přidávat nové položky na začátek nebo konec seznamu, nastavit aktuální kurzor na první nebo poslední položku, přidávat novou položku před nebo za aktuální kurzor a v neposlední řadě i funkce odebírající prvky ze začátku a konce, nebo odebrání aktuálního prvku. Při těchto metodách bylo nutné hlavně ošetřit, aby např. při odebírání aktuálního prvku nedošlo k špatnému nastavení ukazatelů, tzn. aby prvky jdoucí po sobě vždy měly správného předchůdce i následovníka. Všechny metody jsou rovněž označené jako „synchronized“. Je to z toho důvodu, že pokud by s jedním seznamem např. pracovala dvě vlákna aplikace, mohlo by se stát, že by jedno vlákno nedokončilo svou operaci (např. přidání nového prvku) a mezitím by druhé vlákno třeba první prvek smazalo. Tím by mohlo dojít k nekonzistenci dat. Klíčové slovo „synchronized“ zaručuje, že po dobu vykonávání takto označené funkce nesmí dojít k přerušení práce vlákna a předání procesorového času vláknu druhému. Není ale vhodné takto zbytečně označovat obsáhlé funkce, které by zbytečně dlouho blokovaly procesor. Místo toho lze v rozsáhlejších metodách umístit do speciálního bloku pouze sérii příkazů, které požadujeme vykonat ve vlákně nepřerušeně. Implementace třídy, ve které jsou tyto metody definovány, je uvedena v příloze B.2.

V této chvíli tedy máme připravenou datovou strukturu vhodnou pro uložení výsledku SQL dotazu. Tuto strukturu lze i serializovat a předat jako proud dat vytvořenému spojení. Ze začátku bude seznam prázdný, tedy všechny tři ukazatelé obsahují hodnotu NULL. Pro zapisování nových hodnot použijeme u dvojitě vázaného seznamu metodu `addLast`, aby hodnoty byly uložené v pořadí, v jakém jsou čteny.

Nejprve tedy tímto způsobem do seznamu uložíme jednotlivé názvy sloupců v tom pořadí, jaké nám vrátí SQL server v objektu `RecordSet`. Následně z prvního řádku vyčítáme hodnoty sloupec po sloupci a jako String je ukládáme vždy na konec seznamu. Když dojdeme na konec, pokračujeme dalším řádkem a tak dále, až budeme na posledním řádku. Tabulku, která má např. 3 sloupce a obsahuje 2 řádky záznamů, převedeme tímto způsobem na pole o 9ti prvcích. Názorně to je zakreslené na obrázku 2.4

Pokud se tedy vrátíme ke scénáři komunikace mezi serverem a klientem, která

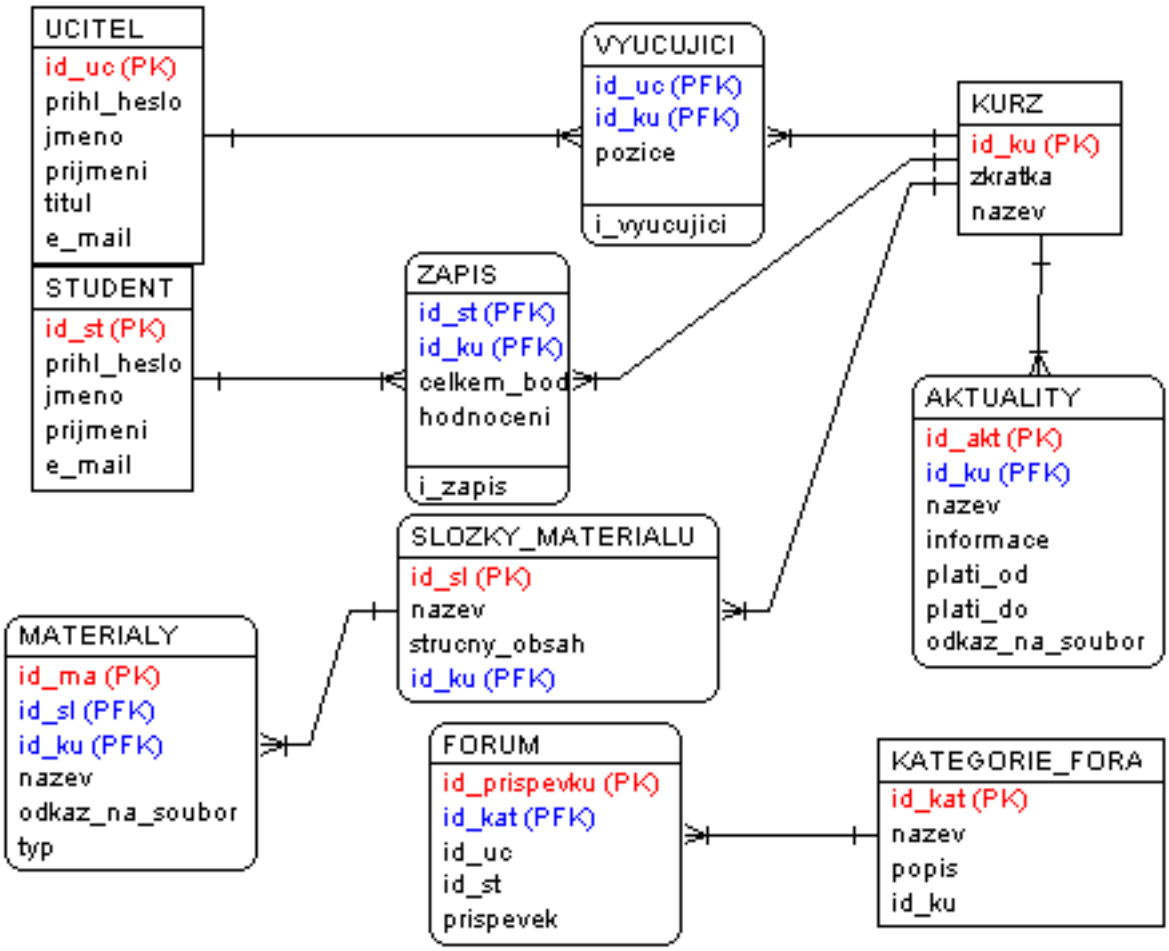


Obr. 2.4: Znázornění průběhu převodu SQL tabulky na prvky dvojitě vázaného seznamu

je zobrazena na obrázku 2.2, tak v bodě 3 klient zašle SQL dotaz např. na výpis tabulky KURZ. Server jeho požadavek přijme a dotaz přepošle do SQL databáze, která vrátí výsledek ve formě tabulky v objektu `RecordSet`. Tento výsledek je následně výše uvedeným postupem převeden na dvojitě vázaný seznam. Poté je v bodě 5 klientovi nejdříve zaslán počet sloupců výsledku a hned poté i serializovaný seznam s hodnotami výsledku. Spojení je následně ukončeno. Aplikace na klientské straně může jednoduchým cyklem FOR vyčítat hodnoty z přijatého seznamu a rekonstruovat z nich původní tabulku. Přečte prvních N hodnot, kde N představuje přijatou hodnotu o počtu sloupců. Těchto N hodnot představuje posloupnost názvů sloupců. Poté přečte dalších N hodnot, což reprezentuje první řádek tabulky. Takto pokračuje až do konce seznamu. Počet jeho prvků je tedy dělitelný číslem N.

2.3 Struktura tabulek v databázi SQL

Celá aplikace e-learningu je založena hlavně na práci s databází. Pro aplikaci e-learningu byl navrhnut ER diagram SQL databáze. Vlastní struktura databáze je zobrazena na obrázku 2.5. Nedílnou součástí je i DF diagram, který je zobrazený na obrázku v příloze A.1. Zde jsou popsány všechny možné procesy pro každý ze tří typů uživatelů (naznačeno přerušovanou čarou). Názorně je také vidět, z jakých tabulek jsou pro daný proces použita data (značeno plnou čarou, pokud vede šipka pouze od datového uložiště k procesu, tak proces má disponovat pouze právem ke čtení). Popis procesů, stejně jako rozbor struktury SQL tabulek je uveden v následující kapitole 2.3.1.



Obr. 2.5: ER diagram SQL databáze pro e-learningovou aplikaci

2.3.1 Popis struktury databáze

Základem databáze e-learningu jsou tabulky UCITEL, STUDENT, KURZ. Popisují hlavní entity, které se účastní všech procesů v rámci e-learningu. V každé ze tří těchto tabulek se nachází primární klíč díky, kterému je každý záznam rozlišen a je unikátní. Další atributy slouží k bližšímu popisu, u tabulky UCITEL a STUDENT je navíc pole obsahující heslo, které slouží k přihlášení uživatele do systému. Administraci všech tří tabulek zajišťuje proces označený v obrázku A.1 jako „1 Správa kurzů, učitelů, studentů“. Tento proces umožňuje zobrazení přehledu, přidání nové položky a editaci vybrané položky. Přímé smazání položky z tabulky je řešeno přidáním atributu „aktivni“, který je celočíselného typu. Při mazání se tak vlastně jen tento parametr nastaví na hodnotu 0 a daný záznam se pak v systému považuje za smazaný, i když fyzicky existuje. Toto opatření slouží k tomu, aby nedošlo k nekonzistenci dat. Pokud bychom fyzicky smazali např. jednu položku z tabulky KURZ a její ID by se současně vyskytovalo v tabulce ZAPIS, tak v této tabulce by

posléze zůstal nekorektní řádek, který by pomocí cizího klíče „ID kurzu“ odkazoval na neexistující záznam.

Nyní se zmíníme o dalších tabulkách, které jsou na obrázku 2.5. Tabulka VYUCUJICI představuje pomocnou tabulku pro realizaci vazby M:N mezi tabulkami UCITEL a KURZ. Tato tabulka obsahuje informaci o učiteli a předmětu, který vyučuje. Dále je tato tabulka doplněna o popis pozice. Je zde definován i index, obsahující „ID učitele“ i „ID kurzu“, tzn. že v tabulce nemohou být dva stejné záznamy. Dvojice Učitel-Kurz musí být pro každý záznam unikátní.

Tabulka ZAPIS představuje něco podobného, co předcházející, jen s tím rozdílem, že zde se jedná o vazbu M:N mezi tabulkou STUDENT a KURZ. Je zde uvedeno, v jakých kurzech je daný student zapsán. Tato informace je doplněna hodnocením a počtem bodů. Podobně je zde i definován index.

Správu těchto tabulek podle DF diagramu na obrázku A.1 zajišťuje výhradně Admin, tzn. provádí registraci pro jednotlivé studenty a pro kurzy přiřazuje učitele. V této pozici může fungovat případně i studijní oddělení nějaké školy. Co se týká tabulky ZAPIS, tak třeba uživatel typu Učitel už nemůže záznamy přidávat, ani mazat, může pouze provádět hodnocení (tzn. editovat položky „celkem bodu“ a „hodnoceni“). Uživatel typu Student pak z tabulky ZAPIS může číst jen jeho záznamy a zjistit si své hodnocení v zapsaných předmětech.

Každý kurz samozřejmě může obsahovat více aktualit, kde Vyučující má možnost zapsaným studentům sdělit důležité informace, které se týkají příslušného kurzu. Aktuality jsou uložené ve stejnojmenné tabulce. Každý záznam má svůj unikátní identifikátor a současně obsahuje i ID kurzu, kterého se daná aktualita týká. Součástí každého takového záznamu je i časový údaj o platnosti, vlastní informace a případně i odkaz na soubor v internetu.

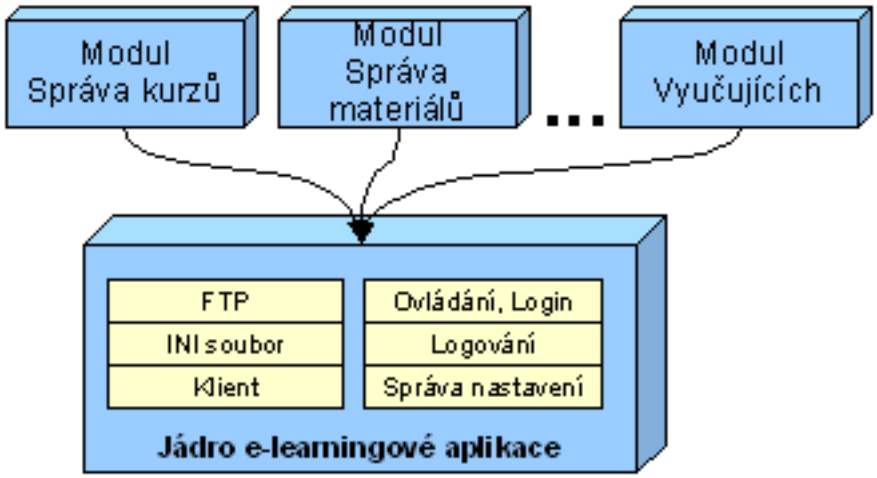
Součástí celého systému je také fórum, které by mělo sloužit ke vzájemné komunikaci jak mezi učiteli a studenty, tak mezi studenty samotnými. Fórum je tvořeno dvěma tabulkami. V první tabulce je seznam jednotlivých kategorií doplněné názvem a popisem fóry, v druhé tabulce jsou uvedeny všechny příspěvky z fóra. Každý příspěvek je rozlišen pomocí ID kategorie, do které patří, a také podle ID studenta (resp. i učitele), který daný příspěvek odeslal.

Poslední dosud nepopsanou částí zůstává Správa materiálů. Učitel pro každý předmět, kde je zapsán jako vyučující, může vytvářet a upravovat jednotlivé složky. Každou složku lze pojmenovat a stručně popsat. (Např.: „Přednášky“, „Laboratoře“,atd.) Následně lze do každé takto vytvořené složky vkládat již samotné výukové materiály. Pro každý takový materiál se uvede jeho stručný název, delší popis (255 znaků), případně i jeho typ (obrázek, PDF, ...) a je vybrán soubor, který je pomocí FTP uložen na server. V tabulce se pak uloží odkaz na tento soubor, který si pak mohou studenti ze systému stáhnout. Popisu FTP připojení se věnuje jedna

z následujících kapitol.

2.4 Modularita systému

Při návrhu základní koncepce byl kladen důraz na to, aby byla samotná aplikace velmi jednoduše rozšiřitelná (dá se říci také modulární), aby se případné budoucí rozšiřování o další funkce obešlo bez větších zásahů do stávajícího kódu. Z tohoto důvodu byl zvolen model, kdy je aplikace tvořena jádrem, ke kterému lze připojovat moduly s potřebnou funkcionalitou. Znázorněno na obrázku 2.6.



Obr. 2.6: Grafické znázornění struktury výsledné aplikace

Samotné jádro obsahuje důležité funkce nutné pro běh celého systému. Jsou zde implementovány následující funkce :

- Správa nastavení - je možné uživatelsky měnit některé parametry aplikace (např. IP adresa serveru, ke které se chce klient připojit, případně číslo portu, apod.) Aby se toto nastavení po ukončení aplikace neztratilo, ukládají se důležité hodnoty do INI souboru.
- Logování - umožňuje připojeným modulům vypsat některé důležité informace jednotným způsobem do logovacího okna.
- Správu přihlašování - centrálně je řízeno správné přihlášení a ověřování přihlašovacích údajů. Na základě přihlášení jsou pak uživateli zpřístupněny příslušné funkce v menu aplikace. Toto blíže rozebírá kap.3.
- FTP přístup - zprostředkovává modulům FTP připojení do souborového prostoru systému. Vhodné pro ukládání např. materiálů pro jednotlivé kurzy. Dále popsáno v následující kapitole.

- Komunikace se serverem - umožňuje jiným modulům zasílat SQL dotazy a následně přijmout výsledek, který se zpracuje. Při vytváření nového modulu tak nemusí programátor ošetřovat průběh vlastního spojení, který je zakreslen na obrázku 2.2. Pouze si vytvoří objekt, reprezentující klientské připojení. Parametry spojení (IP adresa serveru a jeho port) se jednoduše získají z nastavení aplikace. Pak už se pouze zavolá metoda a jako parametr se jí předá SQL dotaz. Po vykonání metody se výsledek nachází ve veřejném atributu objektu.

Programování modulů se pak může omezit už jen na vlastní práci z daty a jejich správné interpretaci. Většinou se moduly skládají z grafického okna, ve kterém je například tabulka zobrazující přehled učitelů. Dále je zde třeba tlačítko pro vložení nového záznamu, nebo pro jeho smazání či editaci, apod.

Instalace nového modulu pak znamená pouze nahrání příslušných JAVA a FORM souborů k aplikaci a vytvoření nové položky v hlavním menu, která tento modul spouští. Každý modul je spouštěn v novém okně.

2.5 Implementace FTP přenosů

Jak již bylo zmíněno výše, jádro aplikace umožňuje modulům jednoduchý transport souborů pomocí protokolu FTP. Vlastní implementace je založena na jednoduchosti. Funkce v modulech budou potřebovat pouze zvolený soubor přenést na FTP server do zadané složky. Není tedy potřeba vytvářet žádné speciální úpravy ani grafické okno.

Z tohoto důvodu byla vybrána již existující knihovna, podporující pro jazyk Java FTP přenosy. K dispozici ke stažení včetně kvalitní dokumentace je na [11]. Tato knihovna se jmenuje „edtftpj“ a je distribuována pod licencí LGPL [5]

Podle dokumentace lze relativně rychle vytvořit jednoduchého FTP klienta, který se umí přihlásit, nastavit aktuální adresář, nastavit mód přenosu, následně odeslat / přijmout potřebný soubor a spojení korektně ukončit. Zdrojový kód takovéto funkce je uveden níže.

```
import java.io.IOException;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import com.enterprisedt.net.ftp.FTPConnectMode;
```

[5]GNU Lesser General Public License, zkráceně GNU LGPL, je varianta licence pro svobodný software GNU GPL, která ale na rozdíl od GPL umožňuje spojování s nesvobodným kódem, a tak je vhodnější např. pro knihovny (za předpokladu, že autor jejich použití v nesvobodných programech chce umožnit).

```
import com.enterprisedt.net.ftp.FTPTransferType;
import com.enterprisedt.net.ftp.FileTransferClient;
import com.enterprisedt.net.ftp.FTPFile;
import com.enterprisedt.util.debug.Level;
import com.enterprisedt.util.debug.Logger;



  FileTransferClient ftp = null; //reprezentace FTP připojení

  try {
      // create client
      ftp = new FileTransferClient();

      // set remote host
      ftp.setRemoteHost(this.ggg.IPadresa); //na jakou IP se připojit
      ftp.setUserName("vems");    //přihlašovací jméno
      ftp.setPassword("vems");    //přihlašovací heslo

      // connect to the server
      ftp.connect();
      //nastavení pasivního módu
      ftp.getAdvancedFTPSettings().setConnectMode(FTPConnectMode.PASV);
      //binární přenos dat
      ftp.setContentType(FTPTransferType.BINARY);
      //nastavení vzdáleného adresáře
      ftp.changeDirectory("file");

      //vlastní UPLOAD souboru, zadává se cesta, kde se soubor nachází,
      //a jak se má soubor jmenovat na FTP serveru
      ftp.uploadFile(this.mypath, this.myfile);

      //odpojení
      ftp.disconnect();

  } catch (Exception e) {
      e.printStackTrace();
  }
```

3 POPIS APLIKACE

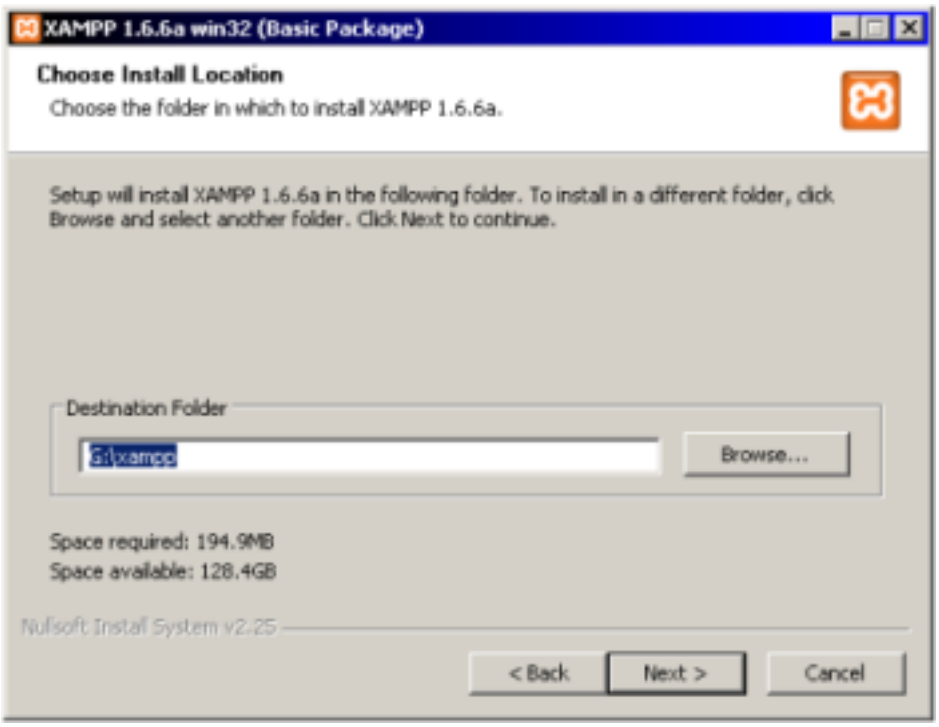
Tato kapitola umožňuje získat znalosti ohledně administrace systému a základního uživatelského ovládání. Popisuje i samotnou instalaci systému na serverové straně aplikace. Celá kapitola je koncipována tak, aby mohla sloužit i jako manuál pro používání systému.

3.1 Instalace systému

Při instalaci se předpokládá, že na počítači současně neběží žádný další webový ani databázový server. Instalace je popisována pro operační systém Microsoft Windows XP. Pro systémy na bázi UNIX existuje XAMPP také, ale instalace probíhá odlišně v závislosti na použité distribuci.

3.1.1 Instalace a zprovoznění programu XAMPP

Program XAMPP v sobě obsahuje webový server Apache s podporou jazyka PHP a Perl. Jeho součástí je i MySQL server s předchystaným prostředním PhpMyAdmin. Z CD tedy spustíme soubor `xampp-win32-installer.exe`. Po úvodním okně



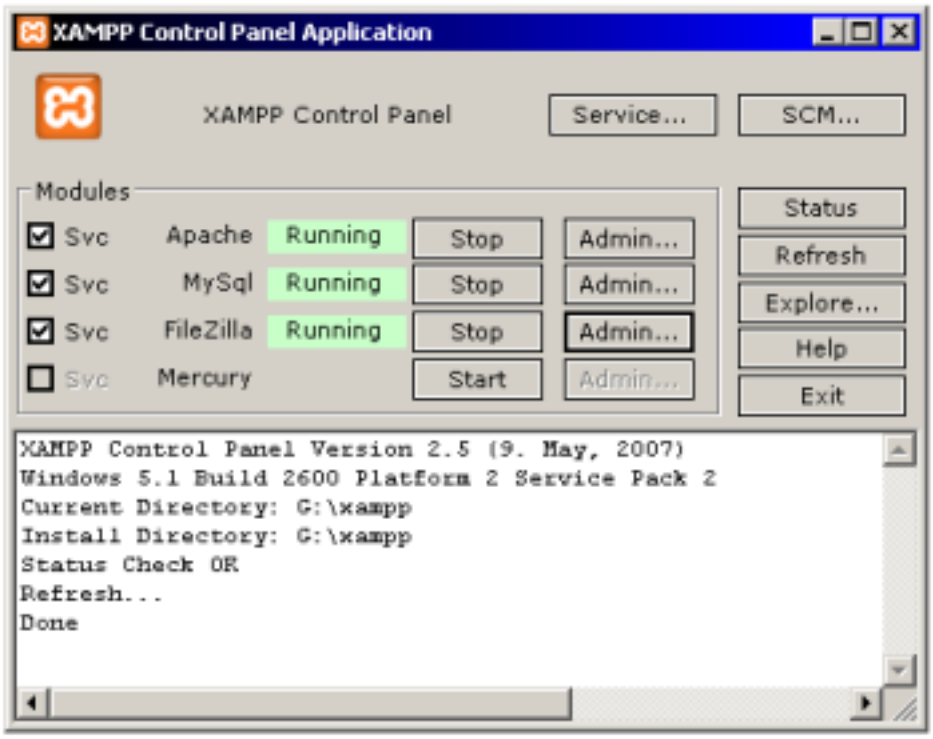
Obr. 3.1: Instalace programu XAMPP - výběr adresáře

nás instalátor žádá (viz. obr.3.1) o zadání cesty k adresáři, kam chceme program nainstalovat.

Zvolíme tedy cestu k adresáři a stiskneme tlačítko Next. V dalším okně se nám zobrazí nabídka některých dalších možností, například vytvoření ikony na Ploše a složky v nabídce Start. Lze také některé části instalovat v systémech Windows XP jako „Service“, tzn. běžící na pozadí. Po kliknutí na tlačítko Install se všechny potřebné soubory nainstalují do zvolené složky, popřípadě se vytvoří ikona na Ploše apod.

Jakmile instalace proběhne, objeví se v pravé dolní části obrazovky Tray Ikona s logem programu (bíle písmeno X v oranžovém čtverci). Kliknutím na ni se spustí „Control Panel“. Tento panel lze spustit i jiným způsobem: v adresáři, kam jsme program nainstalovali, spustíme soubor `xampp-control.exe`.

Zobrazí se okno podobné tomu na obrázku 3.2. Zde je potřeba všechny tři služby pouze spustit. Výsledkem by měl být u každé služby zobrazený zelený obdélník s nápisem „Running“. Není potřeba žádné další nastavení, pouze u služby FileZilla, což je FTP server, je nutné nastavit správně uživatele (viz. kapitola 3.1.3)



Obr. 3.2: Kontrolní panel programu XAMPP

Pokud jsou tedy všechny služby spuštěné, můžeme si ověřit jejich funkčnost. Otevřeme si libovolný internetový prohlížeč a do adresního řádku zadáme : `http://localhost`.

Pokud vše funguje, zobrazí se internetová stránka jako na obrázku 3.3. Zde se dozvíme základní informace, včetně dokumentace k použití systému.

Obr. 3.3: Kontrolní panel programu XAMPP

3.1.2 Příprava MySQL databáze

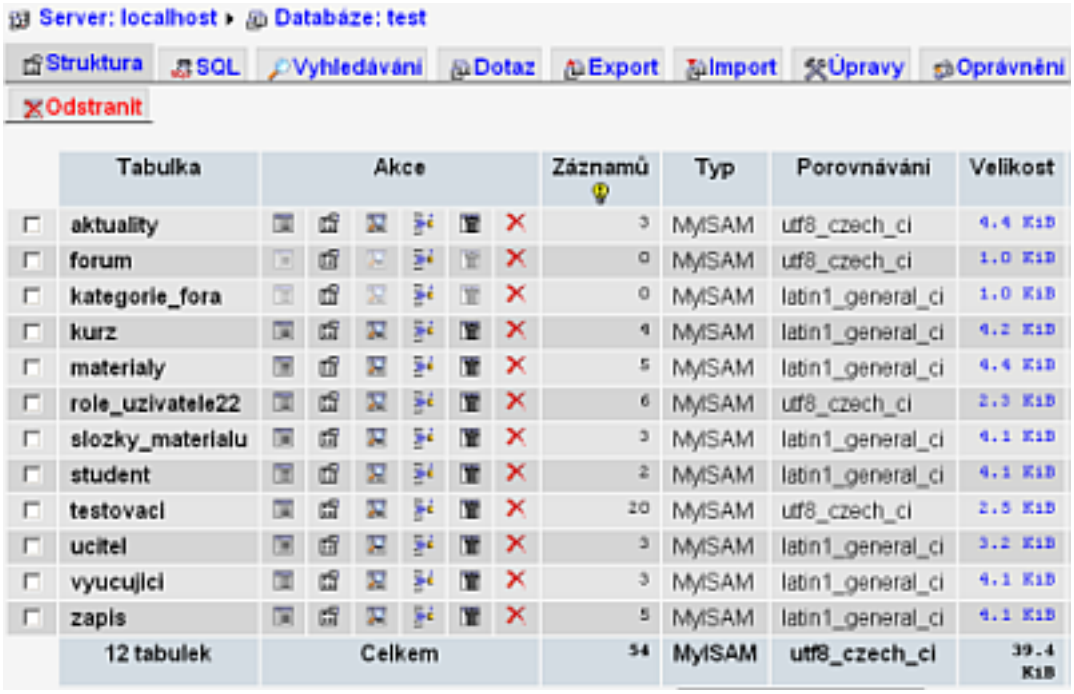
Jak již bylo několikrát uvedeno, celá aplikace e-learningu je postavena na databázi. V této chvíli již databáze funguje, ale není připravená pro práci, neobsahuje potřebné tabulky. Zadáme tedy v internetovém prohlížeči adresu `http://localhost` a v levém (červeném) menu klikneme na odkaz „phpMyAdmin“ v sekci Tools. Zobrazí se nám rozhraní pro správu MySQL databáze. Rozhraní je lokalizované i do češtiny a jeho ovládání je velmi intuitivní. Pro případ potřeby je k dispozici přímo na hlavní stránce i kvalitní dokumentace.

Nyní tedy potřebujeme vytvořit databázi, která bude sloužit pro aplikaci e-learningu. V této hlavní stránce u bodu „Vytvořit novou databázi“ zvolíme její jméno (např. „test“) a zvolíme i tzv. porovnávání, tedy kódovou tabulku pro textové řetězce. Zvolíme kódování „utf8 czech ci“. Je to z toho důvodu, že Java aplikace zasílá textové řetězce jako Unicode a databáze by jinak české znaky špatně interpretovala. Když následně klikneme na tlačítko „vytvořit“, dostaneme se přímo do

nově vytvořené databáze. Tato databáze neobsahuje zatím žádné tabulky, proto je struktura prázdná.

Vytvoření struktury tabulek a naplnění daty se provede jednoduše. Klikneme v horním menu na druhou položku označenou jako „SQL“. Zobrazí se nám víceřádkové vstupní pole pro zadání SQL dotazu. Otevřeme si tedy z přiloženého CD adresář SQL a obsah souboru „sql.txt“ zkopírujeme do tohoto pole na stránce a klikneme na tlačítko „Proveď“.

V databázi by měly být nyní všechny potřebné tabulky i s potřebnými testovacími daty. Pokud teď klikneme na položku „Struktura“, měla by se nám zobrazit podobná stránka jako na obrázku 3.4.



	Tabulka	Akce	Záznamů	Typ	Porovnávání	Velikost
☐	aktuality		3	MyISAM	utf8_czech_ci	4.4 KiB
☐	forum		0	MyISAM	utf8_czech_ci	1.0 KiB
☐	kategorie_fora		0	MyISAM	latin1_general_ci	1.0 KiB
☐	kurz		4	MyISAM	latin1_general_ci	4.2 KiB
☐	materialy		5	MyISAM	latin1_general_ci	4.4 KiB
☐	role_uzivatele22		6	MyISAM	utf8_czech_ci	2.3 KiB
☐	slozky_materialu		3	MyISAM	latin1_general_ci	4.1 KiB
☐	student		2	MyISAM	latin1_general_ci	4.1 KiB
☐	testovaci		20	MyISAM	utf8_czech_ci	2.5 KiB
☐	ucitel		3	MyISAM	latin1_general_ci	3.2 KiB
☐	vyucujici		3	MyISAM	latin1_general_ci	4.1 KiB
☐	zapis		5	MyISAM	latin1_general_ci	4.1 KiB
	12 tabulek	Celkem	54	MyISAM	utf8_czech_ci	39.4 KiB

Obr. 3.4: Struktura databáze v prostředí phpMyAdmin

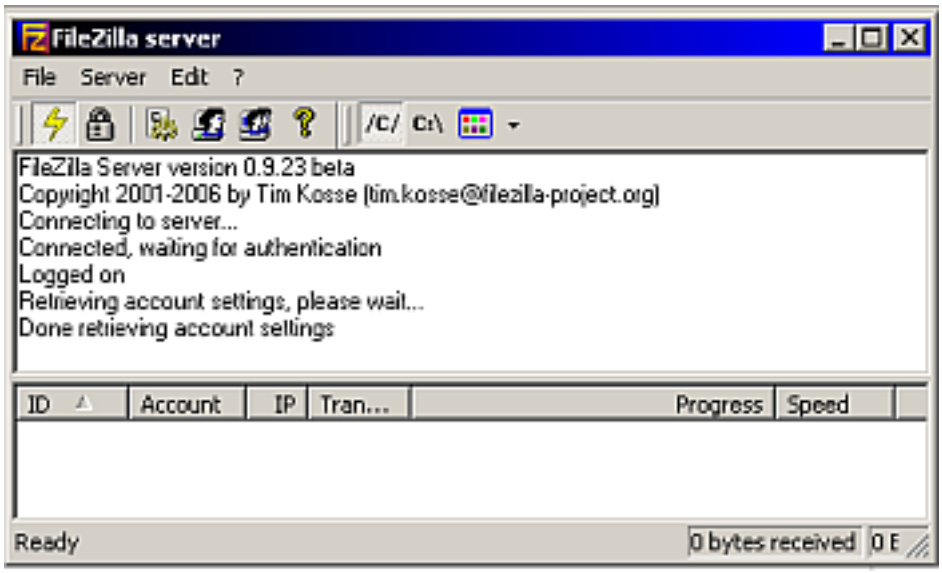
Databázi máme připravenou. Posledním krokem je pouze v záložce „Oprávnění“ vytvořit nového uživatele a nastavit mu všechna práva pouze nad touto databází. Jako jméno i heslo zvolíme „testjava“. Stejně jako název databáze můžeme použít samozřejmě i jiné označení, je ale nutné pak vše správně nastavit na serverové straně aplikace.

3.1.3 Příprava prostoru pro webové stránky a FTP přístupu

Soubory webových stránek serveru XAMPP se nachází v adresáři `G:/XAMPP/htdocs`. Pro potřeby e-learningu zde vytvoříme adresář `VEMS`, do kterého zkopírujeme všechny soubory z přiloženého CD (nachází se ve složce `WWW`). Není třeba žádných dalších

kroků. Nyní když v internetovém prohlížeči zadáme adresu `http://localhost/vems/`, načte se nám webové rozhraní, díky kterému mohou do systému přistupovat studenti. Popisu ovládání tohoto rozhraní se věnuje kapitola 3.4.

Dále je také nutné správně nastavit FTP server pro přístup k těmto souborům, aby klientská aplikace mohla pomocí protokolu FTP nahrávat na stránky přístupné materiály pro studenty. Otevřeme si tedy kontrolní panel programu XAMPP a u položky „FileZilla“ stiskneme tlačítko „Admin…“. Budeme dotázáni na vyplnění údajů o IP adrese serveru a na vyplnění administrátorského hesla. Údaje ponecháme v původním tvaru a potvrdíme přihlášení. Otevře se nám nové okno (viz. obr.3.5), umožňující administraci FTP serveru.

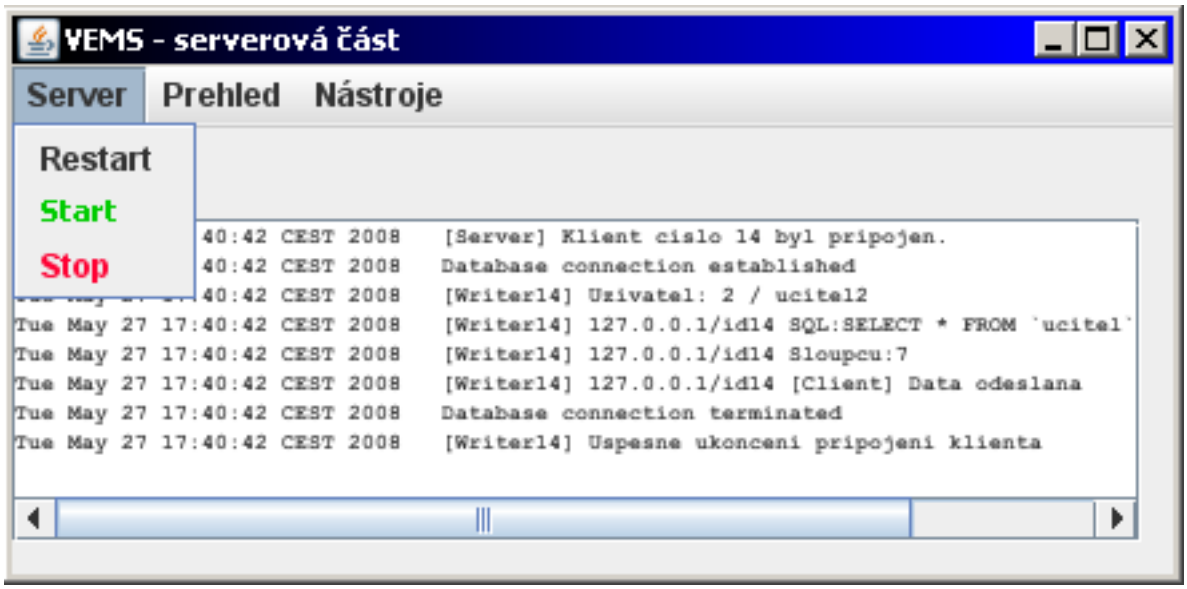


Obr. 3.5: Administrace FTP serveru FileZilla

V hlavním menu zvolíme postupně položku „Edit“ a následně „Users“. V otevřeném okně pak přidáme nového uživatele. Jméno i heslo u něj nastavíme jako „vems“. Přepneme se do záložky „Shared folders“ a zde pro nového uživatele nastavíme všechna práva na námi vytvořený adresář `G:/XAMPP/htdocs/vems`. Nastavení u nového uživatele uložíme a okno zavřeme. Tímto máme vše potřebné připraveno pro správnou funkci celého e-learningového systému.

3.2 Administrace serverové části aplikace

Serverová část aplikace umožňuje klientské části přístup k databázi. Její okno se skládá z jednoduchého menu a z prostoru pro výpis logovacích informací, jak je ostatně zobrazeno na obrázku 3.6.



Obr. 3.6: Grafický vzhled serverové části

Na obrázku je i vidět, že každý logovaný záznam je uvozen časem a datem přidání. V logovacích zprávách lze také najít informace o tom, který klient z jaké IP se kdy připojoval a jaký zasílal SQL dotaz. Pokud klikneme pravým tlačítkem na logy, objeví se kontextové menu, umožňující všechny informace smazat, nebo je lze také uložit do souboru.

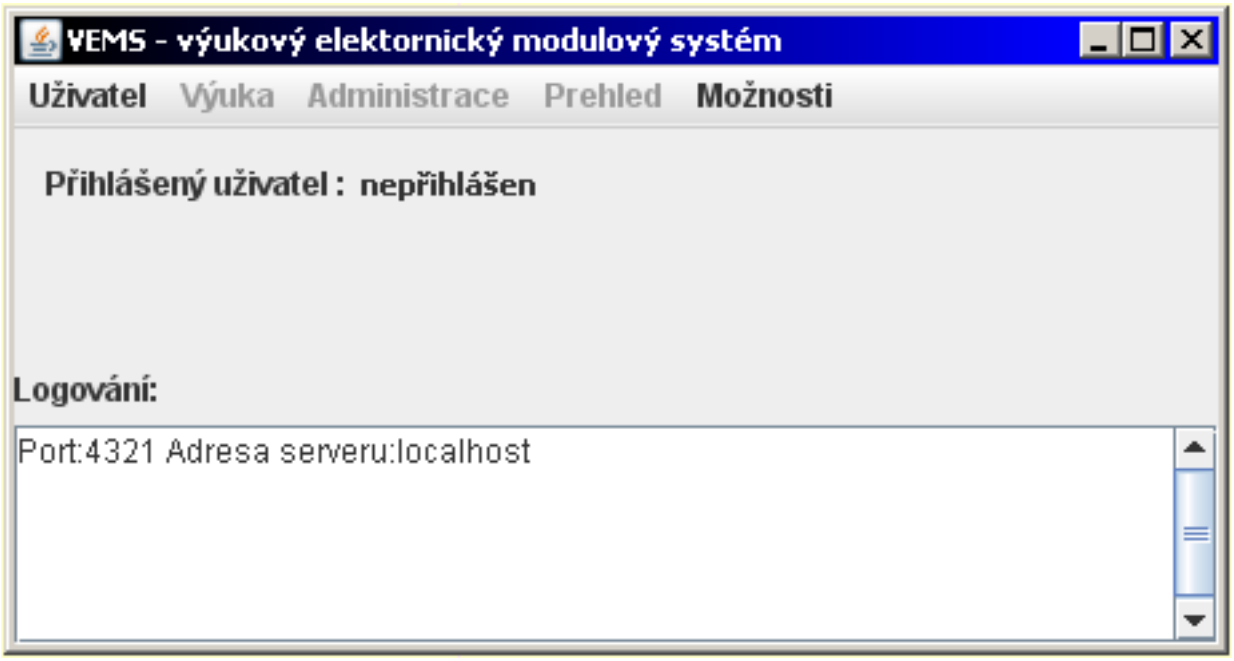
Ovládání vlastního chodu serveu se náchází pod položkou „Server“ v hlavním menu. Je zde umožněn start serveru, ukončení jeho činnosti a v případě změny nastavení i restart. Pokaždé když server začne nebo přestane naslouchat na zadaném portu, vypíše se tato informace i do logovacího okna.

V menu pod položkou „Nástroje - nastavení“ lze nastavovat parametry serveru. Jedná se o nastavení čísla TCP portu, na kterém server čeká na příchozí komunikace. Také se zde určuje, na který SQL server budou přeposílány požadavky. Lze tedy zadat IP adresu SQL serveru a jméno databáze, se kterou se pracuje (implicitně 127.0.0.1 a jméno databáze „test“), v neposlední řadě to mohou být také přihlašovací údaje (defaultně jméno i heslo „testjava“).

3.3 Popis a základní ovládání klientské části

Klientská část aplikace umožňuje obsluhu e-learningového systému a interpretaci získaných dat. Je členěna na jednotlivé moduly, které jsou uživateli přístupné podle toho, jaké má oprávnění. Uživatel typu Admin má v menu přístupné všechny položky, týkající se administrace a položky pro výuku jsou Adminovi nepřístupné. U učitele je tomu přesně naopak.

Grafický vzhled hlavního okna klientské aplikace je zobrazen na obrázku 3.7.



Obr. 3.7: Grafický vzhled klientské části

Po spuštění aplikace není přihlášen žádný uživatel, proto jsou aktivní pouze dvě položky v menu. Pokud zvolíme „Možnosti - nastavení“, otevře se nové okno, ve kterém lze nastavit parametry klienta. Jedná se o zadání IP adresy serveru a číslo portu, na které se aplikace připojuje. Je zde i připraven prostor pro další nastavení, např. nově přidaných modulů.

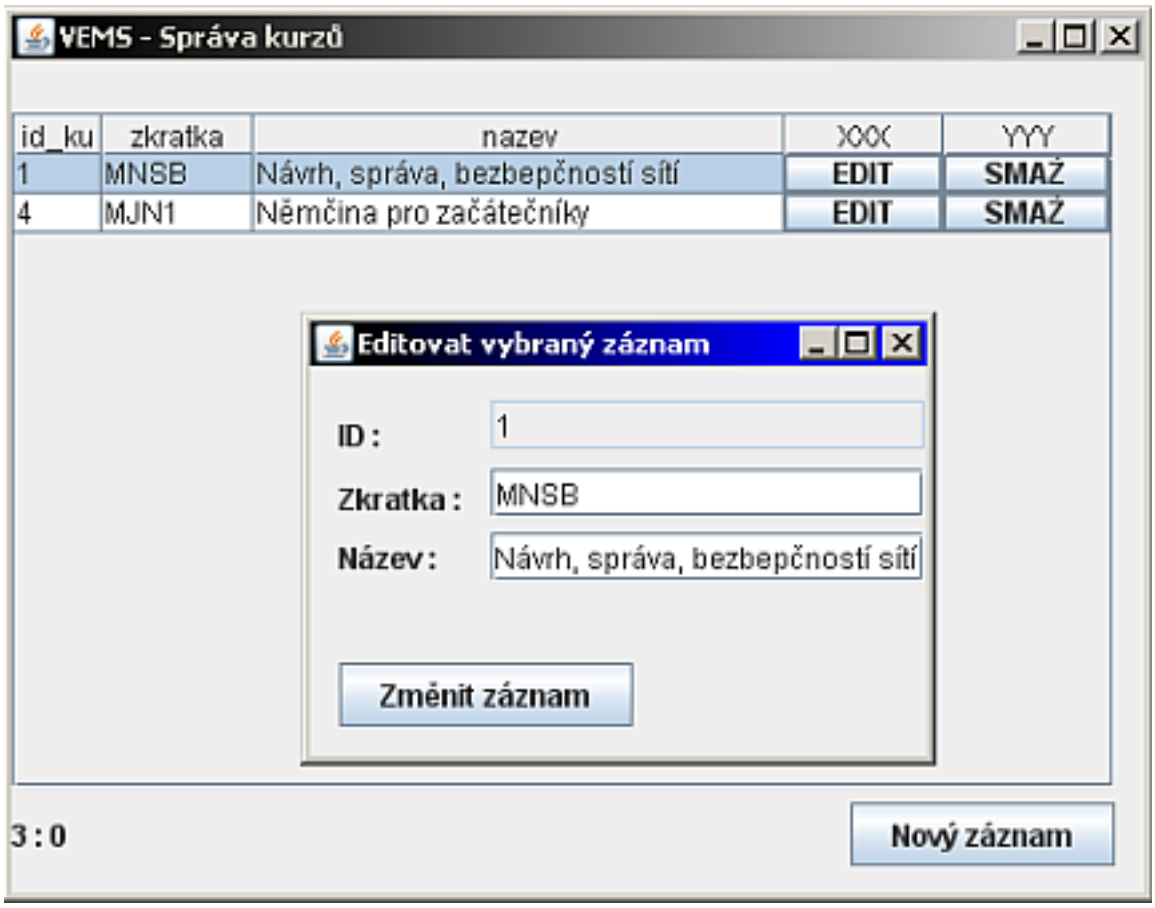
Při přihlašování je po uživateli požadováno zadání jeho ID a přístupového hesla. Aplikace pak následně u serveru ověří správnost těchto údajů a na základě toho buď oznámí špatné přihlášení, nebo uživateli zpřístupní příslušné moduly v menu aplikace. Po úspěšném přihlášení je také v hlavním okně uvedeno jméno a příjmení uživatele.

Jak již bylo uvedeno, při ověřování uživatele na serveru se z databáze získá také informace o jeho úrovni oprávnění. Na tomto základě se uživatelé mohou dělit do dvou skupin. Jedná se o uživatele typu Učitel a typu Admin.

3.3.1 Přihlášení uživatele jako Admin

Nejprve budou popsány možnosti ovládání ze strany uživatele typu Admin. Podle DF diagramu, zobrazeného na obrázku A.1, je vidět, že Admin provádí správu kurzů, učitelů, apod. Pro testování je v databázi již vytvořen uživatel s ID:1 a heslem „admin“. Po úspěšném přihlášení mu je zpřístupněna v menu položka „Administrace“.

Základem je správa kurzů, učitelů a studentů. Tyto správy slouží k práci s těmito tabulkami, v nichž je umožněno přidávat, mazat a editovat záznamy. Pokud uživatel spustí „Správu zápisů“, zobrazí se mu grafické okno jako na obrázku 3.8.



Obr. 3.8: Modul správa kurzů

Tento modul nejprve odešle na server následující SQL dotaz :

```
SELECT id_ku,zkratka,nazev FROM ‘kurz‘ WHERE aktivni = 1
```

Získaná data jsou čtena z příchozího dvojitě vázaného seznamu a postupně zobrazena v tabulce. V každém řádku jsou navíc přidány dva sloupečky obsahující ovládací tlačítka. Podmínka v SQL dotazu `aktivni = 1` slouží k tomu, aby se nezobrazovaly již smazané kurzy. Pro mazání záznamů v těchto tabulkách nelze použít příkaz `DELETE`, protože například v jiné tabulce by se k tomuto kurzu mohl vázat jiný záznam a došlo by k nekonzistenci dat. Proto se při mazání pouze mění atribut „aktivní“ na hodnotu 0 a při znovunačtení tabulky již tento záznam není zobrazen. Na obrázku 3.8 je také vidět, jak vypadá okno pro editaci záznamu, kdy se provedené změny promítnou do databáze pomocí příkazu `UPDATE`. Také je takto řešeno i přidávání nových záznamů. Jen s tím rozdílem, že je použit příkaz `INSERT`. Na

podobném principu fungují také moduly pro správu Učitelů a Studentů, proto je zde nebudeme podrobněji rozebírat.

Dalším z modulů, který je určen pro Admina, je Správa zápisů. Centrálním prvkem je zde tabulka, zobrazující všechny dosavadní zápisy. Jak je ale vidět z obrázku 3.1., tak tabulka ZAPIS funguje jako pomocná tabulka pro vytvoření vazby M:N mezi tabulkami STUDENT a KURZ. Tato tabulka obsahuje pouze dva identifikátory jako ukazatele do těchto dvou tabulek a dva atributy hodnocení. Pokud bychom jí zobrazili pouze v tomto tvaru, získali bychom pouze nepřehledné dvojice čísel, z nichž by nebylo patrné, který student má zapsaný příslušný kurz. Z tohoto důvodu bylo nutné všechny tyto 3 tabulky spojit v jednom SQL dotazu, aby bylo možné pomocí id studenta zjistit i jméno a příjímení studenta a pomocí id kurzu zjistit název kurzu. Takový SQL dotaz vypadá následovně :

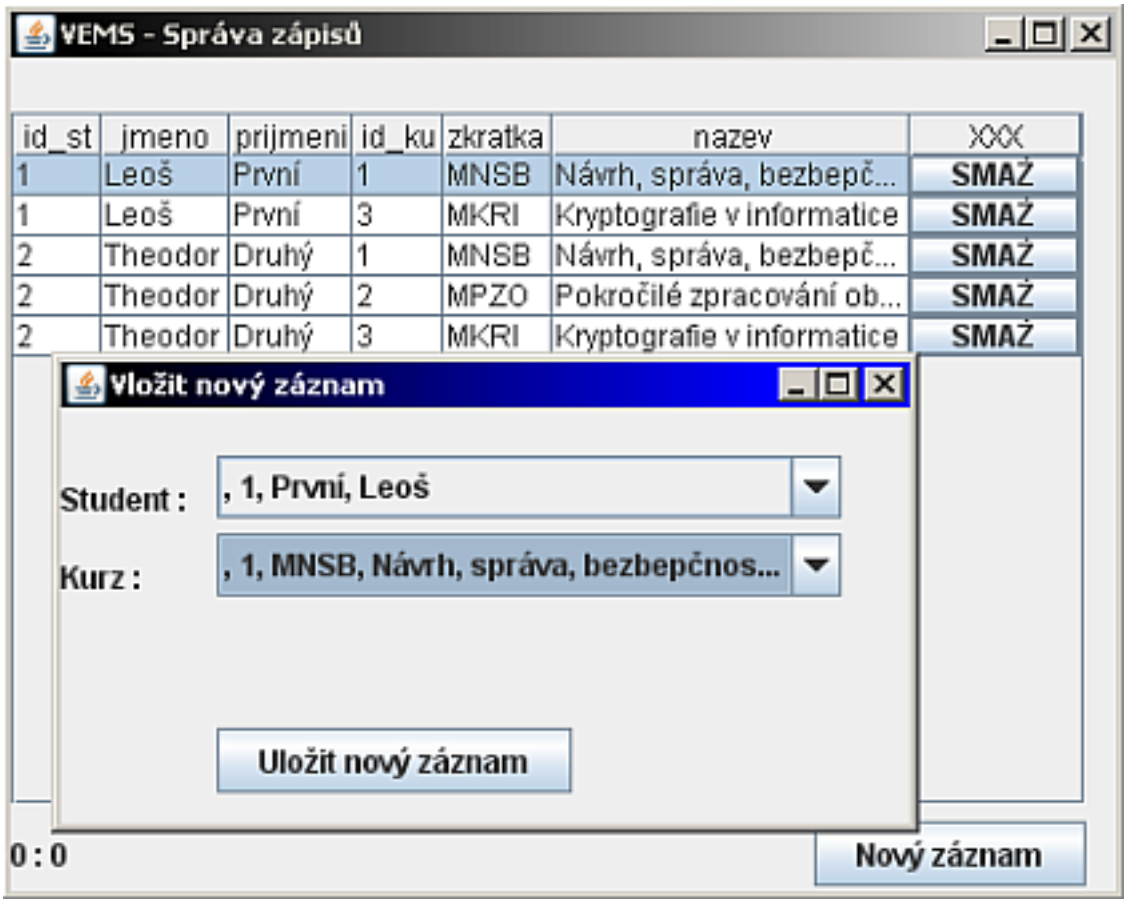
```
SELECT S.id_st, S.jmeno, S.prijmeni, K.id_ku, K.zkratka, K.nazev
FROM ‘student‘ AS S , ‘kurz‘ AS K, ‘zapis‘ AS Z
WHERE (S.id_st = Z.id_st) AND (K.id_ku = Z.id_ku)
```

Na obrázku 3.9 pak tabulka zobrazuje již výsledek takového dotazu. Zároveň je na obrázku zachyceno menší okno určené pro vytváření nového zápisu. Nejdříve se SQL dotazem do dvou „ComboBoxů“ naplní seznamy studentů a kurzů. Admin pomocí nich pak může vybrat studenta, kterého zapíše na daný kurz. Po stisku tlačítka pro uložení nového záznamu se do tabulky ZAPIS vloží nový řádek, obsahující ID příslušného studenta a příslušného kurzu. Oba atributy hodnocení jsou nastaveny na nulové hodnoty. Podobně funguje modul na přiřazení učitelů k jednotlivým kurzům, jen s tím rozdílem, že se zde řeší vazba M:N mezi tabulkami UCITEL a KURZ.

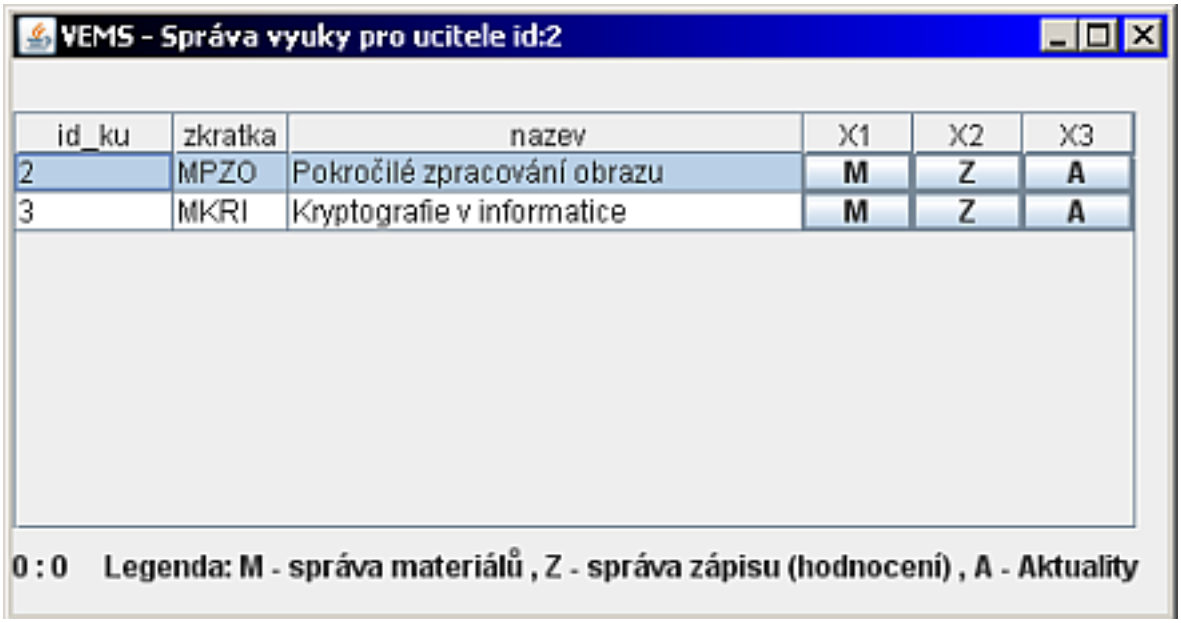
3.3.2 Přihlášení uživatele jako Učitel

Pokud má přihlášený uživatel úroveň oprávnění Učitel, je mu v menu zpřístupněna položka „Výuka“. Položka „Administrace“ zůstane zašedlá. Pro testovací účely exituje s touto úrovní oprávnění uživatel ID:2 a jeho heslo je „ucitel2“. Při spuštění modulu Výuka se uživateli zobrazí seznam všech předmětů, u kterých je uveden jako vyučující. Na řádku u každého z předmětů má 3 tlačítka rozlišené písmenem, jak je naznačeno na obrázku 3.10

Tlačítko A umožňuje spravovat aktuality pro daný předmět. Učitel může přidávat a odebírat již nepotřebné aktuální informace. V rámci akutalit lze zadávat i jejich časovou platnost a případně i internetový odkaz. Tlačítko Z zpřístupňuje Zápis, resp. umožňuje hodnocení žáků zapsaných na zvolený předmět a tlačítko M nabízí přístup k materiálům. Po jeho stisku se pro daný předmět zobrazí přehled složek materiálů. Je možné složky přidávat i mazat. Na řádku u každé složky je

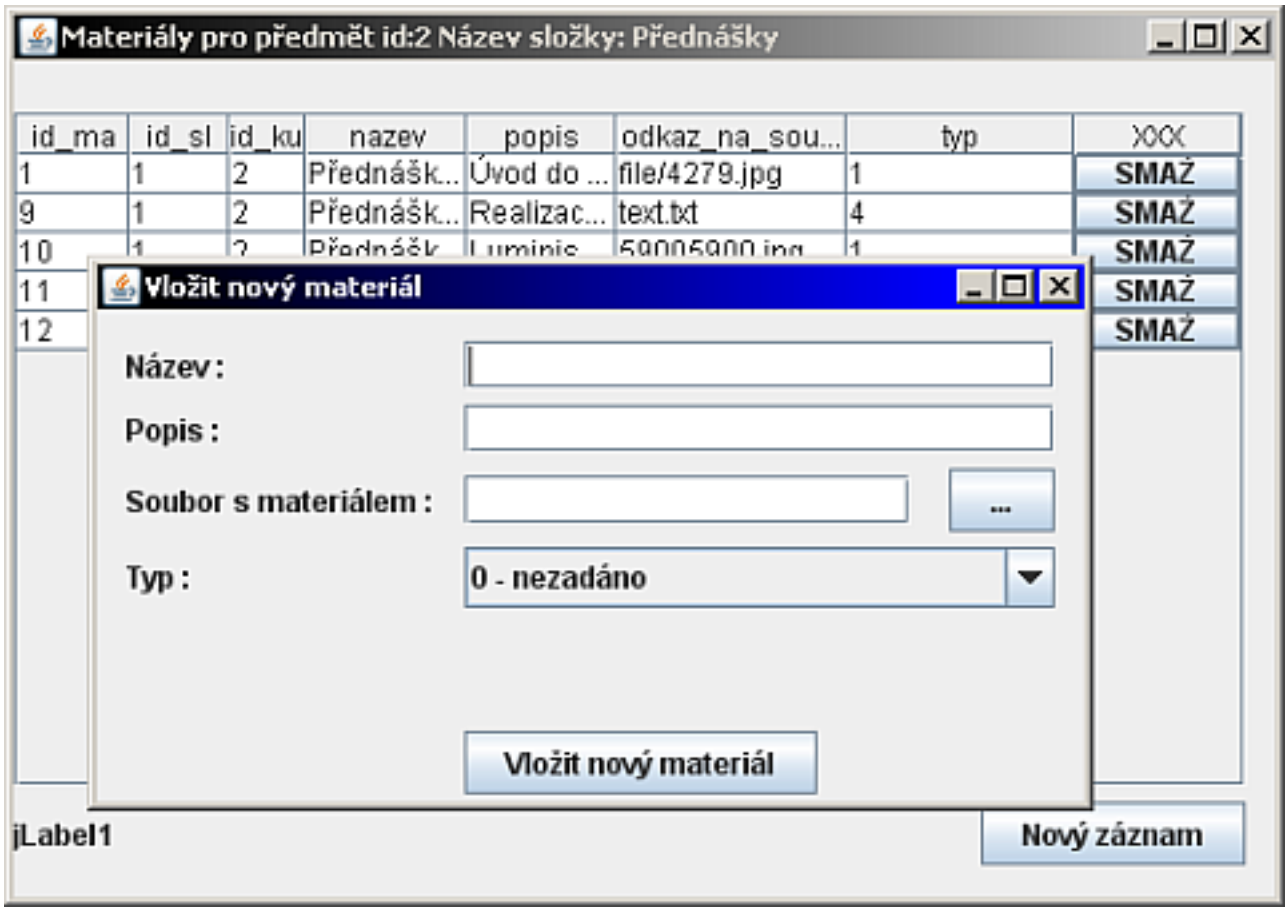


Obr. 3.9: Modul správa zápisů



Obr. 3.10: Modul správy výuky pro učitele

tlačítko „Materiály“. Po jeho stisku se zobrazí přehled všech materiálů obsažených v dané složce, kde lze přidávat a odebírat záznamy. Pokud chce uživatel přidat nový materiál do vybrané složky, vyplní pouze název tohoto materiálu, jeho stručný popis o maximální délce 255 znaků a jeho typ (např. zda-li se jedná o soubor PDF, nebo o obrázek apod.) Následně stiskne tlačítko s třemi tečkami, čímž se mu otevře dialog pro výběr souboru na disku. Soubor potvrdí a při stisku tlačítka na vložení nové hodnoty dojde současně k přenosu vybraného souboru pomocí protokolu FTP. Do databáze se pak do atributu „odkaz na soubor“ uloží pouze jméno souboru, který je pak následně přístupný studentům přes webové rozhraní. Grafická okna správy materiálu jsou zobrazena na obrázku 3.11.



Obr. 3.11: Modul správa materiálů

3.3.3 Webové rozhraní pro přístup studentů do systému

V rámci této diplomové práce byly vytvořeny navíc i webové stránky, které pro práci s daty využívají jazyka PHP. Účelem těchto stránek je umožnit studentům přístup

ke zveřejněným materiálům pro zapsané kurzy. Stejně tak mohou číst aktuality a získat informace o svém hodnocení.

K vytvoření tohoto webového rozhraní bylo přistoupeno z toho důvodu, že se předpokládá, že by studenti neměli vždy přístup k Java aplikaci. Bylo by tedy pro studenty vhodnější, aby potřebné informace mohli získat z internetových stránek.

Byla navrhnuta jednoduchá a přehledná grafická podoba stránek. Na obrázku 3.12 je zobrazena úvodní obrazovka, umožňující přihlášení. Pro testování bylo použito následující přihlášení: ID:1 , heslo je „prvni“.

Obr. 3.12: Grafická podoba webových stránek

Po přihlášení je studentovi zobrazena tabulka se všemi předměty, do kterých je přihlášen. Vedle každého předmětu jsou na řádku další tři odkazy. Student si přes tyto odkazy může zjistit aktuální informace pro daný předmět, svoje hodnocení a samozřejmě si také může procházet složky materiálů pro zvolený předmět a zveřejněné materiály si může otevřít nebo stáhnout na disk. Vlastní ovládání této aplikace není složité a navede uživatele samo, a proto je připravena ihned k použití.

4 ZÁVĚR

E-learning se v posledních letech stává čím dál více oblíbenou formou vzdělávání, a to nejen ve firmách, kde umožňuje zaměstnancům zvyšovat jejich jazykové nebo odborné vzdělání, ale i na školách a univerzitách, kde se stává nenahraditelným pomocníkem pro distanční formu výuky a v prezenčích formách doplňuje klasickou výuku elektronickými oporami.

Cílem této diplomové práce bylo popsat obecné principy e-learningu a prostudovat jeho hlavní výhody a nevýhody. Na základě těchto znalostí byl rozebrán stav současných e-learningových systémů na vybraných univerzitách, včetně informací o platformách, na kterých jsou instalovány.V rámci praktické části této práce byl navrhnut základní koncept e-learningové aplikace, který umožňuje správu modulů a základní administraci e-learningového systému. Tento koncept je následně realizován v programovacím jazyku JAVA. Během realizace byl řešen problém ohledně formy vzájemné komunikace obou částí aplikace, včetně vhodné reprezentace výsledku SQL dotazu a přenosu tohoto výsledku přes vytvořené TCP spojení. Dále byl také popsán použitý pseudoprotokol, umožňující klientovi se přihlásit, sdělit serveru svůj požadavek a v dohodnutém tvaru následně přijmout výsledná data. V závěrečné části je popsán vytvořený systém z hlediska uživatelského ovládání a administrace. Tato část je koncipována tak, aby mohla současně sloužit jako manuál celé aplikace i jako návod na její instalaci. Jsou zde rozebrány možnosti práce jednotlivých typů uživatelů. Pro přístup studentů do systému bylo navíc vyvinuto webové rozhraní v jazyce PHP, které zpřístupňuje uživatelům potřebné informace a materiály ke studiu.

Výsledná aplikace je vhodná pro menší vzdělávací organizace, které potřebují jednoduchým způsobem spravovat své kurzy a posluchačům těchto kurzů zpřístupňovat studijní materiály a důležité informace. Celý systém lze podle potřeb uživatele měnit a přidávat nové moduly, které by umožňovaly aplikaci rozšířit o nové funkce a služby. Při tvorbě těchto nových modulů již není potřeba ošetřovat průběh komunikace. Programátor se věnuje pouze práci s daty a jejich reprezentaci pro uživatele. Aplikace samotná je nenáročná na výpočetní prostředky, při testování fungovala bez problémů i na notebookovi Toshiba Libretto 100CT, který disponuje procesorem pouze o frekvenci 200MHz a operační pamětí 64MB. Klientská část vyžaduje pouze operační systém umožňující běh JVM (Java Virtual Machine). Pro správný běh serverové části aplikace je navíc potřebná jen správně nastavená SQL databáze a webový server.

Tato diplomová práce byla vysázena v systému LaTeXza použití šablony vytvořené na Ústavu telekomunikací. Všechny potřebné soubory a programy potřebné pro spuštění aplikace jsou k dispozici na přiloženém CD a mohou být dále použity pro další vývoj.

LITERATURA

[1] BAREŠOVÁ, Andrea. *E-learning ve vzdělávání dospělých*. První vydání. Praha : VOX, 2003. 174 s. ISBN 80-86324-27-3

[2] DARWIN, Ian F. *Java : kuchařka programátora*. Vyd. 1. Brno : Computer Press, 2006. 798 s. ISBN 80-251-0944-5

[3] HEROUT, Pavel. *Java : grafické uživatelské prostředí a čeština*. Dotisk prvního vyd. České Budějovice : Kopp, 2006. 316 s. ISBN 80-7232-237-0

[4] HEROUT, Pavel. *Učebnice jazyka Java*. 3., rozšíř. vyd. České Budějovice : Kopp, 2007. 381 s. ISBN 80-7232-115-3

[5] SPELL, Brett. *Java : programujeme profesionálně*. Vyd. 1. Praha : Computer Press, 2002. 1022 s. ISBN 80-7226-667-5

[6] *Wikipedie - Otevřená encyklopedie* [online]. poslední aktualizace 8. 10. 2007. Dostupné z: <http://www.wikipedia.org> [cit. 10. 5. 2008].

[7] *Java portál - konference* [online]. poslední aktualizace 8. 1. 2008. Dostupné z: <http://www.java.cz> [cit. 10. 3. 2008].

[8] *The Source for Java Developers* [online]. poslední aktualizace 8. 1. 2008. Dostupné z: <http://java.sun.com> [cit. 11. 4. 2008].

[9] *Builder.cz: diskuzní fórum Java* [online]. 2001, poslední aktualizace 21. 12. 2007 [cit. 17. 4. 2008]. Dostupné z: <http://forum.builder.cz/list.php?14>.

[10] *Owebu.cz: seriál Java programování* [online]. poslední aktualizace 8. 4. 2008. Dostupné z: <http://www.owebu.cz> [cit. 21. 4. 2008].

[11] *Java Library FTP - Documentation* [online]. poslední aktualizace 8. 5. 2008. Dostupné z: <http://www.enterprisedt.com/products/edtftpj/> [cit. 21. 5. 2008].

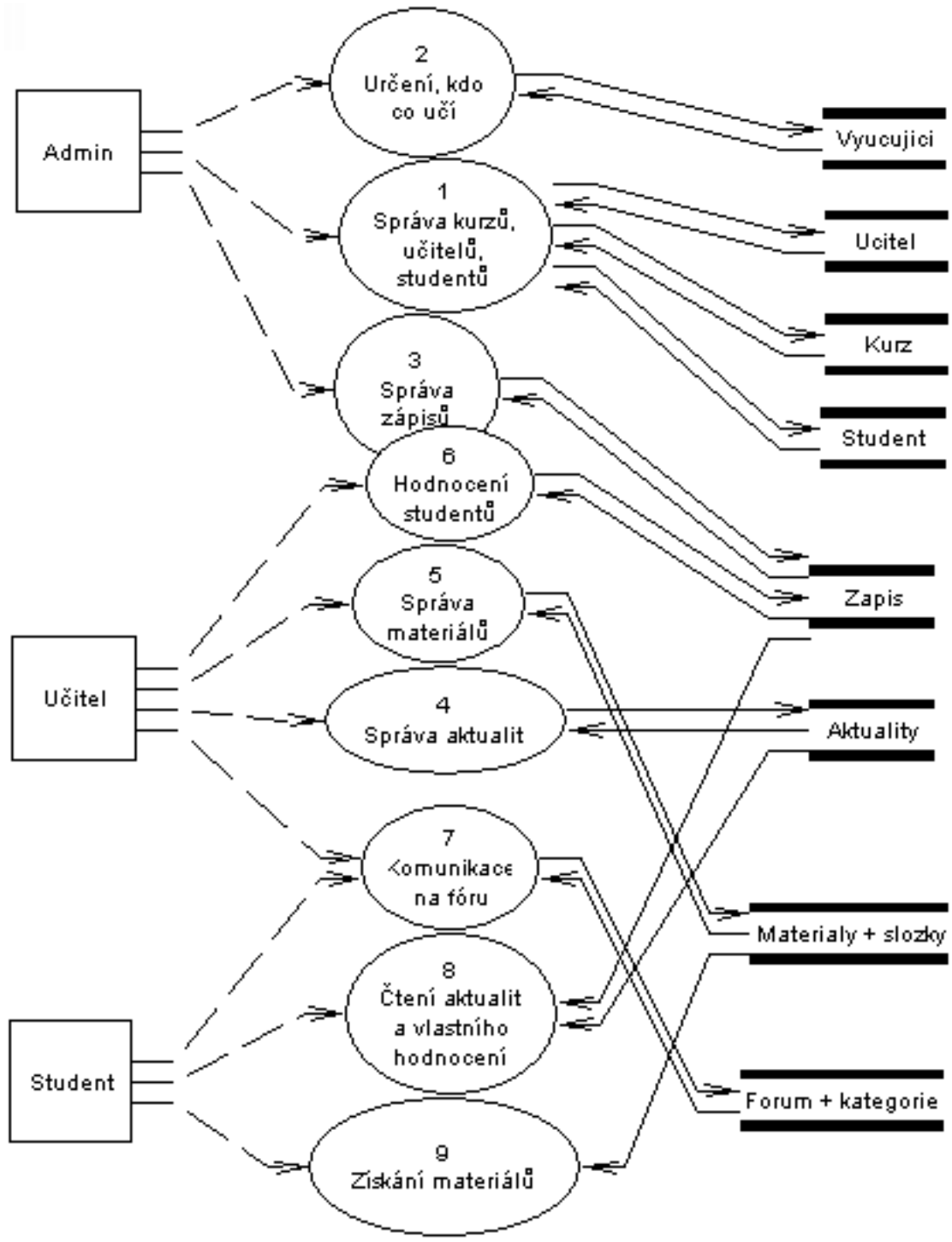
[12] *MoodleDocs* [online]. poslední aktualizace 8. 1. 2008. Dostupné z: <http://docs.moodle.org/cs/> [cit. 21. 1. 2008].

[13] *Články k IS MU* [online]. poslední aktualizace 12. 4. 2008. Dostupné z: <http://is.muni.cz/clanky/> [cit. 21. 11. 2007].

[14] *Informační systémy MZLU v Brně* [online]. poslední aktualizace 12. 10. 2007. Dostupné z: <http://www.mzlu.cz/is/> [cit. 23. 11. 2007].

A OBRAZOVÁ PŘÍLOHA

A.1 DF diagram aplikace e-learningu



Obr. A.1: DF diagram aplikace e-learningu

B PŘÍLOHA ZDROJOVÝCH KÓDŮ

B.1 Implementace jedné položky dvojitě vázaného seznamu

```
package vems;

import java.io.Serializable;

public class ConcreteItem implements Item,Serializable{

  private String data; //data
private Item prev;   //predchozi
private Item next;   //dalsi

    /** Creates a new instance of ConcreteItem */
    public ConcreteItem() { //konstruktor
    }


public String getData() {
return data;
}

public Item getNext() {
return next;
}

public Item getPrev() {
return prev;
}

public boolean isFisrt() {
return prev == null;
}

public boolean isLast() {
return next == null;
```

```
}

public void setData(String data) {
this.data = data;
}

public void setNext(Item item) {
this.next = item;
}

public void setPrev(Item item) {
this.prev = item;
}

}
```

B.2 Implementace dvojitě vázaného seznamu

```
package vems;

import java.io.Serializable;

public class ConcreteDoubleLinkedList implements DoubleLinkedList,Serializable {
    // objekt obsahuje 3 atributy, odkaz na First,Act a Last

private Item last;
private Item first;
private Item act;

public ConcreteDoubleLinkedList() { //konstruktor, prazdny seznam
this.act=null;
this.first=null;
this.last=null;
}

public synchronized void addAfterAct(Item item) {
if (this.isEmpty()) {
this.act = item;
this.last = item;
this.first = item;
} else {
item.setPrev(act);
item.setNext(act.getNext());
if (act.getNext() == null) {
act.setNext(item);
last = item;
} else {
act.getNext().setPrev(item);
act.setNext(item);
}
}
}

public synchronized void addBeforeAct(Item item) {
if (this.isEmpty()) {  //pokud je seznam prazdny
```

```
this.act = item;
this.last = item;
this.first = item;
} else {
item.setNext(act);
item.setPrev(act.getPrev());
if (act.getPrev() == null) { //pokud act ukazuje na prvni prvek
act.setPrev(item);
first = item;
} else {
act.getPrev().setNext(item);
act.setPrev(item);
}
}
}

public synchronized void addFirst(Item item) {
if (this.isEmpty()) {
this.act = item;
this.last = item;
this.first = item;

} else {
first.setPrev(item);
item.setNext(first);
item.setPrev(null);
first=item;
}
}

public synchronized void addLast(Item item) {
if (this.isEmpty()) {
this.act = item;
this.last = item;
this.first = item;

} else {
last.setNext(item);
item.setPrev(last);
```

```
item.setNext(null);
last=item;
}
}

public synchronized Item getAct() {
return this.act;
}

public synchronized Item getFirst() {
return this.first;
}

public synchronized Item getLast() {
return this.last;
}

public synchronized boolean isEmpty() {
return first == null;
}

public synchronized void nextAct() {
this.act = this.act.getNext();
}

public synchronized void prevAct() {
this.act = this.act.getPrev();
}

public synchronized void removeAct() {
if (this.act != null) {
if (this.act.getPrev() == null) {
if (this.act.getNext() == null) {
//jedinny prvek
this.act = null;
this.first = null;
this.last = null;
} else {
//vice prvku, Act ukazuje na prvni
```

```
this.act.getNext().setPrev(null);
this.act = this.act.getNext();
this.first = this.act;
}

} else {
if (this.act.getNext() == null) {
//vice prvku, Act ukazuje na posledni
this.act.getPrev().setNext(null);
this.act = this.act.getPrev();
this.last = this.act;
} else {
//vice prvku First,Last a Act jsou nezavisle
this.act.getPrev().setNext(this.act.getNext());
this.act.getNext().setPrev(this.act.getPrev());
this.act = this.act.getPrev();
}
}
}
}

public synchronized void removeFirst() {
if (this.first != null){
if (this.first.getNext() == null){
this.act = null;
this.first = null;
this.last = null;
} else {
if (this.act == this.first) {
this.act = this.first.getNext();
}
this.first.getNext().setPrev(null);
this.first = this.first.getNext();
}
}
}

public synchronized void removeLast() {
if (this.last != null){
```

```
if (this.last.getPrev() == null){
this.act = null;
this.first = null;
this.last = null;
} else {
if (this.act == this.last) {
this.act = this.last.getPrev();
}
this.last.getPrev().setNext(null);
this.last = this.last.getPrev();
}
}
}

public synchronized void setActToFirst() {
this.act = this.first;
}

public synchronized void setActToLast() {
this.act = this.last;
}

}
```

C ELEKTRONICKÁ PŘÍLOHA

C.1 Obsah přiloženého CD

Součástí této diplomové práce je i přiložené CD, které obsahuje následující položky :

- Text této práce ve formátu PDF i s přiloženými zdrojovými soubory systému LaTeX.
- Spustitelné soubory obou částí aplikace (klientské i serverové) s příponou JAR.
- Zdrojové kódy aplikace v jazyku Java včetně přiložených knihoven.
- Textový soubor obsahující SQL dotazy pro vytvoření struktury databáze podle obrázku 2.5.
- PHP soubory webového rozhranní pro přístup studentů do systému.
- Instalační soubor programu XAMPP, který obsahuje webový server s podporou PHP i FTP přenosů. Jeho součástí je MySQL databáze.
- Instalaci Java Runtime Environment, pro spuštění Java aplikace.