eKit - ASPEKTY OVLIVŇUJÍCÍ VÝVOJ INTRANET APLIKACE

Libor Gála, Michal Harsa, David Krch

Katedra informačních technologií, Vysoká škola ekonomická, nám. W.Churchilla 4, 130 67 Praha 3, ČR, {gala|harsa|krch}@vse.cz

Abstrakt
Many organizations are in the process of introducing new Internet/Intranet based solutions. This paper discusses some aspects that affect a developing Intranet application. We tried to mention some of them on the basis of our experience with the real university Intranet application "eKIT". We focused especially on issues of development & maintenance, performance and security of Intranet applications and balance among these crucial aspects.

1. Projekt eKit, základní charakteristiky

Projekt eKit vznikl na Katedře informačních technologií VŠE Praha na základě potřeby snížit čas strávený učiteli i studenty administrativou, která je spojená s organizací a průběhem vyučovaných kurzů, a zároveň se záměrem ověřit si, zda i v prostředí vysokých škol je vhodné naplňovat heslo „otevřeno 24 hodin denně". Výsledkem řešení projektu je systém označovaný jako eKit, který je provozován v doméně *ekit.vse.cz* a který je nyní užíván dvěmi tisíci studentů v cca 50 kurzech.

Systém eKit charakterizujeme:

- *v oblasti technologické* jako systém s architekturou klient/server s „tenkým klientem" (v současné době je reprezentován browserem a nativními protokoly http a https), s objektovou architekturou COM, s datovým serverem Oracle 8.0.5 a servery WWW (Worl-Wide Web), SMTP (Simple Mail Tranfer Protocol), certifikačních služeb společnosti Microsoft. Celý systém je nyní provozován na jednom počítači HP NetServer LH4 (Pentium III Xeon 550MHz, 2MB L2 Cache, 512MB RAM, Ultra2 SCSI 36GB HDD).
- *v oblasti datové* jako systém s vysokým podílem transakčních dat s pravidelnou (pololetní) úplnou změnou cca 40% datové základny a správou nestrukturovaných dokumentů, které jsou uloženy v různých datových formátech.
- *v oblasti uživatelů*, jako systém s vysokou „fluktuací", tj. častou změnou uživatelů či jejich přístupových práv.

Z hlediska aplikačních funkcí systém eKit v současné podobě zahrnuje:

- *jádro systému*, které zajišťuje centralizovanou správu uživatelů a rolí, jejich přístupových práv a zároveň poskytuje společné služby (převážně datové) dalším modulům. Jádro systému dále zajišťuje:
  - pro jednotlivé výukové kurzy definici časové osy kurzu, tj. možnost definovat přístup k dalším modulům a jejich funkcím dle definované časové posloupnosti včetně definice věcné návaznosti a
  - pro jednotlivé uživatele z řad studentů udržování historie jejich dílčích i celkových výsledků pro kurzy, které absolvovali.
- *modul „Dokumenty"*, který slouží k vystavování a zpřístupnění dokumentace kurzu (v různých datových formátech) např. sylaby, plné texty přednášek, apod. definovaným uživatelům s možností interního či externího uložení.

- *modul „Zkouška“*, který řeší celou administrativu spojenou s ukončením kurzu, od vypsání termínů učitelem, přes přihlašování se studenty, zápis hodnocení zkoušky, až po generování výstupů pro studijní systém školy.
- *modul „Práce“*, který je zaměřen na podporu administrativy spojené se zadáváním prací studentům, elektronickým odevzdáváním výsledků, včetně možnosti realizace úkolu v pracovním týmu, tj. organizaci týmů.
- *Modul „Anketa“*, který umožňuje realizovat libovolnou anonymní anketu a to v neanonymním prostředí, tj. podporuje tvorbu anket, odpovídání na otázky, zpracování výsledků, přičemž je možno na otázky odpovídat výčtem, hodnotou či volným textem.

Systém svojí otevřeností, která je dána jak zvolenou technologií (s dodržením standardů a doporučení), tak i aplikační architekturou, umožňuje další rozvoj jak v oblasti nových aplikačních modulů (elektronické diskuse, on-line testy, apod.), tak i v oblasti nových trendů v komunikaci (WAP, SMS, apod.).

Během vývoje systému a jeho zatímního provozu vznikla v pracovním týmu řada doporučení, která jsou, dle našeho názoru, při implementaci obdobného systému vhodná respektování. V další části příspěvku se zaměříme převážně na ta doporučení, která směřují do oblasti zajištění bezpečnosti a výkonu jakožto klíčových vlastností systému (samozřejmě vedle požadované funkcionality) a do oblastí standardů a procedur umožňujících „bezproblémový“ rozvoj systému. Podotýkáme, že systém eKit je v neustálém vývoji a stávající verze některá doporučení prozatím nedodržuje.

2. Standardy a procedury – předpoklad úspěšného vývoje a dalšího rozvoje

Za předpoklad pro úspěšný vývoj systému a jeho další rozvoj považujeme především užití metodik vývoje a provozu, definovaných pravidel, procedur a užívání standardů.

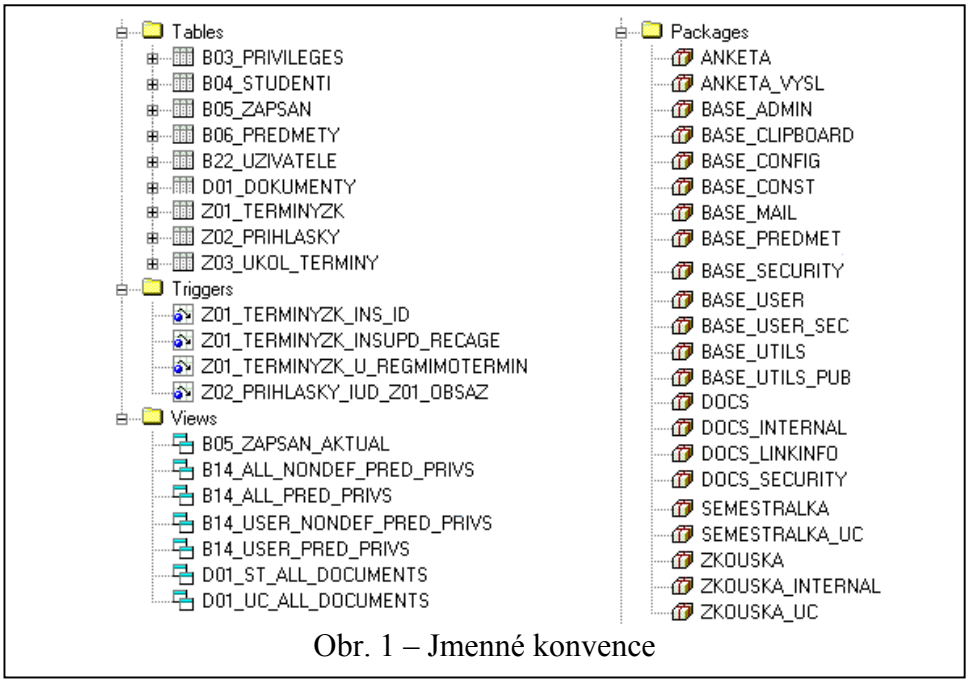
V procesu vývoje systému jsme užívali standardních metod a technik datové a funkční analýzy tak, jak je to obvyklé při vývoji jakéhokoliv systému. Jednalo se zejména o užití ERD (Entity Relationship Diagram) a DFD (Data Flow Diagram; diagram datových toků) modelů [1]. Za stěžejní považujeme oblasti:

- jmenných konvencí,
- strukturování programových celků a
- návrh celkové architektury.

Z důvodu specifika pracovního týmu, které lze charakterizovat vysokou obměnou pracovníků (v týmu jsou zastoupeni i studenti), jsme byli nuceni přijmout některá opatření, která umožní bezproblémovou komunikaci mezi pracovníky týmu vývoje a zároveň rychlé zapracování pracovníků nových. Jedná se především o *jmenné konvence*, které ve výsledku hráli i významnou roli při udržení konzistence celého řešení.

Příkladem jmenných konvencí je např. úroveň databáze (obr. 1), kde názvy jednotlivých knihoven (Package) začínají vždy názvem modulu, ke kterému se vztahují. Tabulky nemají ve svém názvu na začátku název modulu, ale pouze počáteční písmeno modulu a dvojmístné pořadové číslo, které spolu s počátečním písmenem tvoří jednoznačný kód tabulky (za ním následuje „popisný název“ tabulky). Tento kód tabulky je pak použit v názvech všech objektů, které se k dané tabulce vztahují (indexy, triggery, apod.).

Z důvodu úzké specializace jednotlivých pracovníků, která se projevuje detailní znalostí (avšak pouze úzké části celkové problematiky), jsme byli nuceni přistoupit k *rozdělení vývoje*



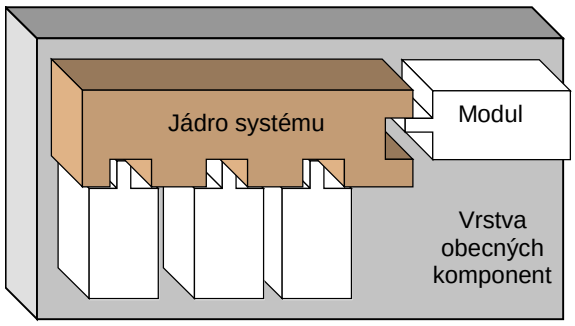
Obr. 1 – Jmenné konvence

jednotlivých modulů dle prostředí. Vývoj modulu je rozdělen na:

- vývoj prezentační vrstvy s užitím jazyků HTML (HyperText Markup Language) a CSS (Cascading StyleSheet),
- vývoj funkcí prováděných na straně klienta s užitím jazyka JavaScript,
- vývoj rozhraní mezi aplikační a prezentační vrstvou s užitím technologie ASP – Active Server Pages a jazyka VBScript (Visual Basic Script),
- vývoj aplikační logiky s užitím jazyka PL/SQL v prostředí Oracle Server,
- vývoj datové logiky v prostředí Oracle Server.

Pro zajištění konzistence řešení jsme užili definovaných jmenných konvencí, pravidel minimální dokumentace kódu a volání jednotlivých „zlomků“ kódů, přičemž role centrálního prvku, který udržuje logiku modulu (či dílčí funkce), byla přiřazena ASP dokumentům. Tento přístup, ačkoliv jeho potřeba byla vyvolána vlastnostmi vývojového týmu, s sebou přináší i relativně snadnou cestu změny vyjádření prezentační vrstvy (např. při přechodu k jazyku XML – Extended Markup Language).

V souvislosti s vývojem prezentační vrstvy považujeme za významné zmínit se o grafickém řešení. V řešení jsme však použili vlastních grafických prvků - ikon (sloužících jako navigační a ovládací prvky), což mělo za následek, že uživatelé (především z řad učitelů) potřebovali daleko více času k tomu, aby systém začali efektivně užívat. Doporučujeme proto užívat s*tandardních a obecně platných ikon a symbolů* nejlépe takových, které kopírují operační či kancelářský systém.



Obr. 2: Architektura zásuvných modulů

Jednou z podmínek řešení projektu bylo zajištění snadného rozvoje systému tak, že ho bude možno rozšířit o další aplikační moduly. Pro zajištění tohoto požadavku jsme přijali *princip centrálního prvku*, tzv. jádra systému, do něhož jsou jednotlivé aplikační moduly zasouvány (obr. 2). Tento přístup umožňuje řešitelům samostatný rozvoj aplikačních modulů a zároveň nutí řešitele vyvíjet moduly nezávisle jeden na druhém. Vazbu mezi moduly (jedná se o vazbu věcnou a vazbu vyjadřující časovou souslednost užití modulů) je možno realizovat prostřednictvím jádra systému a to prostřednictvím datové logiky. Při řešení takto uspořádaného systému bylo velmi významné udržet rozsah jádra pouze na úrovni, která mu byla přiřazena. Všechny další, především podpůrné funkce, přestože byly společné více modulům, byly umístěny do samostatné vrstvy, která může být volána libovolným modulem. Tato vrstva je pak charakteristická nejen tím, že obsahuje znovupoužitelné obecné komponenty, ale i tím, že je velmi podrobně dokumentována.

Další pravidla pak byla definována pro oblast zajištění výkonu a bezpečnosti systému a jsou uvedena v dalších částech příspěvku.

3. Výkon systému – předpoklad spokojenosti uživatelů

Výkon systému chápeme jako jednu z klíčových vlastností, která může ovlivnit spokojenost koncových uživatelů a proto jsme na tuto oblast kladli velký důraz.

Přestože eKit svým pojetím odpovídá definicím, které specifikují vlastnosti aplikací Intranetu [2] (přístup k systému mají pouze členové organizace na povolení, systém je ve vlastnictví organizace a udržuje proprietární data), nelze u něj přesně definovat některé vlastnosti, jež mají na výkon reprezentovaný dobou odezvy systému významný vliv. V našem případě se jedná především o výkonnost koncových stanic jednotlivých uživatelů (kterou nelze kontrolovat), typ klientské aplikace, přenosovou rychlost Internetu (studenti mohou k systému přistupovat prostřednictvím veřejné sítě). Pro tyto části systému jsme v návrhu definovali minimální vlastnosti, které jsme dále respektovali a které ovlivňovaly návrh tak, aby bylo možno docílit maximálního výkonu při definovaných zdrojích.

Z celé problematiky řešení výkonu, která je obecně značně široká, jsme se zaměřili na řešení v oblasti programátorské a datové. Jedná se především o optimalizaci:

- HTML kódů,
- přenosů po veřejné síti a
- programového kódu aplikace serveru.

V oblasti HTML kódu jsme využili specifikovaných doporučení [3] a dále jsme se v optimalizaci zaměřili především na *vhodné použití tabulek* (značka <table>). Princip renderování tabulek browserem (zobrazení tabulky je provedeno až po úplném načtení jejího obsahu) zvláště u větších objemů dat nebo u nevhodně vnořených tabulek vedl často uživatele k provedení operace znovunačtení dokumentu (obnova, refresh), protože se mylně domnívali,

že server neodpovídá. Odstranění tohoto problému jsme řešili vhodným rozdělením tabulky do několika tabulek dílčích, ne však vnořených. Zároveň při specifikaci rozměrů tabulky a sloupců bylo důsledně uplatňováno *pravidlo použití absolutních hodnot*. Téhož pravidla bylo použito i v případě použití grafických prvků.

V oblasti minimalizace přenosů dat po veřejné síti jsme se zaměřili na:

- provádění důsledné *kontroly vstupních dat na straně klienta* (browseru) pomocí implementovaných funkcí v JavaScriptu.
- *oddělení a samostatné uložení* informací, které se opakují v jednotlivých dokumentech. Jedná se především o popisy stylů (CSS) a funkcí jazyka JavaScript, u kterých předpokládáme, že jsou po prvním načtení uloženy v lokální cache paměti klienta.
- *poskytování pouze relevantních dat a přímé cesty k informaci,* kde optimalizace směřovala do:
  - *efektivního a přitom minimálního použití hodnot atributů* TITLE (značka A) a ALT (značka IMG). Přestože na počátku projektu jsme zvažovali možnost užití těchto atributů jako jedné z možností realizace nápovědy, došli jsme na základě reakcí uživatelů v pilotním projektu k závěru, že uživatelé preferují rychlost komunikace před tzv. „prací s průvodcem“. Omezili jsme tedy užití pouze na atribut ALT u obrázků, kde uvedený text supluje obsah obrázku v případě, že uživatel natažení obrázků ve svém browseru potlačil.
  - *poskytnutí pouze těch dat, která uživatel žádá*. Tato vlastnost vyplývá z charakteru systému jako takového, kdy na tvůrce systému nebyl kladen požadavek, jako je tomu např. u systémů prezentačních, které musí uživateli nabídnout co nejvíce informací tak, aby ho v systému „udrželi“ co možná nejdelší dobu.
  - *poskytnutí alternativních cest získání informací*. V zásadě se jedná o obdobu řešení výše uvedeného problému. V této oblasti je uživateli nabízena přímá cesta k informaci (tj. není nucen k průchodu dokumenty, které v zásadě nežádá) a zároveň je mu dána možnost získat informaci i jiným komunikačním kanálem, např. zasláním elektronickou poštou.

V oblasti programového kódu aplikace serveru jsme se zaměřili na optimalizaci kódů ASP [4] a uložených procedur včetně dotazů SQL (Structured Query Language).

Při optimalizace kódu ASP jsme vedle standardních doporučení přistoupili k *důslednému užití metody write objektu response* místo užití konstrukce <%=obsah%> a k *minimalizovánému užití tzv. session a aplication proměnných*, které slouží k odstranění nedostatků způsobených bezstavovostí protokolu http, ale které zvyšují zátěž serveru.

Další způsob optimalizace vyplynul přímo z procesu vývoje ASP kódu. Přestože v procesu vývoje kódu pracujeme pouze s jednou kopií zdrojového kódu, která obsahuje plnou funkcionalitu, tj. zahrnuje funkce určené všem uživatelům, rozhodli jsme se užít postupu podobného *podmíněné kompilaci*, což znamená, že zdrojový kód obsahuje definované proprietární meta-značky (obr. 3), rozdělující kód do sekcí dle typu uživatele. Vyvinutý kód je pak na základě těchto meta-značek fyzicky rozdělen tak, že pro každý typ uživatele (student, učitel, administrátor) existuje samostatná aplikace.

```
...
společná část kódu
...
<!-- EKIT_STUDENT -->
...
část kódu určená pouze studentské sekci
...
<!-- /EKIT_STUDENT -->
...
společná část kódu
...
<!-- EKIT_UCITEL -->
...
část kódu určená pouze učitelské sekci
...
<!-- /EKIT_UCITEL -->
...
společná část kódu
...
```

Obr. 3 – Proprietární meta-značky

V oblasti přístupu k datům uloženým v relační databázi byl zvolen model, ve kterém *kód ASP nepřistupuje k datům přímo*, ale volá uloženou proceduru serveru prostřednictvím standardního rozhraní ODBC (Open DataBase Connectivity). Uložené procedury serveru však neobsahují pouze dotaz jazyka SQL, ale zajišťují realizaci aplikační logiky dané funkce. Tím, že systém užívá kompilované uložené procedury, dochází k výraznému zvýšení výkonu jak datového serveru tak web serveru, který aplikační logiku nemusí realizovat a u kterého nelze kompilaci skriptu zajistit. Zvolený model nám zároveň v okamžiku nedostatečného výkonu systému umožní snadno distribuovat výkon na dva počítače, kdy jeden bude zajišťovat služby Internetu (web, smtp) a šifrování a druhý bude realizovat aplikační logiku a správu dat.

V oblasti optimalizace dotazů jsme využili standardních metod váhové i syntaktické optimalizace, které zvolené prostředí (Oracle) nabízí, a i když jsme při vývoji použili sadu testovacích dat a dotazy optimalizovali, tak po ukončení pilotního projektu a provedení auditu náročnosti dotazů, jsme byli nuceni realizovat některé změny, které vedly k přeformulování některých dotazů, přidání indexů, atd.

4. Bezpečnost systému – předpoklad užití v prostředí otevřených sítí

Velký důraz při návrhu a implementaci systému jsme kladli na zajištění bezpečnosti. Tento požadavek vyplývá z charakteru systému, kdy:

- porušení důvěrnosti např. v oblasti hodnocení studenta ohrožuje vůbec užití systému (jeho provoz může být napaden s ohledem na zákon o ochraně osobních údajů či s ohledem na vnitřní předpisy VŠE Praha),
- výpadek může narušit průběh kurzu, kdy např. nebude možné získat práce odevzdané k hodnocení, nebude možno se přihlásit ke zkoušce, apod.,
- neautorizovaný zásah může zdiskreditovat hodnocení studenta, apod.

Stěžejní část řešení bezpečnosti směřuje do zajištění důvěrnosti přenosu, zajištění uložených dat a programových kódů, zajištění účtovatelnosti akcí a dostupnosti systému a zajištění autentizace uživatele.

Zajištění důvěrnosti jsme řešili v několika rovinách. Jedná se o zajištění bezpečným komunikačním kanálem, zajištění přístupovými právy, ochranou vlastnostmi SŘBD (Systém řízení báze dat) a vlastnostmi operačního systému a ochranou fyzickou. Pro zajištění bezpečného komunikačního kanálu, autentizaci serveru a zajištění integrity dat při přenosu využíváme protokolu SSL (Secure Socket Layer) společně s digitálním certifikátem serveru. Jelikož VŠE není certifikační autoritou, tak certifikát serveru, který je protokolem SSL vyžadován, je zároveň kořenovým certifikátem, tj. je podepsán vlastním privátním klíčem.

Systém přístupových práv využívá uživatelského jména a hesla ke kontrole přístupu k datům a je implementován deklaratorní formou (tj. přidělováním práv na úrovni databázových objektů a databázových uživatelů). Protože bylo nutné zajistit, aby aplikace pracovala s přístupovými právy na úrovni výskytů jednotlivých záznamů nebo i určitých atributů jednotlivých záznamů, navíc s rozlišením na práva čtení a zápisu, bylo nutné tuto deklaratorní formu doplnit i programovou kontrolou na úrovni jednotlivých procedur.

Další úrovní zajištění důvěrnosti je užití vlastností SŘBD a operačního systému, tj. zajištění hesly a u operačního systému Windows NT i použitím souborového systému NTFS včetně jeho správné konfigurace.

Na úrovni fyzické bezpečnosti je zabráněn fyzický přístup nepovolaným osobám k serveru, čímž eliminujeme možnost obejít systém NTFS prostřednictví zavedení systému MS-DOS nebo provést krádež disků.

Vedle těchto řešení byla přijata opatření směřující do implementace kódu, kdy není povoleno používat metody GET protokolu http při zasílání senzitivních informací, protože by tyto informace mohly být uloženy v historii prohlížeče a následně později zneužity jiným uživatelem.

Při řešení problematiky zajištění dat a programových kódů jsme se orientovali především na vypracování metodiky archivace dat a její dodržování, využití metody ukládání dat na diskové pole RAID level 5 a důsledný zápis operací do auditních log souborů. Vedle toho jsme (jak již bylo zmíněno výše v jiném kontextu) do méně bezpečného prostředí web serveru neumístili významné informace, které by mohly ohrozit chod systému.

Problém zajištění účtovatelnosti akcí a dostupnosti je v systému řešen jednak v souvislosti ochrany před útoky typu DOS (Denial of Services) a jednak jako ochrana před reklamacemi uživatelů. Systém využívá auditních záznamů jak na úrovni WWW serveru, databázového serveru, tak také na úrovni operačního systému. V oblasti útoků DOS jsme přijali opatření označované jako „planý poplach“, kdy v okamžiku zaznamenané nestandardní situace je generována SMS zpráva administrátoru systému. Kvalitně vystavěný systém vyhodnocení auditních souborů je pak také užit při odmítání neoprávněných reklamací uživatelů (s vysvětlením příčin).

Největší problém v řešení bezpečnosti spojujeme s autentizací uživatele. V současné verzi systému je autentizace řešena následujícím způsobem:

- převzetí databáze uživatelů, kterým má být umožněn přístup k systému eKit, tj. uživatelských jmen, která jsou jednoznačnou identifikací uživatele v rámci počítačové sítě VŠE z centrální databáze studentů,
- generování databáze hesel pro jednotlivé uživatele a jejich uložení v systému eKit,
- distribuce hesel a
- autentizace uživatele pomocí uživatelského jména a hesla proti vlastní databázi uživatelských jmen a hesel prostřednictvím bezpečného komunikačního kanálu s užitím certifikátu eKit.

V tomto procesu je klíčovým bodem způsob distribuce hesel, kde *nelze doporučit použití mechanismu běžně na Internetu užívaného*, kdy heslo je uživateli zasláno elektronickou poštou. Nemožnost použít tento mechanismus je dán tím, že požadavek na zajištění bezpečnosti je tak vysoký, že nelze heslo zasílat nechráněným komunikačním kanálem,

kterým elektronická pošta bezesporu je. Je zde totiž jistá pravděpodobnost odposlouchávání přenosu po síti. Proto doporučujeme distribuci hesel řešit prostřednictvím osobního předání hesla a to tak, že student navštíví pracoviště distribuce hesel, kde po základní lustraci (kontrola totožnosti dle vysokoškolského indexu) je mu heslo administrátorem vydáno. Tento způsob řešení má několik nevýhod, mezi které patří obrovská administrativní náročnost. Proto v současné době vyvíjíme dvě základní varianty řešení. První, která užívá *osobních digitálních certifikátů* a druhá, kterou lze chápat jako variantu systému označovaného *„Secure Single Sign On"*. Obě uvedené varianty mají své přednosti i negativa. Zatímco varianta první kopíruje současný trend v užívání standardů X.509 a systémů označovaných jako PKI (Public Key Infrastructure), neodstraňuje problém s distribucí certifikátů (tj. jejich vydávání) a jako taková až do okamžiku implementace této technologie v rámci celé VŠE by kvalitativně nenahradila stávající systém. Druhá varianta, která se zaměřuje na užití autentizačních mechanismů implementovaných v systému Novell Netware, kdy uživatel je autentizován oproti systému NDS (Novell Directory Services), odstraňuje základní problém s distribucí hesel, neboť uživatel aplikuje při autentizaci mu již známé heslo. Jediným problémem této varianty jsou výkonové charakteristiky, tj. doba odezvy systému Novell Netware na požadavek verifikace (počet uživatelů Novell Netware ze strany studentů je cca 10 000), což při součtu s dobou odezvy, na kterou má významný vliv přenosová rychlost ve veřejné síti, může značně snížit spokojenost uživatelů a na druhé straně zvýšit množství opakovaných požadavků.

5. Kolize vývoje, výkonu a bezpečnosti – kde je priorita

Zajistit současně kvalitní vývoj, rozvoj a snadný provoz, vysoký výkon a bezpečnost při omezených především technických zdrojích, je velmi složitou otázkou. V zásadě se jedná o kolizní situace, kdy vyzdvižení jedné vlastnosti vede ke snížení kvality druhé. Proto je nutné definovat v rámci řešení projektu priority, které vytvoří mantinely řešení a minimální vlastnosti systému tak, aby bylo možno najít optimální řešení cestou kompromisů. V zásadě lze konstatovat, že při nezměněných technických podmínkách zůstává výkon v rozporu jak s možností snadného užití, tak i s vysokou bezpečností. Tuto skutečnost lze dokumentovat následujícími příklady.

Při vlastním návrhu a implementaci aplikace jsme narazili na problémy, které s sebou nese současný požadavek výkonnosti a snadného rozšiřování funkčnosti aplikace pomocí dalších modulů. Zatím co požadavky na výkonnost jasně vedou k maximálnímu využití uložených (a tedy předkompilovaných) procedur a vytváření speciálních „maximálně" optimalizovaných procedur pro provádění jednotlivých akcí, požadavek na snadné rozšiřování funkčnosti vede spíše k vytváření obecných parametrizovaných procedur, které by využívaly ve velké míře dynamicky sestavovaných dotazů. V tomto okamžiku naše rozhodnutí vedlo k optimalizaci výkonu, protože při nepředvídatelném počtu současně pracujících uživatelů by se taková procedura mohla stát úzkým „hrdlem" systému.

Obdobně jsme posuzovali i vztah mezi zajištěním výkonu a zajištěním bezpečnosti a to v okamžiku, kdy jsme rozhodovali, ve které části použít šifrovanou komunikaci, jež výrazně prodlužuje dobu odezvy. Dospěli jsme k názoru, že z důvodu zabezpečení proti odposlechu je nutné šifrovat celou komunikaci.

Podobné rozhodnutí jsme museli podstoupit při rozhodování o způsobu implementace přístupových práv, kdy jsme se nakonec rozhodli pro již zmíněnou programovou kontrolou přístupových práv úrovni jednotlivých databázových procedur zajišťující řízení přístupu

aktuálního uživatele k jednotlivým záznamům tabulek a to i přesto, že tato kontrola zvyšuje zátěž serveru.

O jednom z problémů mezi zajištěním bezpečnosti a snadným vývojem, zajištěním provozu a rozvojem jsme se zmiňovali v části, která diskutuje problém distribuce hesla.

6. Závěr

Vzhledem k překotnému vývoji a šíři nabídky technologií použitelných pro Internetové aplikace není snadné zvolit především vhodnou platformu a vývojové prostředí dané aplikace. A to tak, aby „přežila“ určitou dobu provozu, byla schopna relativně snadného rozšiřování a náklady vynaložené na vývoj tak nepřišli vniveč díky brzkému přechodu na jinou platformu. S jistou mírou nadsázky můžeme říci, že vyvíjíme aplikace na již zastaralých technologiích, pokud vezmeme v úvahu čas potřebný pro absorbování a následný vývoj v technologiích nových samotnými vývojáři. Proto je dost pravděpodobné, že některé aspekty, které jsme v našem příspěvku zmínili a které jsme byli nuceni brát v úvahu v rámci našeho projektu eKIT, pozbudou své váhy, některé získají na významu a zároveň mnohé jiné se jistě v budoucnosti objeví.

Na závěr bychom pouze rádi zopakovali a vřele doporučili známý fakt, že předprojektová příprava/analýza všech možných aspektů, která alokuje zpočátku jisté zdroje se v budoucnu mnohonásobně vyplatí.

Literatura

1. ŘEPA, Václav. Analýza a návrh informačních systémů, 1. vyd. Praha: Ekopress, 1999, 160s. ISBN 80-86119-13-0
2. GREER, Tyson. Intranety principy a praxe, 1. vyd. Praha: Computer Press, 1999, 5s. ISBN 80-7226-135-5
3. Building High Performance HTML Pages, 2001, Microsoft Corporation, http://msdn.microsoft.com/workshop/author/perf/perftips.asp
4. HOMER, Alex, Sussman, Dave, Francis, Brian. Active Server Pages 3.0 profesionálně, 1. vyd. Praha: Computer Press, 2000, ISBN: 80-86097-47-1