TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových
inženýrských studií

DIPLOMOVÁ PRÁCE

Návrh, vývoj a implementace vnitropodnikového informačního systému založeného na třívrstvé architektuře

Design, development and implementation of information intradepartmental system based on three level architecture

Liberec 2003 **Roman Špánek**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových
Inženýrských studií

Studijní program: 2612 M – Elektrotechnika a informatika
Studijní obor: 2612 T – Automatické řízení a inženýrská informatika

Návrh, vývoj a implementace vnitropodnikového informačního systému založeného na třívrstvé architektuře

Design, development and implementation of information intradepartmental system based on three level architecture

Jméno: **Roman Špánek**
Vedoucí diplomové práce: **RNDr. Ladislav Mečíř**
Konzultant: **Ing. Pavel Hujer**

Rozsah práce:

Počet stran : 63
Počet obrázků: 22
Počet příloh: 8
Počet CD-ROM: 1

19.5.2003

Špánek Roman DP – 2003 Ved. DP: RNDr. Ladislav Mečíř

Návrh, vývoj a implementace vnitropodnikového informačního systému založeného na třívrstvé architektuře

Resumé:

Cílem diplomové práce je vytvořit databázovou aplikaci, která by vyřešila problémy a minimalizovala náklady spojené s vedením agendy o zaměstnancích společnosti Peguform Bohemia, k.s..

Aplikace je vyvinuta v prostředí intranetu s využitím databázového serveru MS SQL Server 7.0, aplikačního serveru MS IIS 5.0 a skriptovacího jazyka PHP.

Schválení aplikace, jako oficiálního telefonního seznamu a registru zaměstnanců, proběhlo 10.4.2003. Spokojenost s produktem vyjádřili uživatelé formou hlasování, které skončilo 112 ku 20 ve prospěch aplikace.

Abstract:

The aim of thesis is make database application which was reduce expense and solve problem with making employee registry Peguform Bohemia, k.s. company.

Application was developed used intranet technology which was based on MS SQL Server 7.0, MS IIS 5.0 application server and language of PHP.

Application was authorized on 10.4.2003 like official registry of employee. Satisfaction of application was expressed take by vote of employees. The score of vote was 112 to 20 for benefit to satisfied users.

Prohlášení:

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/200 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TU má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že souhlasím s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 18. května 2003

Podpis:

Úvodem práce bych rád poděkoval panu Ing. Pavlu Hujerovi ze společnosti Peguform Bohemia, k.s., který mi byl velmi nápomocen při vytváření aplikace a dokumentace. Nedocenitelné byly zejména jeho zkušenosti s cílovým prostředím společnosti, které umožnily tak kladné přijetí aplikace uživateli.

Velký dík rovněž patří vedoucímu diplomové práce, panu RNDr. Ladislavu Mečířovi, který přispěl podmětnými připomínkami vycházejícími z jeho bohatých zkušeností s internetovými aplikacemi.

1. Úvod

Posláním diplomové práce, zadané společností Peguform Bohemia k.s., která je jedním z předních dodavatelů plastových dílů v automobilovém průmyslu, je zlepšení současného stavu evidence údajů o telefonních číslech a účastnících interního telefonního seznamu, včetně dalších doplňujících údajů.

Ekonomický rozvoj společnosti, získávání nových dlouhodobých zakázek, s tím související stavba nového závodu a nástup značného počtu nových zaměstnanců, přináší zvýšené nároky na údržbu a zpracování personálních a s nimi souvisejících agend.

Jednou z těchto personálních agend společnosti zadavatele je telefonní seznam, jenž byl z historických a organizačních důvodů udržován v podobě Excelovského sešitu, který byl pravidelně distribuován k účastníkům interního telefonního spojení a později byl převáděn do statické podoby v HTML formátu a tak zpřístupňován uživatelům podnikového intranetu.

Protože tento způsob vytváření, udržování a prezentace dat měl značné nevýhody, odrážející se především v jeho nízké aktuálnosti, rozhodl se zadavatel pro vytvoření nové verze telefonního seznamu a registru zaměstnanců v podobě dynamické intranetové aplikace.

Softwarový projekt zahrnující vytvoření dynamické intranetové aplikace je obsahem této diplomové práce.

1.1. Informační technologie

Pod, především v poslední době, často používaný název informační technologie, lze bez obav zařadit vše, co se byť jen okrajově dotýká problematiky výpočetní techniky a sdílení dat. Pro řešení daného projektu je samozřejmě použita jen malá část těchto prostředků. Použité prostředky budou popsány podrobněji.

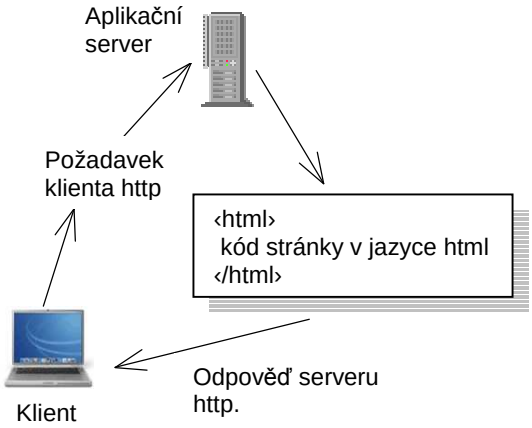
1.1.1. Databázové systémy

Lidé od chvíle, kdy jsi uvědomili sami sebe a začali přemýšlet a zkoumat své okolí, měli potřebu uchovávat informace pro pozdější použití. Na počátku jen v paměti, později na jednoduchých papírových seznamech, až po relativně dobře propracované kartotékové systémy, jak jsou známé ještě dnes, například u lékaře. S nástupem výpočetní techniky se však už nejeví jako dostačující a ke slovu se

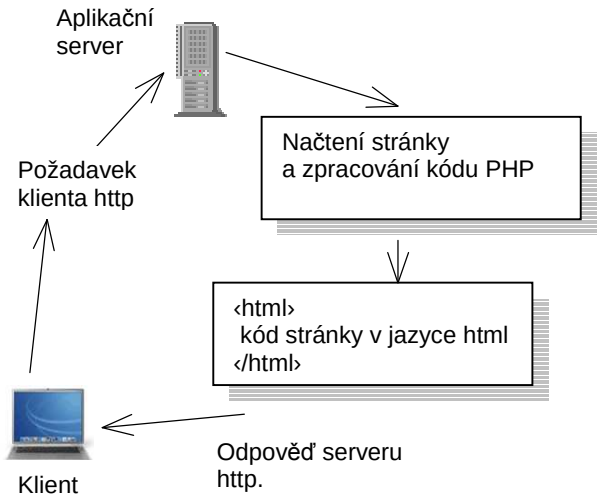
dostávají elektronické systémy pro uchování dat. Na počátku byly využívány soubory, které jsou do jisté míry dobře použitelné pro statické počítačové aplikace a malý objem dat. Pro moderní gigabajtové aplikace jsou však zcela nevyhovující a jsou nahrazovány systémy s řízením báze dat a dnes převládajícím relačním modelem.

1.1.2. Aplikační servery

Při prohlížení internetových stránek vyvstane otázka, odkud se stránky vzaly? Na tuto otázku je jednoduchá odpověď, stačí jen nastínit princip fungování internetu (Viz obr. 1). Na počítači je zadána adresa společně s názvem stránky,



Obr. 1 Komunikace na internetu



Obr. 2 Zpracování kódu PHP

která je požadována a odeslána jako požadavek na internetovou síť. Podle adresy je určen počítač, na kterém se stránka nachází a ten ji „ochotně“ pošle zpět formou odpovědi. Tento počítač se nazývá aplikační server a jeho funkce byla na počátcích internetu opravdu taková, jak je popsána výše. Dnes je prostředí internetu tvořeno převážně stránkami s dynamicky generovaným obsahem, to znamená, že na aplikačním serveru je nainstalován program, který podle uživatelských požadavků vytváří stránky s patřičným obsahem (Viz obr. 2).

1.1.3. Skriptovací jazyky

Skriptovací jazyky jsou „kouzelnou hůlkou“, pomocí které lze měnit stránky podle požadavků uživatele. Dynamické generování stránek lze provádět na straně serveru nebo na straně klienta. Velkou výhodou skriptovacích jazyků pracujících

na straně serveru je odstranění problémů s kompatibilitou prohlížečů, která je v prostředí internetu velkým a do jisté míry stále nevyřešeným problémem. Mezi další výhody patří rychlejší načítání stránek (kód skriptu je proveden na serveru a není posílán společně s odpovědí zpět klientovi) a v neposlední řadě lepší zabezpečení internetových stránek. Skripty jsou do stránek vloženy pomocí speciálních značek, které pak řeknou odpovídajícímu modulu, který je na aplikačním serveru nainstalován, jakou část kódu bude zpracovávat. Server provede příkazy obsažené v požadavku uživatele a vytvoří podle nich výslednou stránku, která je složena jen z kódu jazyka HTML. To znamená, že uživatel, který si prohlíží naše stránky, nikdy neuvidí zdrojový kód skriptovacího jazyka.

1.1.4. Tenký klient a uživatelské rozhraní

Doposud byla rozebírána jen problematika spojená se zpracováním dat od klienta, ale stejně důležitou částí je podání informací pro klienta srozumitelnou formou. Tenký klient slouží pro zobrazení výsledné stránky, kterou zpracoval aplikační server a skriptovací stroj. Jeho úkolem je převést data z jazyku HTML do grafické podoby. Pro zadaný úkol je použit přístup uživatelů k systému pomocí vnitropodnikové informační sítě (intranetu). Řešením je tedy použít jeden ze stávajících internetových prohlížečů, v našem případě byl zvolen MS Internet Explorer v kombinaci s alternativní Mozillou. Vlastní uživatelské rozhraní musí být co možná nejjednodušší a použitelné i uživatelům, kteří nemají žádné nebo jen základní znalosti práce s internetem či intranetem. Nutností je použití co nejjednodušších formulářů v kombinaci s dobře zvolenou grafickou podobou.

1.2. Informace a ekonomika

Informace je slovo které se začíná objevovat ve spojení s ekonomikou stále častěji. Jedním ze základních prvků pro dobré řízení podniku je dokonalá znalost a aktuálnost informací o zaměstnancích. Pro daňový a personální útvar, management, ale i pro řadové zaměstnance, kteří obvykle pracují v týmech, je tedy nutná dobrá komunikace a k té jsou potřebné čerstvé informace o umístění a zařazení pracovníků, kteří mohou daný problém pomoci řešit. Informaci lze tedy bez nadsázky označit za klíčový prvek a dnes i obchodní artikl s velkou cenou.

2. Teoretické základy práce

2.1. Vývoj databázových systémů

Vývoj databázových systémů je do značné míry závislý na možnostech výpočetní techniky. Dnes nejrozšířenější relační model dat, byl v mnoha ohledech překonán objektovým modelem, ale s ohledem na nedostatečné možnosti výpočetní techniky, nedošlo k jeho masivnějšímu nasazení.

2.1.1. Manuální systémy (kartotéky)

Manuální systémy byly prvním pokusem o uložení informací do celku, do kterého by bylo možné snáze přidávat nebo ubírat informace, stejně jako umožnit rychlejší vyhledávání. Jejich velkou nevýhodou byly velké náklady na údržbu dat. [1]

2.1.2. Agendové informační systémy

Byly prvním pokusem o strojové zpracování informací. Principem bylo vytvoření programu, který měl jako vstupní parametry svá data. Hlavní nevýhodou tohoto systému je závislost programu a dat. Každý program měl svá vlastní data, která byla často použita i v jiných programech, následkem čehož vznikala duplicita dat. Toto vedlo ke zvýšení nákladů na údržbu a pořizování dat, tento jev se nazývá *redundance dat*. Dalším rizikem je porušení *integrity dat*, souvislosti mezi daty nebudou důvěryhodné. [1]

2.1.3. Báze dat

Báze dat je odpovědí na hlavní nedostatky agendového systému. Báze dat je soustředěním všech potřebných dat, se kterými programy pracují, do jednoho centrálního bloku. Tím byla odstraněna redundance a porušení integrity dat, ale stále přetrvával hlavní nedostatek a to závislost dat na programu, ze kterého většina problémů pocházela. [1]

2.1.4. Databázový systém

Databázový systém je složen z báze dat, která je doplněna o SŘBD (Systém Řízení Báze Dat), jehož alternativou je anglická zkratka DBMS (DataBase Managmenet System), která má vše podstatné o organizaci dat definováno v katalogu.

SŘBD tvoří nadstavbu nad bází dat a zajišťuje:

- definici i redefinici dat,
- vytvoření přístupového mechanizmu,
- ochranu dat (proti výpadku, proti neoprávněnému zásahu ..),
- aktualizaci dat (přidat, měnit, rušit),
- vstup a výstup do BD,
- třídění,
- vybírání dat podle podmínky.

Pomocí SŘBD byly odstraněny problémy předchozích řešení, tedy především:

- závislost programu a dat,
- redundance,
- kompatibilita dat,
- ochrana dat,
- integrita dat.

SŘBD přináší také následující výhody:

- Programátor nemusí znát organizaci ani fyzické uložení dat v bázi dat.
- Programátor přistupuje k datům pomocí jazyka strukturovaných dotazů SQL (z anglického Structured Query Language). [1]

2.2. Implementační datové modely

Slouží k popisu uložení a vzájemných vazeb dat, tak jak jsou fyzicky uloženy v databázi. Pro vyjádření datového modelu lze použít různých prostředků, jako například množinu formálních pravidel, grafický návrh nebo souhrn deklarací typů dat.

2.2.1. Souborový systém

V literatuře se obvykle setkáme s anglickou zkratkou FMS (File Management System). Model vycházel z popisu souboru, ve kterém jsou data uložena sekvenčně tj. za sebou. Programátor tedy bezpodmínečně musel znát přesné pořadí dat a strukturu souboru. Předností modelu je jednoduchost.

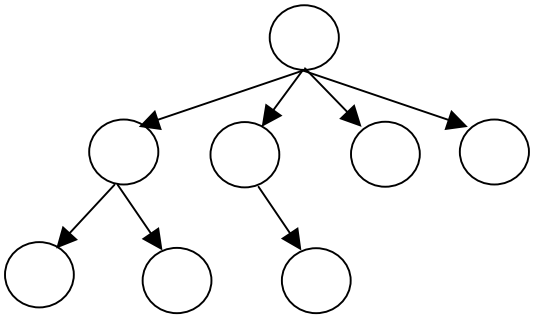
Hlavní nedostatky lze demonstrovat na příkladu tří hlavních operací, které jsou obvykle prováděny s daty:

- *Změna struktury* – tedy například přidání nebo odebrání položky:
  - sekvenční čtení souboru a zapisování do nového souboru požadovaného tvaru.
  - po vytvoření nového souboru je třeba starý vymazat a nový přejmenovat na jméno původního.
- *Třídění* – je třeba znovu sekvenčně načíst celý soubor dat, setřídit a poté zapsat v novém pořadí. Velmi nevhodné zejména u velkých souborů.
- *Vyhledávání* – opětovné sekvenční načtení souboru a porovnání každé položky s kritériem pro vyhledávání.

Je tedy patrné, že správa takové databáze je velmi složitá a časově náročná. Rovněž zranitelnost například výpadkem napájení nebo selhání hardwaru je značná. [1]

2.2.2. Hierarchický model dat

Základní myšlenkou modelu je uchovaní dat ve stromové struktuře (viz obr. 3), tedy systém odkazů otce na syna.



Obr. 3 Stromová struktura

První verze modelu spatřila světlo světa koncem šedesátých let a vyvinuly ji společnosti IBM a North American Aviation pro projekt vesmírných letů Apolo. Hierarchický model nemá stanoven žádný standard. Základní jednotkou je záznam (segment), který je tvořen poli obsahující jednotlivé údaje o prvcích záznamu. Fyzická struktura dat je obvykle realizována pomocí SŘBD a většinou se jedná o zřetězený seznam položek s ukazateli. Bývá tvořen dvou až tří prvkovým seznamem ukazatelů. První obvykle ukazuje na místo, kde je informace reprezentovaná, druhý na následující prvek a třetí na prvek předcházející, pokud žádný prvek nenásleduje, má tento hodnotu

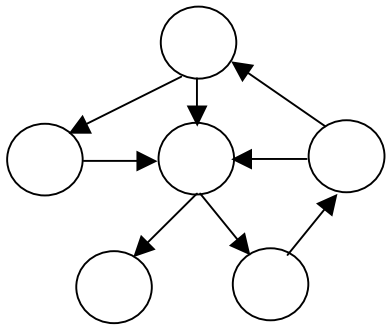
nula (NULL). Hlavní nedostatky tohoto modelu lze shrnout do následujícího seznamu:

- Změna struktury obnáší vybudování celého seznamu znovu.
- Řetězec pointrů je náchylný na chyby, jak hardwaru tak i softwaru. Chyba obvykle znehodnotí celou databázi.
- Model je nepružný, pokud požadujeme jiné vazby mezi prvky, než jaké jsou definovány.
- Operace vkládání a rušení záznamů jsou složité.
- Nejvážnějším nedostatkem je nemožnost definovat vztahy m:n (více k více) prvků, aniž by došlo k redundanci dat.

S výhodou lze tento model použít tam, kde i fyzická struktura je hierarchická. [1]

2.2.3. Síťový model dat

Síťový model, standardizovaný v roce 1972 výborem Database Task Group (DBTG), vznikl v rámci konference o jazycích datových systémů (Conference on Data Systems Languages – CODASYL). Model byl pojat jako rozšíření hierarchického modelu odstraňující jeho hlavní nevýhody, tedy omezení vazeb mezi prvky (Viz obr. 4).



Obr. 4 Síťový model dat

Síťový model umožňuje vytvářet libovolné vazby mezi prvky stejné, nižší nebo vyšší úrovně. Správné definování těchto vztahů je velmi důležité pro efektivní využívání databáze. Tato volnost ve vytváření vztahů je největší výhodou, ale zároveň i slabinou modelu. Lze snadno vytvořit nové vztahy a dosáhnout tím potřebné struktury, ale velmi složitá struktura se špatně a nákladně udržuje. Komplikovanost vztahů rovněž vede k náročnější realizaci běžných úkolů, jako je například vložení nebo odstranění dat.[1]

2.2.4. Relační model

Relační model dat vznikl, na rozdíl od předchozích modelů, nejprve na teoretických základech a je založen na teorii množin a predikátové logice. Jeho zakladatelem je E. F. Codd, který své výsledky publikoval v roce 1970.

Nejprve definujme základní pojmy relační algebry:

- *Def. 1 Kartézský součin*

  Nechť je dán systém $\{D_i \mid 1<=i<=n\}$ neprázdných množin. Potom kartézským součinem $R = D_1 \times D_2 \times ....D_n$ nad $D_1, D_2, ...D_n$ nazýváme množinu všech uspořádaných n-tic $(d_1, d_2 ....d_n)$ takových, že $d_i \in D_i$ pro $1<=i<=n$.

- *Def. 2 Relace*

  Nechť je dán systém $\{D_i \mid 1<=i<=n\}$ neprázdných množin. Potom podmnožinu kartézského součinu $R \subset D_1 \times D_2 \times ....D_n$ nazýváme relací stupně n (n-ární relace) nad $D_1, D_2, ...D_n$. Prvky relace jsou uspořádané n-tice $(d_1, d_2 ....d_n)$ takové, že $d_i \in D_i$ pro $1<=i<=n$.

Relaci lze považovat v jistém slova smyslu za matici nebo tabulku o rozměrech m x n.

Dále je třeba definovat pojmy:

- *Def. 3 Báze dat*

  je konečná množina v čase proměnných konečných relací, které jsou definovány nad doménami ze systému množin $\{D_i \mid 1<=i<=n\}$. Změna v čase spočívá v možnosti přidání, vynechání jednoho nebo více řádků v některých relacích, nebo ve změně některých hodnot v již existujících řádcích některých relací.

- *Def. 4 Populace*

  Relaci R(t) v čase t_i nazýváme populace.

- *Def. 5 0-tá normální forma*

  Pokud některá z domén je opět relací, říkáme, že relace je v 0-té normální formě.

Relace je možné chápat, z matematického hlediska, jako matice a lze s nimi provádět odpovídající matematické operace. Jako alternativní zápis, bude vždy uveden příklad zápisu matematické operace příkazem jazyka SQL, bude tím jasněji definovano spojení mezi matematickým modelem a jeho fyzikální realizace. Proto je nutné definovat základní příkaz jazyka SQL - SELECT. Tento má velmi složitou syntaxi, která je z důvodu srozumitelnosti zjednodušena na:

SELECT ‹SeznamPolí›
FROM ‹SeznamMnožinZáznamů› JOIN ‹SpojovacíPodmínka›
WHERE ‹VýběrováKritéria›
GROUP BY ‹SeznamPolíKSeskupení›
HAVING ‹VýběrováKritéria›
ORDER BY ‹SeznamPolíKSeřazení›;

Kde: ‹SeznamPolí› je výčet polí, které bude obsahovat výsledek, pokud je uvedena „*“, pak bude vybrán kompletní seznam polí
‹SeznamMnožinZáznamů› je seznam tabulek nebo pohledů, na kterých provádíme příkaz SELECT
‹SpojovacíPodmínka› je podmínka, která je použita pro spojení výsledků
‹SeznamPolíKSeskupení› je seznam polí, která budou ve výsledku seskupena
‹VýběrováKritéria› slouží pro omezení výsledku na určitou množinu polí, která splňují danou podmínku
‹SeznamPolíKSeřazení› výsledek je seřazen podle polí, která jsou zde uvedena

Pak můžeme definovat následující operace:

- *Def. 6 Průnik relací*

  Nechť R_1 a R_2 jsou n-ární relace nad stejnou množinou domén $\{D_i \mid 1<=i<=n\}$. Potom $R = R_1 \cap R_2 = \{(d_1,d_2 ....d_n) \in R \mid (d_1,d_2 ....d_n) \in R_1 \Lambda (d_1,d_2 ....d_n) \in R_2\}$ je průnik relací R_1 a R_2.

  Analogicky:

  SELECT JmenoTabulky1.* FROM JmenoTabulky1

LEFT JOIN JmenoTabulky2
ON(JmenoTabulky1.JmenoPole1=
JmenoTabulky2.JmenoPole2)
WHERE JmenoTabulky2.JmenoPole3='hodnota'
Výsledkem bude spojení dvou množin pomocí společných polí.

- *Def. 7 Kartézský součin*

Nechť R a S jsou relace stupně m a n, potom kartézským součinem RxS nazýváme relaci stupně m+n takovou, že platí:

RxS = {r.s| $r \in R \ \Lambda \ s \in S$} kde r.s je m+n - tice tvaru (r_1,r_2, ...r_m,s_1,s_2, ...s_n)
Kartézský součin umožňuje z jednoduchých relací vytvářet složitější relace s větším počtem atributů. Prvky kartézského součinu jsou všechny m+n–tice, které mohou vzniknout, tedy každý prvek R v kombinaci s každým prvkem z S.
Analogicky:
SELECT JmenoPole FROM JmenoTabulky

- *Def. 8 Projekce*

Nechť R je relace a A je její atribut, potom R do A je opět relace R[A] definovanou R[A] = {r.A | $r \in R$} a nazvanou projekce
Analogicky:
SELECT JmenoPole1,JmenoPole2, FROM JmenoTabulky
ORDER BY JmenoPole1
Z původní množiny polí vrací pouze pole s nazvy JmenoPole1 a JmenoPole2.

- *Def. 9 Restrikce*

Nechť R je relace, A je její atribut a konstanta k je prvkem domény atributu A. θ je binární operátor {=, <, >, $\neq$, $\geq$, <=} potom restrikce R je opět relace označovaná R[A θ k] a definovanou R[A θ k] = {r| $r \in R \ \Lambda$ r.A θ k}.

Kde: r.A hodnota atributu A v n-tici r

Analogicky:

SELECT *FROM JmenoTabulky WHERE

JmenoPole = 'podmínka'

Výsledek obsahuje pouze ty záznamy, které vyhovují podmínce.

- *Def. 10 Spojení*

Nechť R, S jsou relace a A, B jejich atributy a relace $T \subset AxB$, potom spojením R a S nazýváme relaci definovanou takto:

$R [ATB] S = \{r.s \mid r \in R \ \Lambda \ s \in S \ \Lambda \ (r.A, s.B) \in T\}$

- *Def. 11 Přirozené spojení*

Nechť v def.9 je T relace rovnosti na $A \cup B$, potom přirozeným spojením relací R a S podle atributů A,B označujeme R [A*B] S definovanou (R [A=B] S) [$\overline{B}$]

Analogicky:

SELECT *FROM JmenoTabulky1

JOIN JmenoPoleVýsledku

ON JmenoTabulky.JmenoPole=

JmenoPoleVýsledku.JmenoPole2

WHERE JmenoTabulky.JmenoPole “Operátor“ 'podmínka'.

Provede spojení dvou tabulek, relací, které mají odpovídající záznamy v zadaných polích. Pokud operace splňuje následující podmínky:

- Operátorem je rovnost.
- Spojení se účastní všechna společná pole obou tabulek.
- Do výsledku se smí zapsat jen jedna množina společných polí.
- Pak mluvíme o přirozeném spojení.
- Pokud je použit jiný Operátor než „=“ pak mluvíme o Theta-spojení.

- *Def. 12 Klíč*

Klíč K relace R je podmnožina atributů relace R, které jsou nezávislé na čase, jednoznačně identifikují každou n-tici

a zároveň platí, že množina atributů tvořících klíč je vzhledem k identifikaci minimální.

- *Def. 13 Primární atribut*

  Atribut, který se vyskytuje alespoň v jednom klíči nazýváme primární. Všechny ostatní atributy jsou neprimární.

V relačním modelu je záznam uložen jako jeden řádek tabulky. Matematická definice relace také zamezuje vložení dvou stejných řádků do stejné tabulky. Jedno pole nebo kombinace více polí záznamu, která slouží pro jednoznačnou identifikaci záznamů tabulky, se nazývá klíč. Spojení záznamů, které jsou uloženy ve dvou různých tabulkách, ale mají stejnou hodnotu klíčů, se nazývá relační propojení. Relační propojení umožňuje snadno získávat odpovědi na nejrůznější dotazy. Různé vztahy mezi tabulkami (relační propojení) se realizují jako množinový součet, tedy jako množina s obecně definovaným algoritmem. Hlavní výhody relačního modelu lze shrnout do následujících bodů:

- Naprostá volnost ve vytváření vztahů.
- Snadnost změn struktury tabulky – přidání, změna, či zrušení sloupce.
- Snadnost vytvoření tabulky jako podmnožiny stávající tabulky, nebo tabulek.
- Transakční zpracování.
- Důslednější zajištění integrity dat.[1]

Důležitým prvkem při návrhu relačního databázového modelu je struktura samotných relací. Prvotním a nejdůležitějším úkolem je navrhnout model tak, aby byl schopen odpovědět na jakoukoli rozumnou otázku. Druhým prvkem je odstranění redundancí a co nejefektivnější využívání prostředků. Tomuto účelu slouží normalizace. Ta stanovuje pět normálních forem, které zaručují, s jistou pravděpodobností efektivní, správný návrh modelu.

- *První normální forma*

  Relace je v první normální formě, pokud jsou všechny její atributy definovány nad skalárními obory hodnot (doménami).

To znamená, že jednotlivá pole obsahují dále nedělitelné údaje. [2]

- *Druhá normální forma*

  Relace je ve druhé normální formě, pokud je v první normální formě a všechny její atributy jsou závislé na celém kandidátním klíči. [2]

- *Třetí normální forma*

  Relace je ve třetí normální formě, pokud je ve druhé normální formě a všechny její neklíčové atributy jsou vzájemně nezávislé.[2]

Výše uvedené normalizace byly v původním návrhu Coddovy formulace relační teorie a většinou plně postačují pro správný návrh relačního modelu. Pro úplnost zde ještě uveďme i ostatní normální formy.

- *Boyce/Coddova normální forma*

  Tuto normální formu lze pokládat za jistou variaci třetí normální formy. Je využívána pro speciální relace s více kandidátními klíči. Boyce/Coddova normální forma je aplikovatelná pokud jsou podmínky:

  - Relace má dva nebo více kandidátních klíčů. [2]
  - Nejméně dva z kandidátních klíčů jsou složené. [2]
  - Kandidátní klíče se v některých atributech musí překrývat. [2]

  Pak relace je v Boyce/Coddově normální formě, pokud mezi kandidátními klíči není žádná funkční závislost.

- *Čtvrtá normální forma*

  Čtvrtá normální forma říká, že v jedné relaci se nesmí spojovat nezávislé opakované skupiny.[2]

- *Pátá normální forma*

  Pátá normální forma se týká spojené závislosti. Je-li entita1 spojena s entitou2, entita2 je spojena s entitou3 a ta je zpětně spojena s entitou1, pak všechny entity musí být součástí jednoho vektoru hodnot.[2]

2.2.5. Objektově orientovaný model dat

Objektově orientovaný model je vhodný pro svůj přirozenější popis dat vzhledem ke skutečnému světu. Hlavní rysy objektově orientovaného přístupu k datům lze shrnout do následujících bodů:

- *Zapouzdření dat*, přístup k objektu je možný pouze pomocí nadefinovaných metod, což zvyšuje bezpečnost.
- *Dědičnost*, nové metody mohou dědit vlastnosti nebo být odvozeny od stávajících metod. Lze tedy používat stávající metody a postupně je rozšiřovat o nové vlastnosti, nežádoucí vlastnosti lze odstranit pomocí přetížení metody.
- *Polymorfizmus*, odvozené objekty musí být typově kompatibilní se svými předchůdci. Je tedy možné využít stávající metody pro vytvoření nových.

V objektově orientovaných SŘBD (dále již jen OOSŘBD) je velmi snadná aktualizace dat, která je však vykoupena složitostí návrhu univerzálního a komplexního modelu. OOSŘBD je tedy velmi náročný na výpočetní výkon počítačů, který byl donedávna nedostačující. V poslední době však došlo v výraznému nárůstu výkonu a tak lze předpokládat výraznější prosazování OOSŘBD v praxi.[1]

2.2.6. Objektově relační model

Složitý návrh OOSŘBD vedl ke vzniku objektově relačního SŘBD (ORSŘBD), které spojuje vlastnosti objektově orientovaného a relačního modelu dat. Tento model je v současné době využíván například v databázovém stroji PostgreSQL a lze jej právem považovat za velmi nadějný model, který bude do budoucna hojně využíván.[1]

2.3. Aplikační servery

Jak již bylo zmíněno v úvodu, aplikační (můžeme se také setkat s názvem webový) server vyřizuje požadavky klientů, které mu jsou zasílány přes internetovou nebo intranetovou síť. Aplikační server je fyzický počítač, např. třídy PC (Personal Computer) se systémem MS Windows či Unix nebo velký sálový

počítač, vybavený systémem Unix. Na zvoleném počítači je nainstalován software, který zajišťuje odpovídající reakce serveru. Aplikační servery lze rozdělit z hlediska finanční nákladnosti na servery komerční a volně šiřitelné.

2.3.1. Microsoft IIS 5.0

Řešením od společnosti Microsoft je aplikační sever IIS (Internet Information Server), který je součástí systémů Windows 2000 Professional a Windows XP Professional. Server patří do druhé skupiny a představuje výhodné řešení zejména pro velké vnitropodnikové (intranetové), ale také veřejné informační servery. Jedná se o robustní server, jehož poslední verze je 5.0. Jeho hlavní nevýhodou je chybějící kompatibilita s jinými operačními systémy, než jsou produkty společnosti Microsoft. To je také jeden z hlavních důvodů, proč tento server nemá dominantní zastoupení na internetu, kde stále většina skutečně velkých aplikačních serverů využívá operační systém Unix.

2.3.2. Apache

Jedná se o volně dostupný, velmi výkonný a oblíbený server. V literatuře se obvykle uvádí zastoupení okolo šedesáti procent v celé internetové síti. Jeho hlavní předností, kromě výše uvedené ceny, je dostupnost tohoto prostředku pro většinu operačních systémů i počítačů. Velkou výhodou je skutečnost, že na oficiálních stránkách vývojového týmu, je možné získat zdrojový kód serveru Apache.

2.3.3. Ostatní

Na poli aplikačních serverů existuje mnoho různých řešení, která mají své výhody a samozřejmě také nevýhody. Za zmínku zajisté stojí Netscape Enterprise Server od společnosti Netscape Communications Corporation. Jedná se o oblíbený server, mezi jehož přednosti patří hlavně:

- Široké spektrum platforem.
- Dobrá dokumentace.
- Jednoduchá správa.

2.4. Statické internetové aplikace

Internetové aplikace (často označované jako stránky), lze z hlediska funkčnosti rozdělit na:

- *Statické*, jejichž obsah se nemění v závislosti na požadavcích uživatele a mají pouze informační charakter.
- *Dynamické*, které reagují na požadavky uživatele a podle jeho zadaní mění svůj vzhled a obsah. Jako příklad lze uvést internetové obchody nebo zásilkové služby.

2.4.1. HTML

Značkovací jazyk označovaný zkratkou HTML (HyperText Markup Language) je základním prostředkem pro publikování v prostředí internetu. Byl vyvinut ze značkovacího jazyka SGML (Standard Generalized Markup Language). Jedná se o značkovací jazyk, což znamená, že jazyk má definované základní značky, které mají určitý význam a říkají, jak má výsledný dokument vypadat. Např. pokud použijeme značku <HR>, bude do textu vložena vertikální linka. Lze si povšimnout, že značky jsou uzavírány do ostrých závorek <> a rozlišujeme značky párové, musejí být zakončeny a jejich význam je aplikován na text mezi nimi, např. text a značky nepárové, které nemusí obsahovat zakončovaní značku, např.
 je značkou pro odřádkování. Hlavní předností jazyka HTML je jeho jednoduchost. Standard jazyka stanovený konsorciem W3C (World Wide Web Consorcium), je ale velmi benevolentní k chybám ve značkování, což způsobuje značné problémy hlavně vývojovým týmům pracujících na interpretech (prohlížečích) internetových stránek. Z toho vyplývají problémy s přenositelností stránek, protože každý prohlížeč má jinou metodu na odstraňování chyb ve značkování a výsledkem jsou tudíž rozdílné interpretace.

2.4.2. XML a XHTML

Stejně jako jazyk HTML je i XML (eXtensible Markup Language) odvozen od předchůdce značkovacích jazyků SGML. XML nemá žádné značkovaní pevně dané a každou značku si může vytvořit uživatel podle svých představ a dát jí také odpovídající význam, což je hlavní rozdíl proti pevně stanovenému značkování HTML. XML je orientované na obsah dokumentu a ne na jeho vzhled, jak tomu

bylo u HTML. Všechny značky jsou definovány buď přímo v dokumentu nebo pomocí externí deklarace v souboru DTD (Dokument Type Definition). XML má také daleko přísnější kontrolu syntaxe, takže chyby zcela běžné v jazyce HTML jsou zde označeny a interpretace dokumentu neproběhne. To je velmi užitečné hlavně v poslední době, kdy se pro prohlížení obsahu internetu používají stále častěji jiná zařízení než PC, např. mobilní telefony a kapesní počítače. Pokud tedy uvažujeme o interpretaci stránek s chybami v syntaxi, pak bude třeba velmi objemných a systémově náročných interpretačních aplikací, což lze akceptovat u stolních počítačů, ale nelze je použít na výše zmiňované alternativní přístupy. Řešením je tedy přísná kontrola syntaxe, tak jak ji přináší XML.

Jazyk XHTML (eXtensible HyperText Markup Language) vychází z jazyka HTML, ale řídí se principy jazyka XML. Hlavní výhodou jazyka XML je jeho malá tolerance syntaktických chyb. Nevýhodou je naopak malá kompatibilita se stávajícími prohlížeči internetových stránek. S podporou XML se setkáme pouze u posledních verzí interpretačních programů. Výhody HTML a XML jsou spojeny do nového jazyka XHTML, který je zpětně kompatibilní se staršími prohlížeči, ale má pevně stanovenou syntaxi, která je kontrolována ještě před vlastním zpracováním stránky. Lze jej tedy použít i na vytvoření stránek, které budou zobrazitelné i na jiných periferiích než jsou stolní počítače.

2.4.3. CSS1 a CSS2

CSS (Cascading Style Sheet) Level 1 a Level 2 jsou prostředky pro grafickou úpravu stránek nebo aplikací. Podstatně rozšiřují základní formátovací značky jazyků HTML a XHTML. Lze například pomocí jednoduchého zápisu přiřadit font a barvu celému obsahu značky <BODY>. Příklad zápisu:

```
<HTML>
<BODY>
<STYLE type="TEXT/CSS">
        body { font-size: 1px;
                font-color: blue;}
</STYLE>
</BODY>
</HTML>
```

Veškerý text uzavřený mezi značkami <BODY> a </BODY>, pak bude velikosti jednoho pixelu a modré barvy.

Velkou výhodou tohoto přístupu je nadefinování způsobu zobrazení pro jednotlivé prvky a následně používání pouze těchto předdefinovaných stylů, což usnadňuje práci a zlepšuje přehlednost výsledného kódu. Velkou výhodou CSS, která se ukazuje hlavně v poslední době, je možnost navrhnout grafickou úpravu stránky, která bude přenositelná do různých prostředí. Lze si snadno přestavit, že ne každý má nastavenou stejnou velikost písma a další atributy, které pomáhají přizpůsobit grafický design systémových prostředků jednotlivým uživatelům. Bude tedy určitě chybou neakceptovat dané nastavení, což je právě případ předdefinovaných značek HTML. Použijeme-li CSS a vhodné jednotky, budou naše aplikace příjemné všem uživatelům.

Další velkou výhodou CSS je možnost připravovat dvě různé verze jedné stránky. Jedna bude určena pro prohlížení a druhá pro tisk, kde budou vynechány nepotřebné údaje a nebo obrázky.

CCS2 neboli CSS Level 2 je rozšíření stávajícího modelu o další vlastnosti pro zobrazovaní. Jeho nevýhodou je malá kompatibilita se staršími prohlížeči a tak lze předpokládat, že své uplatnění najde především v budoucnu.

2.5. Dynamické internetové aplikace

Doba, kdy internetové stránky sloužily pouze jako informační centra bez potřeby rychlých změn, je již dávno pryč. Dnes se moderní internetové sídlo neobejde bez kvalitního aplikačního serveru a nějakého druhu dynamického programování. Dynamické technologie lze rozdělit na technologie na straně klienta a na straně serveru.

Technologie na straně klienta má řadu nevýhod, jak bylo zmíněno v úvodu. Proto se pozornost obrací právě na technologie pracující na straně serveru. Těchto prostředků je celá řada a výběr podléhá celé řadě kritérií, ale obvykle se tyto metody kombinují.

2.5.1. PHP

Historie PHP (Personal Home Page) sahá do roku 1994, kdy Rasmus Lerdorf vypracoval první skripty, aby zjistil, kdo navštěvuje jeho stránky. Od té doby prošlo PHP velmi rychlým vývojem, až do dnešní doby, kdy je mu

věnována velká pozornost a vychází celá řada publikací, které se právě tímto skriptovacím jazykem zabývají. Dnes je PHP vyvíjeno jako Open Source (volně šiřitelný software s otevřeným kódem) a jeho poslední oficiální verze je 4. PHP (dnes je uváděna zkratka PHP Hypertext Preprocesor) je skriptovací jazyk zabudovaný na straně serveru (Viz obr. 2) s velmi podobnou syntaxí jakou má jazyk C/C++. Velkou výhodou PHP je jeho univerzálnost a kompatibilita s různými platformami jako jsou MS Windows, Unix či Linux a různými aplikačními servery jako třeba Apache nebo IIS. PHP pracuje uvnitř dokumentu HTML a podle proměnných, které reprezentují požadavky klienta, je tento dokument zpracován a jako výsledek je klientovi zaslána odpověď ve formátu dokumentu HTML. Kód jazyka PHP je do HTML dokumentu vložen pomocí speciálních značek, začátek uvozují „<?php“, pak následuje vlastní kód a konec je označen pomocí „?>“. Dá se rovněž použít několik alternativních zápisů, z nichž stojí za zmínku <% %>, který je kompatibilní s ASP, o kterém bude řeč v následující kapitole.[3]

2.5.2. ASP a ASP.NET

ASP (Active Server Pages) od společnosti Microsoft je také skriptovací jazyk na straně serveru a lze ho označit spolu s PHP za hlavní skriptovací stroj používaný na internetu. ASP má velmi podobný princip jako PHP, je také vloženo do stránky HTML a podle předaných proměnných od klienta je dokument zpracován a odeslán zpět klientovi. ASP má jinou syntaxi než PHP, která je do značné míry podobná MS Visual Basic a je do dokumentu vložena pomocí značek „<% %>“, mezi které je vložen vlastní kód aplikace. V poslední době došlo k rychlému vývoji tohoto skriptovacího nástroje a jako poslední verze je označována ASP.NET, která se do značné míry od svého předchůdce liší. ASP je velmi výkonný skriptovací nástroj hojně na internetu používaný, zejména pro náročnější aplikace a webová sídla komerčního charakteru. Největší nevýhodou proti PHP je cena produktu a závislost na systémech od společnosti Microsoft a s tím související platformě Intel.[3]

2.5.3. Java

Java je programovací jazyk vyvinutý společností SUN Microsystems. V tomto jazyce lze psát stránky JSP (Java Server Pages), které pak lze pomocí Java Servletu zpracovávat na straně serveru a tím rozšiřovat funkčnost webového

serveru. JavaServlet je program spouštěný na straně serveru, který obsluhuje požadavky klienta ve formě http a výsledky vrací klientovi ve stejné formě. Princip vložení kódu do dokumentu HTML je opět totožný jako u ASP a PHP, pokud na ně webový server narazí, pak se pokusí spustit servlet a výsledek vrátí prohlížeči.

Hlavní nevýhodou tohoto jazyka je jeho složitost, která často způsobuje velké problémy zejména začínajícím programátorům.[3]

2.5.4. JavaScript

JavaScript na straně serveru (Server-Side JavaScript, SSJS) od společnosti Netscape Communications Corporation lze považovat za konkurenta ASP na poli komerčních skriptovacích strojů. Jeho princip je opět totožný s výše jmenovanými jazyky, tedy kód je do stránky HTML vložen pomocí značek a je pak zpracován na straně serveru. Velkou výhodou SSJS je to, že používá JavaScript, což je standardní jazyk internetu. Má však dva závažné nedostatky, jako první, méně závažný, lze zmínit nutnost komprimace SSJS před jeho spuštěním a jako druhý, závažnější, je třeba jmenovat závislost na aplikačním serveru Enterprise Server od společnosti Netscape Communications Corporation, který svými parametry značně zaostává za MS IIS i Apache. To brání masivnějšímu použití tohoto jinak velmi zajímavého jazyka.[3]

2.5.5. CGI

Jedná se o nejběžnější internetovou technologii na straně serveru, která je dnes podporována drtivou většinou aplikačních serverů. CGI (Common Gateway Interface) může být napsán v libovolném jazyce, např. Perl. Pokud server přijme požadavek, je vytvořen celý nový proces, kdy CGI načte program a prostředí pro zpracování. Po ukončení běhu programu je výsledek přečten ze standardního výstupu a odeslán klientovi. Hlavní nevýhodou CGI je jeho neškálovatelnost, to znamená, že je nutné vytvořit pokaždé nový proces včetně přidělení paměti. Lze tedy předpokládat, že zpracování velkého množství požadavků bude značně zatěžovat server. S touto problematikou se různé aplikační servery vypořádávají pomocí podpůrných programů a knihoven s různou efektivitou.[3]

2.6. Nástroje využívané při realizaci SW projektu

2.6.1. MS Projekt pro plánování a sledování postupu projektu

MS Project je nástroj, který umožňuje plánování a sledování vývoje projektu, jak v časové, tak i úkolové rovině. Tento nástroj lze využít pro plánování všech typů projektu např. ze stavebnictví, vývoje nových výrobků nebo vývoje SW.

V rámci realizace tohoto projektu byl sestaven ve spolupráci se zástupci zadavatele projektový plán, který zahrnoval všechny projektové fáze (příprava, analýza, návrh, vývoj, testování, optimalizace,..)

Plán sestavený v MS Project je součástí přílohy (Viz příloha 1)

2.6.2. Case Studio pro funkční a datové modelování

Zadaný projekt si vyžádal důkladnou datovou a funkční analýzu. K těmto účelům byl použit program Case Studio, který umožňuje modely graficky znázorňovat a upravovat, což je velmi praktické a pohodlné. Výstupem jednotlivých modelů jsou skripty v jazyce SQL, které vytvářejí fyzickou strukturu databáze (Viz příloha 5). Protože byl použit databázový server od společnosti Microsoft, byl tento produkt výhodný i z hlediska kompatibility, která je u relačních databází do značné míry problematická. Program umožňuje zvolit cílové databázové prostředí, čímž je usnadněna přenositelnost modelu na jiné databáze.

2.6.3. PHP Code Pro

Jedná se o volně distribuovaný softwarový produkt, který slouží pro psaní vlastních skriptů v jazyce PHP. Za editor lze použít například standardní nástroj Poznámkový blok, který je součástí systému MS Windows XP, ale vývoj v takovémto editoru je obtížný. Program PHP Code Pro zlepšuje orientaci v kódu barevným rozlišením HTML a PHP kódu. Umožňuje otevření velkého počtu souborů, což je obvyklé při vývoji větší aplikace a nabízí také řadu jiných zajímavých funkcí, jako je třeba hromadná náhrada určité části kódu.

2.6.4. MS Query Analyzer

MS Query Analyzer (dále jen MSQA) je součásti produktu MS SQL Server 7.0 a slouží pro testování dotazů do databáze a spouštění skriptů napsaných v jazyce SQL. Při vývoji aplikace je třeba dotazy testovat na syntaktické chyby, což je efektivnější, pokud se provádějí přímo na databázovém stroji. MSQA nabízí

možnost spuštění velmi komplikovaných skriptů vytvořených pomocí modelu, které jsou barevně odlišeny podle funkčnosti a usnadňují programátorovi orientaci. Pro vytvoření celé databáze spolu se souvisejícími objekty je třeba velmi složitý skript (Viz příloha 5), ten je možné pomocí MSQA upravit, případně uložit do souboru pro pozdější použití. MSQA je také nepostradatelný při přenosu dat z testovacího severu na server provozní, kdy je na testovacím serveru vygenerován skript pro vytvoření veškerých objektů, následně je tento skript spuštěn na provozním serveru pomocí MSQA a data je pak možné bezpečně importovat do databáze.

3. Realizace SW projektu - návrh a vývoj databázové aplikace

3.1. Příprava projektu a definice jeho rozsahu

Velmi důležitou součástí vývoje softwarového produktu je jeho úvodní fáze, ve které dochází ke stanovení jeho rozsahu včetně specifikace problémů a požadavků. Přibližně polovina všech chyb, které vzniknou při vývoji aplikace, je způsobena právě nedostatečnou specifikací problému (udává se 56%, 26% chyby v návrhu, 7% chyb v programování a 11% na ostatní chyby). [4]

Zadavatel požadoval vytvoření databázové aplikace, která by odstranila stávající problémy s udržováním aktuálních kmenových dat, umožnila snadné vyhledávání údajů, sjednotila stávající evidenci a doplnila ji o číselníky pracovišť, funkcí a hospodářských středisek. Výsledný produkt musí být otevřený a snadno rozšiřitelný.

Druhým požadavkem bylo vytvoření aplikace, která by umožnila všem zaměstnancům, kteří mají přístup k výpočetní technice, snadné vyhledávání podle různých kritérií v grafickém režimu tak, aby, bylo srozumitelné a použitelné všemi uživateli. Následně vytvořit administrační modul, který umožní spravovat celou databázi s použitím uživatelsky příjemného grafického rozhraní.

Administrační modul by měl umožnit přidávávání administrátorů a celý modul by měl být dostatečně zabezpečen.

Aplikace byla navržena a vytvořena tak, aby bylo možné v případě předpokládaného rozšiřování požadavků snadno doplnit nové funkce.

Zadavatel vyžadoval využití maximálního možného počtu stávajících softwarových (dále jen SW) a hardwarových (dále jen HW) prostředků v zájmu ochrany již vynaložených investic.

Dalším cílem bylo snížení stávajících finančních a pracovních nákladů na vedení agendy o zaměstnancích s cílem minimalizovat počet osob, které budou vytíženy správou databáze.

3.1.1. Plán projektu

Plán projektu byl vytvořen v aplikaci MS Project, kde na základě standardní projektové šablony byly definovány jednotlivé fáze vývoje, včetně přiřazení zdrojů určených k jejich realizaci. Základní vývoj probíhal ve vývojovém prostředí, které nebylo provozováno u zadavatele, s následným testováním v testovacím prostředí, které odpovídalo provoznímu prostředí, ve kterém byla aplikace nasazena.

Cílem plánování projektu je minimalizování časových i finančních nákladů vývoje. Plán projektu je součástí přílohy 1. Základní pracovní silou byl Roman Špánek, jako programátor, grafický designér a programátor databázových struktur a Ing. Pavel Hujer zaměstnanec zadavatele, jako odborný poradce, konzultant a oponent. Veškeré činnosti jsou rozděleny mezi výše uvedené zdroje s výjimkou zkušebního provozu aplikace, kterého se účastnili i další pracovníci zadavatele. Celková plánovaná doba realizace projektu odhadnutá na základě jeho rozsahu, by měla být zhruba sto třicet tři dní, tedy více než čtyři měsíce.

3.1.2. Souhrn problémů a požadavků

Zadavatel sestavil dokument „Souhrn problémů a požadavků“, který je součástí přílohy 2. Dokument obsahuje vytipování základních problémů, které měl projekt řešit a definování požadavků na nový systém.

3.1.3. Podklady nutné pro realizaci projektu

Základním dokumentem byl stávající telefonní seznam v tištěné a elektronické podobě. Dokument sloužil jako hlavní stavební kámen pro realizaci projektu. Během návrhu zadavatel stanovil, které položky je nutné zachovat i v nové aplikaci a naopak, které lze vynechat. Tímto způsobem byl v průběhu návrhu projektu minimalizován počet vyžadovaných položek, které bylo nutné importovat do databáze pomocí konverzních a importních rutin. Výsledkem byl upravený dokument, který obsahoval pouze nezbytně nutná data bez redundancí, které by později mohly způsobit komplikace při programování vlastní aplikace.

Na začátku vývoje aplikace byly zadavatelem stanoveny základní stavební prvky celé aplikace. Jednalo se především o sadu číselníků. Aby bylo možné do databáze vložit co možná nejaktuálnější data, byla tištěná a elektronická podoba těchto číselníků dodána krátce po dokončení datového modelu. Datový model byl vytvořen na základě dokumentů, které neobsahovaly aktuální informace, ale měly již konečnou strukturu dat.

3.2. Analýza

Problematika současných informačních systémů (dále již jen IS) je natolik složitá, že vývoji každé aplikace musí předcházet detailní analýza. Analýzu lze dle přístupu rozdělit na:

- *Funkční analýzu*, systém je modelován jako množina vzájemně působících funkcí. Nejčastějším vyjádřením jsou DeMarcovy diagramy datových toků DFD (Data Flow Diagram). Systém je pak vyjádřen pomocí procesů spojených datovými toky.
- *Datovou analýzu*, která je reálným popisem aplikace. Struktura dat v reálném světě je obvykle stabilní (např. skoro každý člověk má jméno a příjmení), naproti tomu struktura dat v počítači je závislá na platformě a jejich stabilita není zaručena. Hlavní úlohy datového modelování jsou:
  - Lepší pochopení složení aplikace.
  - Lepší definice vztahů mezi aplikacemi.
  - Zlepšení komunikace mezi uživatelem a analytikem.
  - Zlepšení komunikace mezi analytikem a programátorem.[5]

Výsledkem analýzy je pak fyzický datový model a také SQL skript, který lze využít pro vytvoření fyzické implementace datového modelu v prostředí DB serveru.

3.2.1. Funkční analýza

Hlavními úkoly funkční analýzy jsou:

- Identifikace systémových funkcí.
- Identifikace událostí.

- Definice transakcí.
- Popis transakcí.
- Identifikace dotazových transakcí.
- Identifikace obecných transakcí.

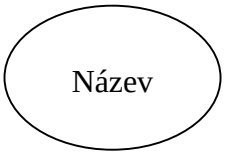
Hlavním prostředkem pro splnění výše uvedených úkolů je DFD.

Funkční model popisuje funkci systému, práci s jednotlivými bloky, aniž by popisoval nebo studoval vnitřní strukturu dat. Výsledkem je sada funkcí, které budou zadaný problém realizovat.

3.2.1.1. Diagram toku dat

Diagram toku dat (dále jen DFD) je grafický prostředek zobrazující funkční model systému. DFD vyjadřuje tok dat a jejich transformace, obsahuje popis funkcí, které jsou pro danou transformaci třeba. Nejedná se však o časovou závislost, ale jen o grafické znázornění závislostí mezi jednotlivými kroky, které probíhají v systému. DFD je složen ze čtyř základních stavebních prvků:

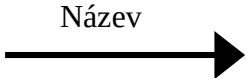
- *Proces* – je místem, kde je provedena transformace vstupu



Obr. 5 Symbol procesu

na výstup, tedy kde je provedena určitá funkce, která vyprodukuje výstupní data.

- *Datový tok* – vyjadřuje přesun dat nebo informací od jedné části



Obr. 6 Datový tok

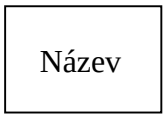
systému ke druhé. Datový tok musí být pojmenovaný a jeho jméno musí jasně reprezentovat obsah dat.

- *Datastore (úložiště dat)* – je dočasné úložiště dat, musí mít

Obr. 7 Datastore

minimálně dva připojené datové toky a to jeden směrem ven a druhý směrem dovnitř.

- *Terminátor* – označuje místo externího napojení na systém,



Obr. 8 Terminátor

napojením může být například člověk, oddělení v oragnizaci, atd.

DFD je obvykle vytvořen ve třech úrovních. První úroveň (Viz příloha 3) vyjadřuje, kde bude hranice celého systému a jak bude komunikovat s okolním světem. Další úrovně postupně detailněji popisují funkčnost jednotlivých procesů na výšších úrovních. Jedná se tedy o tzv. top - down rozklad, tedy rozklad zhora dolů.[5]

3.2.2. Datová analýza

Datová analýza má za úkol popsat vztahy mezi daty, která v systému existují, a jejich strukturou. Pomocí vhodného grafického znázornění zobrazit násobnost vztahů, datovou strukturu jednotlivých entit společně s jejich názvy.

3.2.2.1. ERD model

Diagram entit a vztahů (ERD - Entity Relationship Diagram) je grafický nástroj používaný k vyjádření struktury entit a vztahů mezi nimi. EDR model je tvořen pomocí tří základních prvků:

- *ENTITA* – je objekt, o kterém chce společnost uchovávat data. Grafický symbol je obdélník.
- *ATRIBUT* – je prvek, který entitu dále charakterizuje.

- *VZTAH* – vyjadřuje spojení entit, tedy jak jsou mezi sebou propojeny, na základě logického propojení zkoumaných objektů.
  - *KARDINALITA VZTAHU* – vyjadřuje mocnost vztahu entit. Rozlišujme tři základní mocnosti:
  - 1:1 , vyjadřuje skutečnost, že obě entity jsou ve vztahu zastoupeny pouze jednou.
  - 1:N , vyjadřuje skutečnost, že k jednomu výskytu entity A existuje více výskytů entity B. Jedná se o nejčastější případ vztahu dvou entit.
  - M:N , vyjadřuje skutečnost, že k jednomu výskytu entity A existuje více výskytů entity B a naopak. Tento vztah není realizovatelný ve fyzickém návrhu databáze a je třeba jej nahradit vazební entitou, která vztah převede na dva vztahy 1:N.
  - *VOLITELNOST VZTAHU* – vyjadřuje zda je účast entit ve vztahu nutná nebo zda je volitelná a nemusí existovat.

Datový model vyjadřuje vztah mezi typy entit bez popisu funkcí, které tyto data vytvářejí nebo je používají.[5]

3.2.3. Výsledky analýzy

3.2.3.1. Diagram datových toků

Diagram datových toků je vytvořen ve dvou základních úrovních.

První úroveň znázorňuje napojení systému na vnější prvky, jak je zobrazeno v příloze

Terminátory jsou uživatel, který bude systém používat přes uživatelskou aplikaci a administrátor, který bude přistupovat k aplikaci po ověření požadovaných údajů a bude mu umožněno spravovat celý systém.

Druhá úroveň popisuje vnitřní funkci systému s podrobnějším pohledem na jednotlivé funkční bloky. DFD druhé úrovně je součástí přílohy 3.

Uživatel má k dispozici tři základní funkční bloky, které mu umožňují vyhledávání v registru zaměstnanců, tisk telefonního seznamu a nahlášení změn vlastních kmenových údajů. Všechny tři procesy jsou napojeny na datastore

Registr zaměstnanců a podle požadavků uživatele zpřístupňují požadovaná data. Proces *1.2 Nahlášení změny* připraví uživateli data k upravení podle zadaných kriterií. Po provedení změn a odeslání jsou veškerá změněná data doplněna o zbývající původní hodnoty a uložena v datastoru *Nahlášené změny.*

Administrátor má možnost provádět úpravy veškerých dat pomocí procesu *1.3 Úprava databáze*, který data vybírá a ukládá do datastoru *Registr zaměstnanců*, kde jsou okamžitě k dispozici uživatelům. Úpravu administrátorů umožňuje proces *1.5 Úprava administrátorů*, který má přístup k datům jednotlivých administrátorů uložených v datastoru *Registr administrátorů*.

Nahlášené změny jsou zpřístupněny procesem *1.1 Odsouhlasení změny*, který, pokud jsou změny potvrzeny, změny uloží do *Registru zaměstnanců* a okamžitě je tím zpřístupní ostatním uživatelům. Umožňuje také změny upravit nebo smazat podle požadavků administrátora.

Posledním procesem, který umožňuje upravovat menu jednotlivých aplikací je proces *1.4 Úprava menu*. Podle zvolené aplikace jsou administrátorovi distribuována data menu uživatelské nebo administrátorské aplikace. Po upravení jsou data zapsána zpět do příslušných registrů a použita ke generování aktualizovaného menu aplikace.

3.2.3.2. Logický datový model

Logický datový model je reprezentován E-R diagramem, který je součástí příloh jako příloha 4.

Hlavní entitou je *xTelSeznam*, který je vytvořen podle struktury původního telefonního seznamu, ze kterého byly odstraněny přebytečné údaje. Všechny číselníky byly vytvořeny jako samostatné entity *xCisFcn, xHS a firma*, které jsou propojeny k entitě *xTelSeznam* pomocí vazeb 1:N.

Entita *xZmeny* slouží k ukládání upravených údajů včetně záznamů o administrátorovi, který provedl odsouhlasení změn. Všechny informace o administrátorech jsou uloženy pomocí entity *xAmin*, která je propojena s entitou *xZmeny* vazbou 1:N.

K uložení telefonních čísel pohotovostních mobilních telefonů slouží entita *xPohotMobil*, která není propojena na žádnou jinou entitu.

Pracoviště jsou zastoupena entitou *xPracoviste*, která obsahuje pouze název. Pomocí relační vazby 1:N je napojena na entitu *xPolozkyPrac*, která

obsahuje všechny údaje o jednotlivých pracovištích. Je-li uložení dat o pracovištích realizováno pomocí toho modelu, lze velmi snadno přidávat nová pracoviště.

Entita *xMenu* a *xMenuAdmin* obsahuje definici jednotlivých tlačítek v navigačním menu, např. *Vyhledávání*. Data potřebná k vytvoření jednotlivých položek menu jsou uložena v entitě *xPolozkyMenu* resp. *xPolozkyMenuAdmin* a jsou propojena vazbou N:1 s entitou *xMenu* resp. *xMenuAdmin.*

3.2.3.3. Fyzický datový model

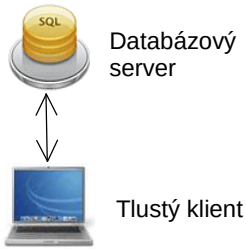
Výsledný skript v jazyce SQL, který vytváří veškeré databázové objekty je součástí přílohy 5.

Skript vytvoří tabulky, včetně nastavení oprávnění k jednotlivým objektům a nastaví mezi nimi relační vazby.

3.3. Návrh

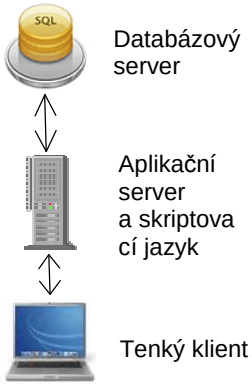
3.3.1. Návrh architektury systému

Aplikace je určena velkému počtu uživatelů, jedná se asi o 700 uživatelů, kteří budou aplikaci využívat. V úvahu připadala dvě řešení, jedno založené na dvouvrstvé architektuře, obrázek 9 a druhé na architektuře třívrstvé, obrázek 10.



Obr. 9 Dvouvrstvá architektura

Zatímco dvouvrstvá architektura odděluje databázový server a klientskou aplikaci, kterou je tlustý klient, třívrstvá je dělena na vrstvu databázového serveru, aplikačního serveru doplněného vhodným skriptovacím jazykem a tenkého klienta, který má korektně zobrazovat data odeslaná aplikačním serverem.



Obr. 10 Třívrstvá architektura

U třívrstvé architektury jsou všechny skripty aplikace uloženy na aplikačním serveru, kde je lze snadno aktualizovat, aniž by bylo nutné zasahovat do nastavení u uživatele. Naproti tomu dvouvrstvá architektura má všechen programový kód implementován v podobě klientské aplikace tzv. „tlustého klienta“ přímo na PC uživatele. Je neakceptovatelné, aby byla aplikace instalována u každého uživatele

individuálně a to i při pozdějších změnách. Proto byla zvolena třívrstvá architektura, která nevyžaduje instalaci nových verzí aplikace na klientských PC a současně také umožňuje rozdělit zátěž systému mezi různé výpočetní prostředky, což je vhodné vzhledem k přenositelnosti a také ke zvýšení bezpečnosti a výkonu aplikace.

3.3.2. Návrh vhodného databázového serveru

Základ aplikace tvoří poměrně rozsáhlá databáze informací o zaměstnancích a dalších potřebných údajů, včetně několika číselníků. Pro tento účel a následná napojení databáze na hlavní registr zaměstnanců byl potřebný výkonný databázový stroj s kvalitním zabezpečovacím mechanismem. Jako volba byla uvedena možnost využít stávající databázový stroj, který již řadu měsíců dobře fungoval u zadavatele, Microsoft SQL Server 7.0. Jedná se o kvalitní, komerční SW, který splňuje všechny požadavky kladené na DB server. Další možností bylo použití volně dostupného serveru MySQL, který byl i součástí balíčku PHP Triad (viz kapitola 3.3.4). Je to poměrně výkonný stroj, který však není vhodný pro větší aplikace, neboť postrádá některé funkce, jako třeba transakce, které jsou pro zadaný projekt nutností. Poslední volbou byl databázový stroj Oracle. Jeho hlavní nevýhodou je cena, která se šplhá řádově do stovek tisíc korun a je tedy pro zadavatele naprosto nepřijatelná.

Konečnou volbou byl systém MS SQL Server 7.0, který má zadavatel již zakoupen a tak nedošlo v rámci tohoto projektu k navýšení nákladů na provoz a pořízení aplikace.

3.3.3. Návrh vhodného aplikačního serveru

Hlavním požadavkem na aplikační server je robustnost a stabilita. Těmto požadavkům však dnes vyhovuje drtivá většina aplikačních serverů a to jak komerčních, tak i volně dostupných. Dalším požadavkem ze strany zadavatele byla kompatibilita se stávající vnitropodnikovou sítí, která je provozována na aplikačním serveru IIS 5.0 od společnosti Microsoft. První volbou bylo využít stávající server. Druhou možnou volbou byl server Apache a to zejména pro nativní podporu skriptovacího jazyka PHP a jeho cenu.

Konečnou volbou byl server MS IIS, který již byl nainstalován a spolehlivě zaběhnut u zadavatele, což bylo důležité vzhledem k termínu dokončení celého

projektu. Protože vývoj aplikace probíhal v jiném prostředí, než bylo testovací a provozní prostředí zadavatele, byl pro vývojové prostředí zvolen aplikační server Apache a to především pro jeho cenovou dostupnost. Výsledné skripty však nejsou na aplikačním serveru závislé a tak tato metoda vývoje neohrozila výslednou funkčnost aplikace a naopak ověřila kompatibilitu aplikace i s tímto rozšířeným aplikačním serverem.

3.3.4. Návrh skriptovacího jazyka a vývojového prostředí

Volba skriptovacího jazyka byla závislá na prostředí zadavatele, kde byl již používán jazyk PHP. Vzhledem k použití aplikačního serveru MS IIS byl další možnou volbou jazyk ASP. Možnosti obou zmíněných jazyků jsou do jisté míry totožné, alespoň tak to bývá publikováno ve zprávách, které se touto problematikou zabývají. Velkou výhodou jazyka PHP je jeho syntaktická podobnost jazyku C/C++, který je velmi rozšířený v povědomí programátorů.

Konečnou volbou byl jazyk PHP, pro který mluvila hlavně jeho funkčnost a použití ve stávajícím prostředí zadavatele.

Rozdíl vývojového prostředí vůči prostředí, kde měla výsledná aplikace běžet, byl ve zvoleném aplikačním serveru, jak je zmíněno výše. Nastavení vývojového prostředí probíhalo ve dvou krocích, prvním bylo nainstalování aplikačního serveru a následné nastavení skriptovacího jazyka. Jako vhodná možnost byla zvolena varianta volně dostupná na internetu pod názvem PHP Triad. Jedná se o balík, který obsahuje aplikační server Apache, skriptovací jazyk PHP a databázový server MySQL. Balík je po nainstalování nastaven pro práci na lokálním počítači a tak nebylo nutné provádět velké změny v konfiguraci. Jedinou podstatnou změnou bylo povolení používání session proměnných, o kterých bude zmínka v kapitole 3.4.4.2..

3.3.5. Návrh webového prohlížeče pro koncového uživatele

Ve světě webových prohlížečů hraje, bez jakýchkoli pochybností, prim produkt Internet Explorer od společnosti Microsoft. Existuje celá řada alternativních možností, ze kterých lze jmenovat zejména volně dostupný prohlížeč Mozilla. Kvalitativně jsou si tyto prohlížeče velmi blízko a nelze tedy volit podle možností, které tyto prostředky nabízejí. Jak je zmíněno v úvodu (kapitola 1.1.4), hlavní problém dnešního internetu je kompatibilita jednotlivých prohlížečů,

která se projevuje v různém zobrazení výsledných stránek napsaných v jazyce HTML.

Volba pro zadaný projekt byla vcelku jednoduchá, protože ve stávajícím provozu měl stoprocentní zastoupení MS Internet Explorer, což bylo výhodné zejména kvůli optimalizaci, která je v tomto případě jednodušší. Jako základní verze MS Internet Explorer byla zvolena verze 5.0, která byla nainstalována na drtivé většině pracovišť.

3.4. Vývoj

Vývoj aplikace neprobíhal v cílovém provozním prostředí a to hlavně z důvodu omezeného prostoru u zadavatele. Proto bylo nutné vytvořit podmínky pro vývoj aplikace na tzv. vývojovém pracovišti a v prostředí zadavatele připravit testovací a provozní prostředí určené pro uživatelské a integrační testy.

3.4.1. Instalace a nastavení vývojového prostředí

Jako vývojové pracoviště byl zvolen počítač platformy Intel s operačním systémem MS Windows XP Professional, s aplikačním serverem Apache, skriptovacím strojem PHP a databázovou platformou MS SQL Server 7.0. Zásadní rozdíl vůči cílovému provozu byl ve zvoleném aplikačním serveru, který však, jak již bylo zmíněno, nemá vliv na kompatibilitu výsledných skriptů. Druhým rozdílem je použitý operační systém. Cílové prostředí pracuje s operačním systémem (dále jen OS) MS Windows 2000 Professional, který je však předchůdcem výše zmíněného OS MS Windows XP, se kterým má celou řadu společných rysů a lze je tedy označit za kompatibilní ve smyslu naší aplikace.

3.4.1.1. Nastavení MS SQL Serveru 7.0

Na vývojovém prostředí bylo nutné nainstalovat databázový server od společnosti Microsoft. Pro vývoj byla zadavatelem poskytnuta zkušební verze, která je plně funkční, ale časově omezená. Instalace je prováděna z instalačního CDROM a probíhá pomocí průvodce instalací, který usnadňuje orientaci v jednotlivých krocích. Hlavním bodem instalace je nastavení kódovací stránky a jazyka pro vyhledávání. Tato část je velmi důležitá, protože po nainstalování produktu již jazykové vlastnosti nelze měnit, aniž by došlo ke ztrátě dat. Volené parametry jsou s ohledem na nastavení databázového stroje u zadavatele:

- Character set: 1250 Central European

- Sort order: Dictionary sort order, case insensitive
- Unicode collation: Czech

V další části je zvolen adresář pro umístění souborů potřebných pro chod aplikace, který byl ponechán na přednastavené hodnotě „C:\MSSQL“. Dalším krokem je vytvoření uživatelského účtu, včetně nastavení hesla a volby typu autorizace. Lze využít autorizaci SQL serverem, autorizaci systémem Windows nebo kombinaci obou metod. Po instalaci je k dispozici několik nástrojů, které slouží ke správě databáze:

- *Query Analyzer* (dále jen QA), který slouží pro spouštění SQL dotazů zadávaných ručně, což je vhodné pro testování správnosti dotazu při vývoji, nebo uložených v souboru s příponou *.SQL. Jedná se o standardní textové soubory, které lze snadno editovat, buď pomocí QA nebo externího textového editoru. QA rovněž barevně rozlišuje příkazy, což velmi usnadňuje orientaci (Viz příloha 6). Ručně napsaný kód lze rovněž uložit a použít později, což je vhodné pro přenos dat mezi fyzicky oddělenými počítači.
- *Service Manager* je ulita sloužící pro spouštění, pozastavování

Obr. 11 Správce služeb

nebo ukončení běhu celého databázového stroje. Jedná se o velmi jednoduchou ulitu (Viz obr. 11), kterou lze nastavit tak, aby spouštěla server automaticky po startu systému. Stejné operace lze provádět i pomocí Správce služeb, který je součástí systému, ale orientace v něm je složitější.

- *Enterprise Manager* (dále již jen EM) je komplexní nástroj, který je hlavním pomocníkem při správě databáze (Viz příloha 6). Při spuštění EM je provedena autorizace uživatele a následně je

v hlavním okně aplikace možnost vybrat, který z běžících serverů bude spravovat. EM umožňuje vytvářet databáze, upravovat přístupová oprávnění k nim, vytvářet tabulky, importovat a exportovat data, spravovat vytvořené spouště, uživatelské datové typy, navrhovat vazby mezi tabulkami v grafickém editoru a mnoho dalších funkcí.

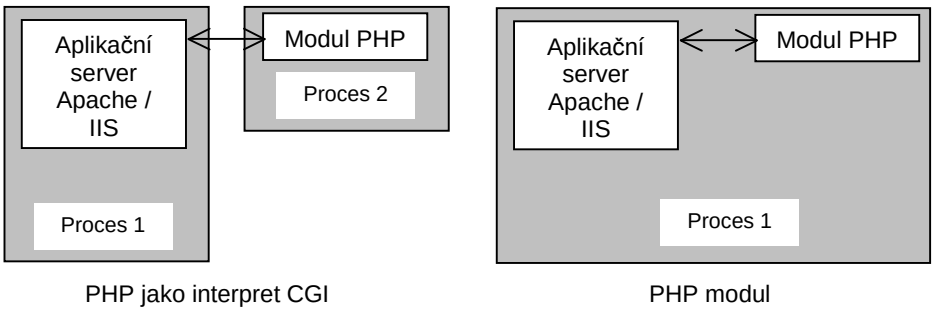
3.4.1.2. Nastavení aplikačního serveru APACHE

Aplikační server APACHE, byl nainstalován v balíčku PHP Triad a jako takový nepotřeboval žádné úpravy v nastavení.

3.4.1.3. Nastavení PHP

PHP může být zkompilováno jako samostatný interpret CGI nebo jako modul Apache (Viz obr. 12).

Pokud je PHP zkompilováno jako interpret CGI je pro něj pokaždé vytvořena nová instance, což má obvykle za následek snížení výkonu. Proto je výhodnější zkompilovat PHP jako modul Apache. PHP pak běží ve stejném adresném prostoru jako samotný server, což je nejen rychlejší, ale nabízí to i vyšší úroveň zabezpečení. Některé funkce jsou dostupné pouze tehdy, pokud je PHP zkompilováno jako modul. Další výhodou modulu je větší odolnost vůči



Obr. 12 Zkompilování modulu PHP

chybně napsaným skriptům, které by mohly způsobit havarijní stav serveru.

Pokud je PHP instalováno pod operačním systémem Windows, je vhodné použít distribuci v podobě binárních (již zkompilovaných) souborů. Po nainstalování souborů potřebných k běhu PHP, je nutné vytvořit konfigurační soubor php.ini a zkopírovat jej do systémového adresáře (např. c:\windows\). Následně je nutné upravit konfigurační soubor php.ini:

- session.save_path=c:\apache\tmp; která určuje adresář kam se budou ukládat proměnné session. Je nutné, aby adresář (tento případ c:\apache\tmp) existoval a byl do něj umožněn zápis.[3]

3.4.1.4. Nastavení ODBC

Rozhraní Open DataBase Connectivity (dále již jen ODBC) slouží jako článek spojující klientské aplikace a databázový server. Protože se jedná o unifikované rozhraní, klient k němu přistupuje způsobem, který nezávisí na použitém databázovém serveru. Lze tedy snadno využít skripty, které používaly databázový server MS SQL, pro systém Oracle bez větších úprav. Hlavní nevýhodou ODBC proti nativnímu spojení (spojení je navazováno a udržováno pomocí ovladačů, které jsou součástí aplikačního severu) je rychlost, která je např. pro produkt Oracle až desetkrát nižší [5]. ODBC je standardním rozhraním, které poskytují systémové zdroje a není tedy nutná jeho instalace.

Průvodce nastavením rozhraní ODBC je uveden v příloze 7.

3.4.2. Implementace fyzického datového modelu

Výsledkem analýzy je fyzický datový model, který reprezentuje model dat, tak jak bude implementován v databázovém systému. Pro implementaci nabízí MS SQL Server dvě základní možnosti:

- *Příkazová řádka*, zadávání příkazů do příkazové řádky spuštěné na serveru SQL, nebo pomocí Query Analyzeru. Tato volba je vhodná zejména při zakládání velkého množství databází, tabulek, pohledů, indexů atd.. Protože nabízí možnost uložení hotového a otestovaného skriptu do souboru a jeho následné opakované použití.
- *Enterprise manager,* grafické prostředí pro správu databázových serverů. Jeho předností je snadná orientace a používání i začínajícími administrátory bez znalostí jednotlivých příkazů. Hlavní nevýhodou je pracnost při vytváření většího množství objektů, tedy zejména tabulek, u kterých se tak zvyšuje časová náročnost jejich zakládání.

Před implementací bylo nutné zkonzultovat výsledky analýzy a fyzický datový model se zadavatelem, aby nedošlo ke změnám modelu v průběhu vytváření aplikačních skriptů, protože případná změna by ovlivnila funkčnost velkého množství skriptů a jejich následná oprava a testování by značně prodloužila a prodražila vývoj aplikace.

3.4.2.1. Vytvoření databáze

Pro vytvoření nové databáze je nutné mít oprávnění typu Administrátor. Prvním způsobem je vytvoření databáze pomocí příkazového řádku. Syntaxe příkazu je následující:

```
CREATE DATABASE jméno_databáze
 [ON
{[PRIMARY] (NAME=logické_jméno_souboru,
  FILENAME=´jméno_souboru_OS´
[,SIZE=velikost]
[,MAXSIZE=maximální_velikost]
[,FILEGROWTH=přírůstek_zvětšení])
} [….n]
]
[LOG ON
      { (NAME=logické_jméno_souboru,
         FILENAME=´Jméno_souboru_OS´
        [,SIZE=velikost])
      } [….n]
]
[FOR RESTORE]
```

kde význam jednotlivých parametrů příkazu je následující:

- *PRIMARY*. Volba určující soubory v primární skupině souborů. Jedná se o skupinu souborů, která obsahuje všechny systémové tabulky nové databáze. Není-li zadáno, pak je jako primární soubor zvolen první soubor uvedený v příkazu.
- *FILENAME*. Volba nastavuje cestu a jméno souboru podle operačního systému.

- *SIZE*. Udává velikost souboru v megabajtech ([MB]) nebo v kilobajtech (přípona [KB]). Její minimální hodnota je 512 KB a pokud není zadána je nastavena na stejnou hodnotu jakou má velikost souboru pro databázi MODEL.
- *MAXSIZE*. Určuje maximální velikost souboru opět v MB nebo v KB. Pokud není uvedena, DB soubor se může zvětšovat až do zaplnění disku.
- *FILEGROWTH*. SQL server bude automaticky zvětšovat soubor, pokud nebude stávající velikost stačit. Volba FILEGROWTH určuje o kolik bude soubor zvětšen. Hodnotu lze zadávat v MB, KB nebo %. Pokud volba není uvedena bude server implicitně zvětšovat soubor po 10% celkové velikosti.

Druhou možností je využití EM, kde lze stejné operace provádět v grafickém režimu (Viz příloha 6).[6]

3.4.2.2. Vytvoření tabulek

K vytvoření tabulek pomocí příkazové řádky je určen příkaz CREATE TABLE, který má následují syntaxi (neúplná):

```
CREATE TABLE jméno_tabulky
(jméno_sloupce datový_typ IDENTITY [(počátek,přírůstek)]
NEVIND() [ NULL | NOT NULL ]
[ ….n])
```

kde:

- *Datový_typ*. Určuje jaký datový typ je aplikován na vytvářený atribut.
- *INDENTITY*. Pokud je uveden, pak je vytvořen atribut identity. Sloupec má systémem generovaný obsah a slouží jako jedinečný identifikátor. Je-li použit atribut Identity, pak datovým typem musí být celočíselný typ a musí být uveden přepínač NOT NULL.

- *Počátek*. Určuje od které hodnoty se bude automaticky přičítat hodnota *přírůstek* k identifikačnímu sloupci.
- *NEVIND ()*. Vypočítá globálně jedinečný 16 bajtový identifikátor GUID (Global Unique Identifier). Datovým typem musí být *uniquidentifie*
- *[NULL | NOT NULL]*. Určuje zda je možné použít prázdný znak pro naplnění atributu, pokud nebude zadán.

Alternativní možností je zakládání tabulek pomocí EM v grafickém prostředí. Tato možnost je vhodná jen pro první vytvoření tabulek pro účely testování a vývoje, jinak je vhodnější využít skript, který byl vytvořen při analýze.

Při vývoji aplikace byly první tabulky vytvořeny ručně pomocí EM pro účely testování exportních modulů. V dalším průběhu byly vygenerovány kompletní skripty pro vytvoření všech databázových objektů, včetně tabulek. Následná implementace do provozního prostředí byla prováděna pomocí těchto skriptů (Viz příloha 5).[6]

Název všech tabulek je uvozen prefixem „x“, protože každá databáze obsahuje řadu systémových tabulek, které obvykle začínají písmenem „s“. Pokud použijeme výše uvedenou konvenci při vytváření názvů tabulek, bude snazší orientace v EM, který implicitně setřídí tabulky podle jejich názvu. Tato konvence je doporučená z praxe.

3.4.2.3. Vytvoření klíčových položek a indexů

Klíčové položky je možné vytvořit při vytváření tabulek nebo pomocí EM po jejich vytvoření. Protože všechny vytvořené tabulky, s výjimkou tabulky xCisFcn, nemají jedinečné identifikátory, jsou jako klíčové položky zvoleny identifikátory typu *identity*. U tabulky xCisFcn je jako klíčová položka zvolen atribut *Kod*, protože je ze své povahy dostatečným unikátním identifikátorem. Ostatní tabulky obsahují atribut ID_*, který je po vytvoření tabulky označen za klíčový pomocí příkazu:

PRIMARY KEY (jméno_sloupce),

nebo pomocí EM.

Pro vytvoření indexů slouží příkaz:

CREATE [UNIQUE] [CLUSTERED] [NONCLUSTERED]

INDEX jméno_indexu ON tabulka (sloupec[,….n])
[WITH
[PAD_INDEX]
[[,] FILLFACTOR=faktor_zaplnění]
[[,] IGNORE_DUP_KEY]
[[,] DROP_EXISTING]
[[,] STATISTICS_NORECOMPUTE]
]
[ON skupina_souborů]
kde:

- PAD_INDEX, slouží k nastavení volného prostoru indexových stránek v nelistové úrovni.
- FILLFACTOR, volba optimalizuje rychlost příkazů INSERT a UPDATE v tabulkách s indexy.[6]

Protože se u cílové aplikace nepředpokládá nárůst položek v číselnících nebo tabulce zaměstnanců nad hodnoty (bývá uváděna hodnota v tisících záznamech), u kterých by indexování urychlilo vyhledávání, nebyly indexy zavedeny. Byly pouze připraveny sloupce v návrhu databáze, nad kterými by bylo vhodné vytvořit indexy, pokud dojde k nárůstu položek nad stanovenou hranici. Jsou to tyto položky:

Tabulka	Sloupec
xTelSeznam	Prijmeni
xTelSeznam	Tel
xTelSeznam	Funkce
xTelSeznam	Firma
xTelSeznam	Ext
xTelSeznam	HS

3.4.2.4. Vytvoření relačních vztahů

K vytvoření relačních vztahů byl použit výhradně nástroj EM, který umožňuje vytvářet relační vztahy nad existujícími tabulkami a doplňuje potřebné informace do všech tabulek automaticky. Při vytvoření vztahu je třeba určit typ

omezení, které má daný vztah reprezentovat. Přehled všech možných typů omezení [6]:

Typ integrity	Typ omezení	Popis
Doménová	DEFAULT	Určuje hodnotu sloupce po provedení příkazu INSERT, pokud nebyla zadána hodnota
	CHECK	Definuje pravidlo pro určení platnosti dat ve sloupci
	FOREIGN KEY	Hodnoty ve sloupci musí odpovídat hodnotám v klíčových sloupcích tabulky, na kterou je odkazováno
Entity	PRIMARY KEY	Jednoznačná identifikace řádku, pokud by se uživatel snažil zadat stejný řádek vícekrát, nebude to povoleno.
	UNIQUE	Zabraňuje opakovaným hodnotám v neprimárním klíči.
Referenční	FOREIGN KEY	Definuje sloupec nebo kombinaci sloupců, které musí mít odpovídající hodnotu primárního klíče v odkazované nebo stejné tabulce
Uživatelem definovaná	CHECK	Popisuje pravidlo stanovení platnosti hodnot ve sloupci

3.4.2.5. Vytvoření uživatelských účtů, rolí a oprávnění k databázovým objektům

K vytvoření rolí a uživatelských účtů byl použit nástroj EM. MS SQL server umožňuje vytvořit systémové a databázové role. Systémové role jsou určeny pro správce systému a databázové role jsou používané pro oprávnění k jednotlivým databázovým objektům.

Pro potřebu aplikace byli vytvořeny dva přihlašovací účty:

- *verejny* – slouží pro přístup uživatelů k vyhledávacímu modulu a k modulu pro nahlašovaní změn. Oprávnění k jednotlivým tabulkám databáze TS jsou následující:

Tabulka	Oprávnění
xZmeny	**INSERT**
Ostatní	**SELECT**

- *administrator* – určený pro správce systému. Má přidělena všechna oprávnění k tabulkám databáze TS.

Protože se jedná o intranetovou aplikaci, nejsou potřeba další účty. Všichni uživatelé se přihlašují do databáze podle modulu ke kterému přistupují, jedním z výše uvedených účtů.

Jako systémový účet, který slouží administrátorovi databázového serveru, byl využit stávající účet administrátora, který využívá pověřený pracovník zadavatele.

3.4.3. Konverze a migrace dat

Stávající data, která byla dodána zadavatelem, byla uložena ve formátu sešitu aplikace MS Excel. MS SQL Server má zabudovánu podporu pro export a import dat v podobě balíčků DTS (Data Trasformation Services), které umožňují transport dat z různých formátů přímo do zvolené databáze. V průběhu importu dat je třeba postupovat v následujících krocích:

- Nastavení zdroje a cíle dat.
- Kopírování, dotaz nebo převod.
- Formátování a přenos.
- Uložení, naplánování a replikace.

Balíčky DTS lze používat v grafickém prostředí aplikace EM nebo lze využít vhodný příkaz příkazové řádky.

Pro převod sešitu MS Excel pomocí balíčku DTS bylo třeba nejprve odstranit veškerou grafickou úpravu, jako jsou především nadpisy. Další úpravou stávajících dat bylo odstranění sloupců dat, které již nebylo vhodné dále uchovávat. Jednalo se například o odstranění sloupců „RS“ a „ZL“. Protože sešit obsahoval více stránek s různým informačním obsahem, například byla v jednom sešitu obsažena telefonní čísla a pohotovostní mobilní telefony, bylo nutné tyto jednotlivé bloky dat rozdělit z logického hlediska do samostatných sešitů, aby bylo možné převést data do připravených tabulek. Další úprava spočívala v nastavení maximální délky u datových typů VARCHAR z navržené při analýze, na 255 (maximální možná délka) a odstranění všech podmínek NOT NULL, aby nedošlo k porušení relačních vztahů při převodu. Podmínka NOT NULL musela být odstraněna zejména u sloupců: Jmeno, Prijmeni, kde datový model předpokládal zaplnění, ale protože stávající data měla údaje o jménu a příjmení zaměstnance

vedena v jedné položce, byla by porušena podmínka relační vazby a převod dat by nebyl databázovým serverem povolen.

Po provedení importu dat pomocí DTS bylo zjištěno, že některé údaje se přes veškeré opatření nepodařilo převést korektně. Proto bylo přistoupeno k použití jiného balíčku DTS, který je určen pro transport dat ve formě textového souboru obsahující hodnoty oddělené středníkem (CSV soubor). Aplikace MS Excel umožňuje uložení dat právě v tomto formátu, který je obecně velmi dobře podporován databázovými servery. Převod tímto postupem proběhl bez problémů a byla převedena veškerá data. Tímto způsobem byla databáze naplněna stávajícími číselníky, hospodářskými středisky a ostatními informacemi, které vyžadovat zadavatel.

Poté byl spuštěn skript, který provedl úpravu přenesených dat do požadované podoby. Jednalo se především o rozdělení jména, příjmení a titulu do tří položek a nastavení výchozích údajů do položek: hospodářské středisko a funkce, která nebyla vyplněna. Následně byla všechna pole upravena do podoby, kterou předpokládala analýza, tedy nastavení podmínky NOT NULL, upravení délky VARCHAR a implementování relačních vztahů, které nabízí databázový server MS SQL.

3.4.4. Vývoj administrátorské aplikace

Prvním krokem při vývoji aplikace bylo vytvoření administrátorské aplikace, která by umožňovala správu celého systému. Jedná se o logický postup při vývoji aplikací, které obsahují administrační rozhraní. Administrační modul je dostupný pouze registrovaným uživatelům, kteří jsou rozděleni do tří skupin podle úrovně oprávnění.

3.4.4.1. Funkční popis

Hlavní funkcí administrátorské aplikace je údržba číselníků a telefonního seznamu prostřednictvím grafického rozhraní, které je dobře pochopitelné pro pověřené pracovníky.

Podrobný popis administračního modulu obsahuje příloha 7.

3.4.4.2. Zabezpečení

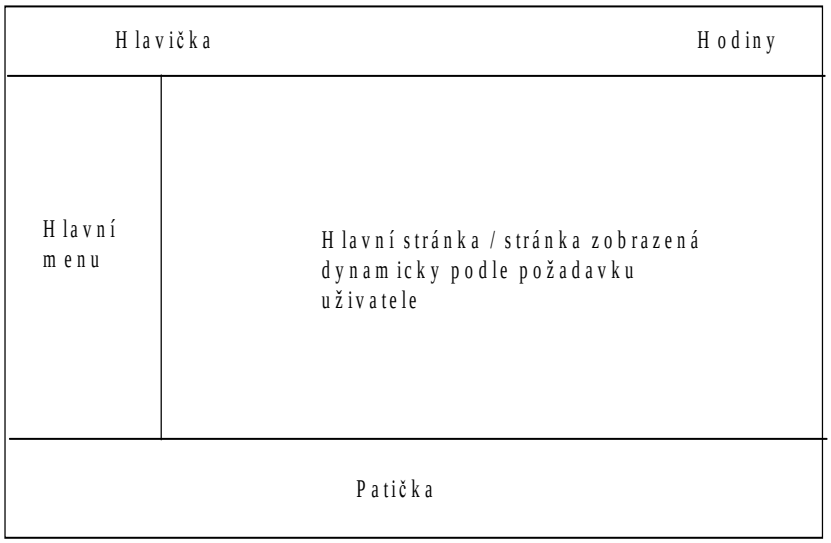
Aplikace je určena pro vnitropodnikovou informační síť, která není propojena na okolní internetovou síť, proto i požadavky na zabezpečení aplikace byly vytvářeny se zřetelem k výše uvedenému faktu.

Soubory pro administrační modul jsou uloženy mimo hlavní adresář se skripty uživatelského modulu, ke kterému je odepřen přístup neautorizovaným uživatelům. Běžný uživatel tedy nemá možnost zkopírovat skripty administračního modulu a provést jejich modifikaci s cílem aplikaci poškodit. Jako adresář pro administrační aplikaci slouží „./admin“.

Obr. 13 Formulář pro přihlášení k systému

K administrační aplikaci se lze přihlásit z uživatelské aplikace pomocí formuláře (Viz obr. 13), kde je nutné zadat přihlašovací jméno a heslo. Následně je v databázi provedeno porovnání zadaného jména a hesla skriptem login.php a pokud jsou zadané údaje korektní, je umožněn přístup do administračního modulu. V opačném případě je uživatel vrácen zpět na úvodní stránku aplikace.

Heslo je do databáze ukládáno po provedení kryptografické funkce MD5(),



Obr. 14 Rozvržení aplikace pro Page Engine

která provede převod řetězce na 32 bajtový řetězec. Například písmeno „r“ má podobu: 4b43b0aee35624cd95b910189b3dc231. Pokud by tedy útočník získal přihlašovací jméno a heslo z databáze, nebude jej moci použít.

Protože je protokol http bezstavový, nejsou uchovávány informace o provedeném přenosu. Pokud by se tedy uživatel přihlásil k jedné stránce, na druhé by již nebylo možné autorizovat jej s použitím zadaných údajů.

Tato problematika je řešena pomocí Session proměnných. Session proměnné jsou uloženy na straně serveru a po zaregistrování jsou automaticky posílány aplikačním serverem s požadavkem uživatele. Skript login.php zaregistruje session proměnné odpovídající přihlašovacímu jméno a heslu zadanému uživatelem a tyto údaje jsou kontrolovány při načtení každé stránky. Nelze tedy vstoupit na kteroukoli stránku administrační aplikace bez předchozího zadání jména a hesla.

Po skončení práce se administrátor odhlásí ze systému a přidělené session proměnné jsou smazány, aby je nebylo možné dále používat nepověřenou osobou.

Posledním druhem zabezpečení je výše zmiňované zaznamenávání všech provedených operací administrátora do databáze a možnost následného vyvolání uložených dat. Pokud by byly provedeny operace bez vědomí administrátora, bylo by to zřejmé zneužití jeho přihlašovacích údajů. Následně by bylo možné změnit jméno a heslo administrátora a tím zabránit jejich dalšímu zneužívání.

3.4.4.3. Grafické provedení

Grafické provedení je spolu s návrhem databáze jednou z nejdůležitějších částí návrhu, která přímo ovlivňuje kvalitu výsledné aplikace. Tato aplikace je určena do prostředí intranetu, proto základní návrh grafického provedení vycházel z internetových aplikací. U uživatelů administračního modulu nelze předpokládat, že jsou seznámeni s prací v grafickém prostředí internetu a tak byl kladen důraz na grafické provedení, které se bude podobat standardním aplikacím, např. aplikacím sady MS Office, na které jsou uživatelé zvyklí. Např. místo obvyklých odkazů ve formě zvýrazněného textu byly používány obrázky s motivem tlačítek.Základní kostra aplikace je vytvořena tabulkou, do které jsou dynamicky vládány hlavní stránky podle volby uživatele. Tento systém je označován názvem *Page Engine* (Viz obr. 14).

Na vrcholu stránky je umístěna hlavička s motivem společnosti Peguform Bohemia, k.s. společně s hodinami, které jsou vytvořeny pomocí jazyka JavaScript. Slouží uživatelům jako informační zdroj a vhodně doplňují hlavičku o aktivní prvek. Tato hlavička je společná pro všechny stránky administračního modulu, stejně jako patička, která obsahuje iniciály autorů společně s jednoduchou grafikou. Levá část je obsazena navigačním menu, které umožňuje přechod mezi jednotlivými funkčními moduly aplikace.

Obr. 15 Hlavní menu aplikace

Pro snazší orientaci uživatelů bylo zvoleno menu vytvořené v jazyce JavaScript (Viz obr. 15) , které je graficky a funkčně podobné menu, které je použito v aplikace MS Outlook, která je běžně používána ve společnosti zadavatele.

Jednotlivé ikony zastupují moduly aplikace. Po stisknutí dané ikony, dojde k načtení požadovaného modulu do oblasti hlavní stránky aplikace, jak to ukazuje obrázek 14.

Pokud chce uživatel použít jiný modul, který není zobrazen v dané nabídce, stiskne tlačítko pod kterým je umístěna příslušná ikona. Pro úpravy tabulek databáze tedy zvolí tlačítko *Úpravy*.

Tlačítko *Odhlásit* slouží k bezpečnému odhlášení z aplikace, dojde tedy k smazání všech session proměnných, které identifikovaly administrátora.

Obr. 16 Stránka aplikace

Každá stránka je složena podle kostry, která obsahuje v levém horním rohu obrázek přibližující funkci aplikace (Viz obr. 16), nadpis uvozující stránku a stručný popis ovládání jednotlivých stránek. Následuje vlastní obsah stránky.

Pro úpravu či vložení nového zaměstnance slouží formulář, který je buď vyplněn pouze výchozími hodnotami pro nového zaměstnance nebo pomocí jednoduchého vyhledávacího formuláře je vyhledán zaměstnanec a formulář je vyplněn stávajícími hodnotami. Administrátor provede potřebné úpravy a stiskne tlačítko *OK,* následně jsou změny uloženy do databáze.

Úpravy ostatních tabulek databáze jsou založeny na víceúrovňovém menu (Viz obr. 17). Administrátor zvolí požadovanou položku v první úrovni a obdobně postupuje v druhé úrovni. Tak administrátor zvolí požadovanou položku, kterou bude upravovat. Pak je zobrazen formulář pro položku, kde lze provést požadované změny.

Obr. 17 Víceúrovňové menu

Pro uživatele je toto ovládaní aplikace příjemné a logické.

3.4.4.4. Modul pro odsouhlasení nahlášených změn

Hlavní problém, který měla aplikace řešit, byl způsob nahlašování změn v údajích zaměstnanců. Tento způsob již delší dobu nevyhovoval, hlavně z důvodu malé pružnosti a rychlosti v provedení změn.

Podrobný popis modulu pro odsouhlasení změn je uveden v příloze 8.

Denně je nahlášeno v průměru 5 změn. Administrátor je tedy schopen provést odsouhlasení v časovém intervalu řádově minut, což je velké zrychlení v porovnání s původní praxí, kdy se změny projevily často až za několik.

3.4.4.5. Modul pro údržbu kmenových dat a číselníků

Dalším požadavkem zadavatele bylo snadné udržování dat uložených v databázi. Tento úkol řeší výše zmíněný modul, který umožňuje provádět veškeré změny v databázi v grafickém prostředí, které je pro uživatele přehlednější a pohodlnější, než stávající úprava v sešitech MS Excel.

Modul je podrobně popsán v příloze 7.

3.4.4.6. Modul pro správu administrační části aplikace

Modul umožňuje spravovat administrátorské účty a prohlížet uložené záznamy o provedených zásazích.

Podrobný popis modulu obsahuje příloha 7.

3.4.5. Vývoj uživatelské aplikace

Uživatelská aplikace je určena pro komunikaci s řadovými zaměstnanci, kteří nemohou být proškoleni v obsluze aplikace. Toto omezení vedlo ke stanovení základního komunikačního „jazyka“, kterým je uživateli sdělováno, jak je třeba postupovat při jednotlivých operacích. Byla vytvořena unifikovaná kostra aplikace založená na stejném aparátu jako administrátorská část (Viz obr. 14). Veškeré formuláře jsou zkontrolovány na správnost vložených údajů ještě před odesláním a komunikace s uživatelem je obsáhlejší a srozumitelnější, než v předchozí části.

K aplikaci je připravena jednoduchá nápověda, která je ve formě stránky podobné standardní nápovědě systému. Je otevřena v samostatném okně, takže uživatel může postupovat podle návodu, bez nutnosti opakovaně volit volby nápovědy.

3.4.5.1. Funkční popis

Aplikace má tři základní funkce. První je podávání aktuálních informací o zaměstnancích ve společnosti zadavatele, formou jednoduchých vyhledávacích kritérií. Druhou je možnost nahlásit změny, které budou předávány k odsouhlasení pověřeným pracovníkům. Poslední funkcí je vytváření tiskových sestav se zvoleným obsahem. Modul je dále schopen exportovat data do souborů MS Excel a formátu XML.

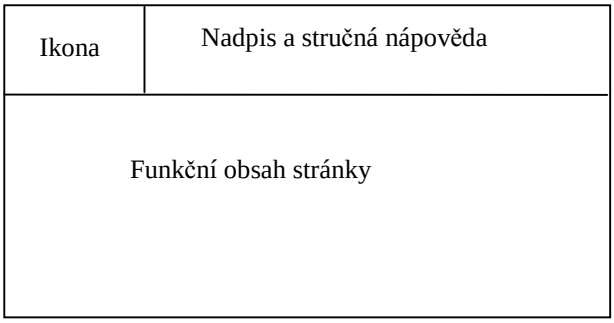
3.4.5.2. Grafické provedení

Uživatelský modul je určen řadovým zaměstnancům a je tedy nezbytně nutné, aby byl přizpůsoben potřebám uživatelů, kteří nemusejí být seznámeni s prací v prostředí internetu.

Veškeré formuláře byly zjednodušeny na nezbytně nutné položky, protože příliš složité by byly pro většinu uživatelů, jak ukázala praxe, neschůdné. Aplikace využívá zavedený model administračního modulu Page Engine. Grafické

provedení navigačního menu, které je umístěné na levé straně obrazovky, je totožné jako u předešlé aplikace, ale soubory nutné pro správnou funkci menu jsou umístěny v jiném adresáři, aby byly moduly co nejlépe odděleny.

Menu bylo získáno z internetu, jako volně dostupný doplněk a bylo upraveno pro potřeby aplikace. Jeho hlavní výhodou je podobnost v grafické i funkční specifikaci, které je shodná, jako menu aplikace MS Outlook. S touto aplikací je většina zaměstnanců seznámena a dokáže ji ovládat.



Obr. 18 Rozložení stránky

Struktura všech stránek je stejná, jako v administrační části (Viz obr. 18). Ikona s grafickým symbolem naznačuje funkci stránky a je shodná pro funkčně odpovídající stránky. Nadpis doplňuje označení stránky a poskytuje uživateli základní informace o možných úkonech, které bude možné provést. Následuje stručná a výstižná nápověda, která slouží k instruování uživatele, jak dále postupovat. Text na stránce je volen větší a přehlednější s optimalizací v rozlišení 1024x768 obrazových bodů, které je ve společnosti zadavatele nejpoužívanější. Aplikace však musí být přehledná i v nižším rozlišení a proto je šířka a výška stránky volena v procentech, což umožňuje prohlížeči automaticky měnit velikost stránky tak, aby byla dobře čitelná v různých rozlišeních.

Grafickou stránku aplikace tvoří kromě tabulek a různých druhů písma i jednoduché grafické ikony a obrázky s motivem tlačítek, které jsou standardně používány systémem, aby uživateli aplikace připomínala klasickou aplikaci, která není založena na internetové technologii.

Grafika je jednoduchá, s důrazem na funkční přehlednost, aby naváděla uživatele ke správnému používání a usnadňovala orientaci.

3.4.5.3. Modul pro vyhledávání kmenových dat

První funkcí, kterou požadovala společnost zadavatele, bylo snadné a přehledné vyhledávání údajů v telefonním seznamu a navazujících číselnících. První návrh vytvořit univerzální vyhledávací formulář podle vzoru standardních a obecně známých aplikací na internetu byl zadavatelem odmítnut, pro jeho

přílišnou složitost v ovládání. Následně bylo přijato řešení založené na sadě jednoduchých, maximálně dvoupoložkových formulářích, které jsou voleny pomocí základní nabídky z navigačního menu. Vyhledávání je možné podle následujících kritérií:

-  *Podle jména, příjmení nebo kombinovaně*. Uživatel zadá do kolonky jméno, příjmení nebo oba údaje a aplikace provede porovnání zadaného řetězce s údaji ve sloupcích jmeno, prijmeni v tabulce xTelSeznam. Je možné zadávat pouze části řetězců, tedy např. pokud bude zadáno ová, budou vyhledány všechny ženy s příjmením končícím ová. Zadaný řetězec je možné vložit v libovolném pořadí, je tedy jedno zda uživatel zadá *jméno příjmení* nebo *příjmení jméno*. Pokud by uživatel chtěl vyhledat všechny zaměstnance, kteří se jmenují např. Petr, pak vloží do formuláře *Petr %*. Znak „%„ je zástupný a označuje všechny povolené znaky.
-  *Podle telefonního čísla*, slouží pro vyhledání osob podle zadaného telefonního čísla. Opět je možné zadat pouze část telefonního čísla a výsledkem budou všichni zaměstnanci, jejichž telefonní číslo zadaný řetězec obsahuje.
-  *Podle hospodářského střediska*, vypíše všechny zaměstnance, kteří patří do zvoleného hospodářského střediska. Volba střediska je provedena stejným způsobem, jako tomu bylo u voleb v administrační části aplikace (Viz Obr. 17). Je možné provést označení většího množství položek, pokud je při volbě podrženo tlačítko *Ctrl.* Vyhledány pak budou osoby patřící do všech zvolených středisek v abecedním pořadí.
-  *Podle firmy*, protože společnost Peguform k.s. není jedinou velkou společností, ale řada zakázek je zadávána externím firmám, které však mají pracoviště v objektech firmy Peguform k.s., je nutné umožnit vyhledávání osob podle zařazení k jednotlivým firmám. Volba firmy je provedena stejným

způsobem jako u Hospodářských středisek s možností zvolit více položek najednou.

- *Vypsání pohotovostních mobilních telefonů*, každé pracoviště má přiřazen pohotovostní mobilní telefon, který má přidělen pracovník pověřený jeho obsluhou. Jedná se o telefonní číslo, které je voláno v případě nouze na pracovišti, takže je nutné jeho snadné vyhledání, bez zadávání přebytečných údajů. Proto jsou po zvolení vypsány všechny pohotovostní telefony, protože jejich počet pravděpodobně nepřekročí hranici dvaceti, je výpis přehledný a snadno je dohledáno požadované spojení.
- *Jinak*, slouží pro vyhledávání pracovníků podle
  - *Podle pracovišť*, firma zadavatele je rozdělena na několik pracovišť, pod které jsou zařazeni pracovníci. Uživatel zvolí požadované pracoviště z Pull Down menu a odešle požadavek na databázový server.
  - *Podle zařazení pracovníka*, jak bylo uvedeno výše je společnost zadavatele rozdělena na několik vzájemně propojených firem. Tato volba umožňuje vyhledat pouze pracovníky kmenové nebo externí zaměstnance.

Obr. 19 Výsledek vyhledávání

Výsledky jednotlivých vyhledávání jsou vypsány formou položek, které jsou odděleny horizontální linkou. Pro přehlednost jsou graficky rozlišeny ženy a muži (Viz obr. 19). Zobrazen je celkový počet nalezených záznamů a následuje soupis položek. Protože údajů o zaměstnancích je mnoho a nebylo by možné zobrazit je všechny bez ztráty přehlednosti a vypovídací schopnosti,

je vypsáno pouze několik základních informací, které budou ze statistického hlediska požadovány ve většině případů. Pokud jsou požadovány i ostatní údaje je možné zvolit odkaz *Podrobnosti*. Následně bude zobrazena osobní karta zaměstnance obsahující všechny údaje. Pro návrat k předchozí stránce slouží grafický symbol zelené šipky.

3.4.5.4. Modul pro nahlášení změn kmenových dat

Řešení aplikace předpokládá, že změny budou nahlašovat sami zaměstnanci.

Po zvolení ikony *Nahlášení změny* z nabídky *Změny* je uživatel vyzván k vyplnění jednoduchého formuláře. Aby bylo zajištěno, že bude zobrazen pouze formulář, který bude obsahovat data požadovaného zaměstnance, je nutné pro nahlášení změny nejprve zadat stávající telefonní číslo a jméno. Aplikace striktně nevyžaduje toto zadání, což je vhodné zejména s přihlédnutím k možnému rozvoji aplikace, a je možné zadat jen některé údaje.

Pokud zadání odpovídá více položek v databázi, jsou zobrazeny formuláře pro všechny vyhledané položky. Aby byl formulář co možná nejpřehlednější, je stejně jako v předchozím modulu nejprve zobrazen zjednodušený formulář, který obsahuje základní údaje, které bude chtít zaměstnanec změnit s největší pravděpodobností. Pokud nejsou zobrazeny požadované údaje, zvolí uživatel odkaz *Podrobnosti*, který zobrazí kompletní seznam dostupných položek.

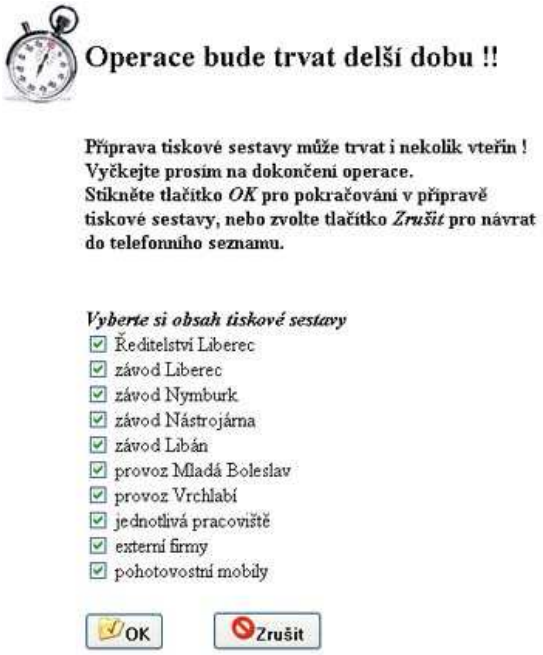
Zaměstnanec provede veškeré požadované změny a odešle je k odsouhlasení pověřenému pracovníkovi, tzn. že změny se projeví až po jejich odsouhlasení a potvrzení. Do té doby jsou pro všechny operace dostupné stávající údaje. Při nahlášení změn je vyžadován důvod změny, který je pak zobrazen administrátorovi a usnadňuje orientaci ve větším množství nahlášeným změn.

3.4.5.5. Modul pro vytváření tiskových sestav a export dat

Volbou položky *Export / tisk* z hlavního navigačního menu je zobrazena nabídka pro vytvoření tiskové verze telefonního seznamu a možnost exportovat data do souboru.

První možnost, tedy vytvoření tiskové sestavy, slouží k navržení tiskové verze databáze, podle individuálních požadavků. Po zvolení ikony  *tisk*

telefonního seznamu je zobrazena nabídka s možným obsahem tiskové sestavy (Viz obr. 20). Je možno zvolit z celkem deseti možností, jako jsou závody, pohotovostní mobilní telefony a jiné, zaškrtnutím požadované položky. Stiskem tlačítka *OK* je vygenerována stránka, která je naformátována a připravena pro tisk. Formátování je provedeno pomocí CSS a volby *media*. Grafická stránka tiskové sestavy je přizpůsobena tak, aby se co nejvíce podobala původní verzi telefonního seznamu. Tisková verze je otevřena v celé velikosti okna, aby bylo možné tisknout pomocí aplikace MS IE. Ve vrchní části okna jsou umístěna dvě základní ovládací tlačítka, jedno pro návrat do aplikace a druhé sloužící k tisku připravené sestavy. Aby bylo možné tisknout i s barevným rozlišením nadpisů je nutné v MS IE povolit tisk barev a obrázků v nastavení aplikace, nebo pomocí ovládacích panelů a volby *Možnosti internetu* .



Obr. 20 Volby tiskové sestavy

Druhá ikona v menu *Export / tisk* - *Export dat* slouží pro vytvoření souboru, který bude možné otevřít v jiných aplikacích. Společnost zadavatele požadovala, aby bylo možné upravovat stávající data pomocí aplikace MS Excel.

Po zvolení výše uvedené ikony je uživatel opět dotázán, které tabulky z databáze bude požadovat v souboru a možnost zvolit jeden ze dvou typů souborů. První je typ HTML, který je kompatibilní s aplikacemi MS IE, MS Excel a mnoha jinými a je upraven pomocí CSS do formátu podobného původní verzi telefonního seznamu.

Druhým formátem je dnes stále populárnější a oblíbenější formát XML se styly, pomocí kterých je opět graficky upraven do podoby starého telefonního seznamu. Tento formát XML byl původně připraven pro MS Excel, ale protože ve společnosti zadavatele je využívána verze MS Office, která ještě formát XML

nevyužívá je tento formát připraven pro pozdější využití. Jeho hlavní předností bude široká podpora aplikací, která se pravděpodobně v budoucnu ještě rozšíří.

Stávající aplikace tedy využívají hlavně starší verzi HTML, se kterou umějí aplikace sady MS Office 2000 pracovat.

3.5. Testování

Důležitým krokem před spuštěním aplikace jsou testy, které lze rozdělit do několika základních úrovní. Toto dělení má za úkol optimalizovat celý průběh testování, aby například kontrola překlepů neprobíhala na testovacím serveru u zadavatele, ale byla provedena bez větších nároků ve vývojovém prostředí, kde lze snadno chyby tohoto charakteru objevit a odstranit.

3.5.1. Unit testy

Unit testy jsou testy jednotlivých jednotek, které jsou prováděny ve vývojovém prostředí a mají za úkol odstranit nejzávažnější chyby ve funkčnosti a grafickém provedení. Slouží programátorovi při vývoji aplikace, kdy jsou v databázi založeny zkušební položky, na kterých jsou prováděny testy jednotlivých jednotek a modulů. Unit testy tedy například slouží k ověření správné funkce právě vyvíjeného nebo čerstvě dokončeného skriptu. K testování slouží EM, který umožňuje snadno určit, zda vyvíjený skript ukládá data korektně, případně zda jsou zobrazená data správná. Druhým nástrojem pro testy je QA, který umožňuje odstranit chyby a testovat vytvářené dotazy jazyka SQL. Po provedení Unit testů jsou z aplikace odstraněny hlavní chyby a nedostatky, které je schopen odstranit a odhalit programátor při vývoji.

3.5.2. Integrační testy komponent a modulů

Integrační testy testují aplikaci z celkového pohledu, testují tedy spolupráci jednotlivých skriptů v modulech. Testování probíhá souběžně v prostředí vývojovém a testovacím, které je připraveno u zadavatele a odpovídá svými parametry provoznímu prostředí. Je tedy například použit stejný aplikační i databázový server, jen není aplikace připojena k vnitropodnikové síti.

Testování probíhalo za spolupráce programátora a konzultanta ze společnosti zadavatele. Testována byla administrační část aplikace s jednotlivými moduly s přihlédnutím k požadavkům na bezpečnost a spolehlivost provozu.

Uživatelská část aplikace byla testována s důrazem na funkčnost a jednoduchost ovládání s přihlédnutím k potřebám zaměstnanců a sladění ovládání se stávajícími vnitropodnikovými aplikacemi. Při objevení chyby ve společnosti zadavatele, byl emailem odeslán programátorovi záznam provedené akce, která neskončila korektně spolu s výpisem chybných skriptů. Po odstranění problému spolu s provedením Unit testů, byl opravený skript odeslán zpět k otestování na testovacím serveru. Tímto způsobem se podařilo odstranit většinu funkčních závad aplikace.

3.5.3. Uživatelské akceptační testy

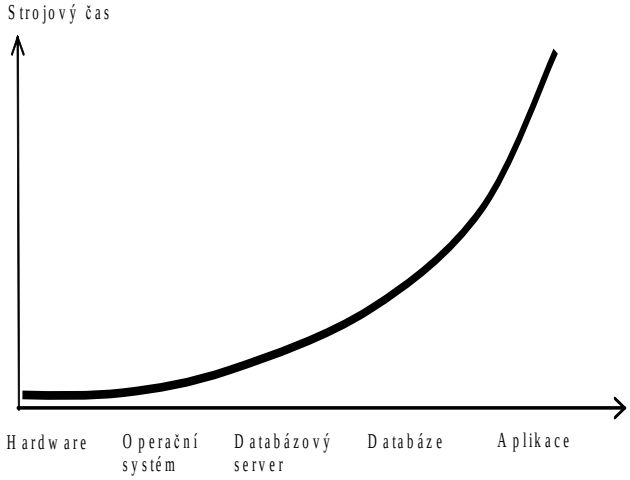
Jedním z nejdůležitějších testů je uživatelský akceptační test, který slouží k přizpůsobení aplikace potřebám uživatelů, pro které je vyvíjena. Společnost Peguform Bohemia, k.s patří mezi velké společnosti, ve kterých je nutné před spuštěním provozu aplikace tohoto charakteru provést předvedení funkčnosti a provedení před pověřeným pracovníkem z osobního oddělení, který odsouhlasí, případně nahlásí další požadavky na aplikaci.

Dále je aplikace předložena vybrané skupině koncových uživatelů a administrátorů, kteří nahlašují své připomínky k funkčnosti ovládání a přehlednosti celé aplikace. V průběhu testování byly vzneseny některé další požadavky na aplikaci, jak v uživatelské tak i administrační části aplikace. Bylo provedeno zaškolení budoucích administrátorů v provádění změn a ovládání administrátorské aplikace.

3.5.4. Systémový test

Systémovým testem je zkoušen především výkon celého programu, aby nebyl, vzhledem k nasazení na vnitropodnikové síti, příliš velkou zátěží provozu.

K testování výkonu dobře slouží QA, který umožňuje sledovat náročnost jednotlivých dotazů v různých měřítcích a usnadňuje jejich optimalizaci.



Obr. 21 Výkon aplikace

Dalším parametrem, který zvyšuje výkon celé aplikace z pohledu databázového stroje, je použití trvalého spojení s databázovým serverem, takže odpadá nutnost otvírání nového připojení při provádění dotazů, což má za následek celkové zrychlení aplikace. Výkon celé aplikace závisí na prostředcích, jak je ukázáno na obrázku 21. [7] Je tedy patrné, že hlavní podíl na výsledném výkonu aplikace má efektivnost jejího kódu a databáze . Vzhledem k tomu, že zvolený databázový server MS SQL disponuje dostatečným výkonem, lze jej z řetězce vypustit, stejně jako operační systém a hardware, který je pro aplikaci více než dostatečný.

Testování je tedy zaměřeno hlavně na zefektivnění kódu aplikace.

3.6. Implementace

Skončením fáze testování je umožněno přejít k vlastnímu nasazení a zprovoznění aplikace v provozním prostředí.

3.6.1. Instalace a oživení aplikace na provozním serveru

Pro aplikaci byl vytvořen zvláštní adresář na provozním serveru, ke kterému byla přidělena práva:

- Kořenový adresář „ ./ “, právo čtení.
- Adresář s administračním modulem „ ./admin “, právo čtení.
- Adresář s grafikou „ ./image “, právo čtení.

- Adresář pro exportované soubory „ ./exp “, práva čtení a zápisu, aby uživatel mohl kopírovat soubory vytvořené exportním modulem.

Následovalo kopírování skriptů do připravených adresářů se zkoušením funkčnosti jednotlivých modulů.

Databázový server byl nastaven ve fázi uživatelských testů a jednalo se hlavně o založení všech objektů databáze s importem dat, jak je zmíněno v kapitole *3.4.3*.. Pro naplnění databáze aktuálními daty byli vybrání pracovníci, kteří provedli aktualizaci pomocí administračního modulu, aby bylo možné spustit aplikaci ve zkušebním režimu.

Posledním krokem bylo vytvoření odkazu v intranetové síti na zkušební verzi nové aplikace.

3.6.2. Zkušební provoz aplikace

Při zkušebním provozu, byla aplikace dostupná všem uživatelům přes odkaz v intranetové síti podniku zadavatele. Souběžně s novou aplikací byla dostupná stará verze telefonního seznamu, aby byly dostupné informace o zaměstnancích při případném vážnějším problému nového programu. Při zkušebním provozu byly také vzneseny další požadavky na aplikaci (uvedeny v kapitole 3.7).

Aplikace je opatřena počítadlem přístupů, takže bylo možné sledovat návštěvnost stránek, která se pohybovala okolo padesáti přístupů za týden, což bylo velmi příznivé číslo vzhledem k ke zkušebnímu provozu aplikace.

3.7. Optimalizace

Při zkušebnímu provozu aplikace se projevily některé nedostatky, které bylo nutné odstranit ještě před spuštěním aplikace. Jednalo se především o požadavek na úpravu tiskové sestavy a o úpravu menu pro několik „nešťastných“ uživatelů s prohlížečem Mozilla.

3.7.1. Tisková sestava a export dat

Tisková verze, jak je popsána v kapitole 3.4.5.5, byla v původním zadání navržena pouze jako doplňková a volitelná funkce, ale protože se ukázalo, že

určitá část pracovníků nemá možnost využívat novou intranetovou aplikaci, bude třeba vytvořit modul pro tisk seznamu. Zkušební provoz poukázal na nutnost výběru obsahu tiskové sestavy, aby bylo možné tisknout pouze určité části podle volby uživatele.

Při zavádění aplikace bylo rovněž provedeno další rozdělení podniku na oddělení, takže bylo nutné tuto skutečnost promítnou do konečné verze produktu.

Po dodatečných úpravách byla tisková verze odsouhlasena a byl povolen provoz aplikace v provozním prostředí, jako oficiální telefonní seznam.

3.7.2. Úprava navigačního menu pro uživatele Unixových systémů

Původní předpoklad, že aplikace bude provozována pouze na systémech od společnosti Microsoft s internetovým prohlížečem MS IE, nebyl zkušebním provozem potvrzen a tak bylo nutné provést úpravu stávající navigačního menu, tak aby vyhovovalo i uživatelům Unixu s prohlížečem Mozilla.

Řešením by bylo použití jiného navigačního menu, které by bylo kompatibilní i s prohlížečem Mozilla nebo dynamické generování stávajícího menu s vytvořením jiného menu kompatibilního s alternativními prohlížeči. První možnost byla, vzhledem ke spokojenosti uživatelů se stávajícím grafickým provedením nevhodná a omezila by většinového uživatele MS IE, protože bylo zjištěno, že prohlížeč MS IE používá drtivá většina pracovníků s výjimkou jednoho pracoviště s výše zmíněnou Mozillou. Proto bylo zvoleno druhé řešení, kde bylo jako alternativní menu zvoleno menu vytvořené pomocí CSS, které bude kompatibilní se všemi stávajícími prohlížeči s výjimkou velmi starých verzí, které však nepřipadají ve společnosti zadavatele v úvahu.

Prvním krokem při vytvoření alternativního menu, bylo naprogramování skriptu, který bude schopen rozpoznávat prohlížeč, který s aplikací pracuje. Jedná se o problém, který lze elegantně řešit pomocí regulárních výrazů a hlavičky $HTTP_USER_AGENT, která je dostupná v každém skriptu. Zde se projevila výhoda použitého systému vkládání stránek do připravené kostry Page Engine, protože postačí provést kontrolu prohlížeče na úvodní stránce *index.php,* protože ostatní jsou do této vloženy, není již třeba kontrolovat každou stránku zvlášť. Dalším faktorem toho usnadnění je i zrychlení celého procesu rozpoznání prohlížeče a tím i celé aplikace.

Druhým krokem bylo vytvoření čtyř tabulek v databázi, jedná se o tabulky:

- xMenu, obsahuje definici hlavních položek menu uživatelské aplikace.
- xPolozkyMenu, obsahuje informace potřebné pro vytvoření jednotlivých ikon v menu pro uživatelskou aplikaci.
- xAdminMenu, obdoba xMenu pro administrační aplikaci.
- xPolozkyMenuAdmin, obdoba xPolozkyMenu pro administrační aplikaci.

Třetím krokem, bylo vytvoření skriptů, které budou vytvářet soubor folders.js, podle obsahu tabulek v databázi. Soubor folders.js obsahuje definici polí proměnných, které využívá skript outbar.js, pro vytvoření nabídky menu.
Rozšíření aplikace o skripty vytvářející menu dynamicky, umožnilo jejich úpravu z administračního modulu v grafickém prostředí, které má shodné provedení jako ostatní nabídky pro upravování položek *(*Viz obr. 17*).* Při úpravě položek menu, je nutné zadat parametry, jako jsou číslo položky, které udává pozici v menu, obrázek, který bude tvořit ikonu položky a skupinu, do které bude odkaz patřit. Jedná se o řadu údajů, které by byly obtížně zjistitelné, proto byla vytvořena nápověda, která tyto informace načítá přímo z databáze, včetně dostupných obrázků, a administrátor zvolí z nabízených položek. Po odeslání požadavku je automaticky generován nový soubor *folders.*js a změna se okamžitě projeví i ve vlastní aplikaci. Tato možnost je výhodná zejména pokud je brán zřetel na budoucí rozvoj aplikace, kdy bude potřeba velmi pružně upravovat položky v menu a změny bude provádět pracovník neznalí jazyka JavaScript .

Při úpravě tabulek, které obsahují informace o menu, je třeba naprogramovat řadu opatření, aby nedošlo k porušení vazeb mezi položkami. Například pokud chce administrátor přesunout jednu položku z jedné nabídky do druhé, je nutné z první nabídky odečíst položku a stávající znovu seřadit a stejné operace je třeba provést v cílové nabídce. Stejné problémy nastávají při vytváření nové nabídky, která nebude nějaký čas obsahovat žádné položky, protože jazyk JavaScipt, který je použit pro naprogramování menu, je velmi choulostivý na chyby v zakončení řádků, interpunkci a dodržení všech pravidel. Následné ladění je velmi komplikované a časově náročné, proto bylo přikročeno k vytvoření alternativního menu pouze pro uživatelskou aplikaci

a stejné operace pro administrační modul budou pouze připraveny pro pozdější doplnění. Byly tedy vytvořeny tabulky i pro administrační modul a vazby na skripty vytvářející menu.

Aplikace je v tomto ohledu velmi flexibilní a menu lze bez větších úprav použít i v jiných aplikacích.

3.7.3. Rozšíření databázové struktury o emailovou adresu

Zkušební provoz odhalil nedostatek systému, který neobsahoval emailovou adresu. Tato položka nebyla v původním návrhu zanesena a její doplnění si vyžádal až zkušební provoz aplikace, kdy tuto informaci zaměstnanci vyžadovali. První řešení problému spočívalo v přepracování databázové struktury a přidání potřebného sloupce do hlavní tabulky telefonního seznamu. Toto řešení je vzhledem k rozsáhlosti skriptů, které používají databázi, velmi nevhodné, neboť by si vyžádalo úpravu všech částí uživatelské i administrátorské aplikace.

Proto bylo vypracováno alternativní řešení, které použilo stávající datovou strukturu, přesněji položku *Obrazek* , která měla být použita pro vkládání obrázků zaměstnanců, ale již toto nebylo realizováno vzhledem k malému zájmu zaměstnanců. Položce *Obrázek* byl změněn datový typ z *Image* na *VARCHAR(255)*, který vyhovuje požadavkům na vložení emailové adresy. Protože ve všech skriptech je tato položka využívána, nebylo třeba velkých změn, které by mohly způsobit značné prodlevy ve spuštění aplikace. Všem zaměstnancům byla vložena výchozí emailová adresa složená z (prvního písmena jména).(příjmení)@peguform.cz po odstranění interpunkce. Takto vložená adresa vyhovuje většině zaměstnanců. U ostatních byla provedena oprava pomocí modulu pro nahlášení změn.

4. Shrnutí

4.1. Vyhodnocení zkušebního provozu

Během zkušebního provozu byly vzneseny i další požadavky než výše jmenované, ale již nebyly realizovány z nejrůznějších důvodů.

Jedním z těchto požadavků bylo i napojení aplikace na firemní sms bránu, která však byla před oficiálním spuštěním aplikace společností Eurotel, která poskytuje mobilní služby společnosti zadavatele, zrušena.

Zkušební provoz potvrdil připravenost aplikace pro spuštění a její dobrou koncepci včetně grafického provedení, které bylo přijato velmi kladně. Během zkušebního provozu byla zaškolenými pracovníky aktualizována data, např. zařazení pracovníků do hospodářských středisek, nastavení firmy, ke které pracovník patří, atd.. Takže aplikace obsahovala skutečně aktuální informace, které umožnily spuštění aplikace bez velkých prodlev vzniklých při nahlašování velkého počtu změn zaměstnanci, jejichž záznamy nebyly korektní.

Došlo také k několika kosmetickým úpravám v názvosloví jednotlivých položek ve formulářích. Na příklad lze uvést položku *zaměstnán*, která označovala stále zaměstnané pracovníky. Tato položka byla označena za nevhodnou a působící špatně na zaměstnance z psychologického hlediska, proto není zobrazována běžným uživatelům a je dostupná pouze v administračním modulu aplikace.

4.2. Průzkum uživatelské spokojenosti

Po uvedení aplikace do oficiálního provozu byl proveden průzkum spokojenosti uživatelů pomocí jednoduchého hlasování (Viz obr. 22). Uživatelé

Obr. 22 Anketa spokojenosti uživatelů

dvanáct dní odpovídali na otázku zda jsou spokojeni s provedením nové aplikace. Výsledek byl velmi pozitivní, poměr hlasů byl 112 ku 20. Hlasování potvrdilo předpoklady, které byly uvažovány při návrhu aplikace, její grafické uzpůsobení, funkční ovládání a návrh formulářů pro vyhledávání. Lze jen položit otázku co nevyhovuje uživatelům, kteří odpověděli záporně, ale vzhledem k počtu koncových uživatelů nelze předpokládat, že by aplikace vyhovovala všem.

Spokojenost s aplikací vyjádřili i administrátoři opovědní za provádění nahlášených změn a správu databáze.

5. Diskuze

Aplikace byla od počátku vytvářena s otevřeným kódem, která bude pružně reagovat na případné změny ve společnosti Peguform Bohemia, k.s..

Tento záměr do značné míry ovlivňuje programátora při psaní aplikace, je tedy nezbytné komentovat všechny důležité příkazy, aby se v kódu aplikace později vyznali i jiní programátoři.

5.1. Bezpečnost aplikace

Aplikace byla od počátku určena k provozování na vnitropodnikové informační síti, nepředpokládal se tedy větší zájem o proražení ochrany aplikace. Aplikace je opatřena v zásadě základním bezpečnostním zabezpečením, které je možné v případě nasazení aplikace k volnému používání mimo podnikovou síť doplnit a zdokonalit.

Zabezpečení aplikace, jak bylo popsáno v kapitole *3.4.4.2,* je postačující pro provoz v chráněném prostředí intranetové sítě a jeho doplnění spočívá zejména v:

- *Zamezení interpretace HTML kódu*, jedná se o metodu, jak obejít zabezpečení aplikace pomocí vložení nebezpečného kódu do formulářových polí. K zamezení lze využít např. funkci *htmlspecialchars($text)*, která převede všechny speciální znaky HTML kódu na entity, které nebudou interpretovány.
- *Script injection,* kdy jsou aplikaci podvrženy proměnné, které využívá, což může mít za následek porušení bezpečnosti. Řešením je používání proměnných *$http_post(get, session)_vars[“název proměnné“].*
- *Zabezpečení SQL příkazů,* zamezení interpretace SQL příkazu, podvržených aplikaci uživatelem. Řešením je použití regulárních výrazů, pomocí kterých lze odstranit nebezpečné znaky a SQL příkaz pak nebude interpretován.
- *Další možné způsoby porušení aplikace,* tato problematika je velmi rozsáhlá a v současné době je jí věnována velká pozornost především v diskusních fórech internetových stránek zabývajících se problematikou programování na internetu.

Bezpečnost na internetu je s ohledem na použitý protokol TCP/IP velmi diskutabilní. Řada firem přinesla vlastní řešení, za povedené a v současné době používané, lze označit protokol SSL od společnosti Netscape Communications Corporation, který pracuje nad protokolem http a přenáší data v šifrované podobě.

5.2. Kompatibilita a přenositelnost aplikace

Návrh aplikace uvažoval prostředí systému MS Windows 95/98/NT/2000 /XP s nainstalovaným prohlížečem MS IE verze 5.0 a vyšší. Předpoklady byly již v průběhu zkušebního provozu překonány a ukázala se potřeba přizpůsobit aplikaci i ostatním operačním prostředím a alternativním prohlížečům. To si vyžádalo přepracování aplikace vzhledem na kompatibilitu, kde hlavním problémem bylo navigační menu vytvořené speciálně pro prohlížeče společnosti Microsoft. Alternativní menu realizované pomocí CSS je již zcela kompatibilní s novějšími verzemi internetových prohlížečů, protože je generováno ze stejné tabulky v databázi, lze upravovat tyto, jinak nezávislé, komponenty současně.

Otázka kompatibility aplikací v prostředí internetové sítě je složitý problém, který se snaží řešit společnost W3C, která odpovídá za standardy internetu, zavedením jazyka XHTML. Přes veškerou snahu je však internet pro programátora oříškem. Velkou část vývojového času zabírají právě otázky kompatibility a přizpůsobení co největšímu počtu uživatelů, což je hlavní snahou všech firem majících své prezentace na internetu.

Budoucí vývoj aplikace by tedy mohl být také zaměřen na přepracování kódu s ohledem na funkčnost v co největším procentu prohlížečů, pokud nedojde ke sjednocení prohlížečů na platformu MS, jak je předpokládáno.

Aplikace není závislá na použitém SW ani HW.

5.3. Perspektivy budoucí integrace aplikace s ostatními DB systémy společnosti zadavatele

Jedním z požadavků společnosti zadavatele, byl otevřený databázový model, který by umožnil napojení aplikace na ostatní databázové systémy, které jsou ve společnosti vedeny. Návrh databáze tady počítal s rozšířením, protože v době návrhu nebyla k dispozici identifikační čísla zaměstnanců, která složí jako jednoznačný identifikátor v ostatních databázích společnosti zadavatele, bylo spojení obou systémů vyřešeno pomocí vazební tabulky, která bude obsahovat údaje propojující tabulky databází. Jedná se o standardní řešení, které je běžné u podobných databázových aplikací. Tímto způsobem je možné propojit libovolné databázové systémy s aplikací, což lze považovat za mocný argument kompatibility.

6. Závěr

Diplomová práce odstranila závažné nedostatky v uchovávaných datech o zaměstnancích společnosti Peguform Bohemia k.s.. Aplikace si kladla za cíl především usnadnit a zefektivnit práci zaměstnanců, sjednotit nejednotné údaje a snížit náklady na jejich udržování, nikoli kompletní přepracování stávajícího systému. Aplikace byla, vzhledem k provedenému průzkumu, kladně přijata a hojně využívána zaměstnanci. Rovněž se podařilo splnit požadavky na otevřenost databázové i aplikační architektury, včetně později doplněných požadavků na kompatibilitu.

Řešení zadaného problému uvažuje přístup k databázi pomocí univerzálního rozhraní ODBC, které však poskytuje menší výkon ve srovnání s nativním přístupem k databázovému serveru, lze tedy doporučit přenesení aplikace do prostředí, které umožní využití plného výkonu aplikace. Společnost zadavatele stále klade důraz na využívání tiskové verze telefonního seznamu, který je součástí aplikace, zajímavým rozšířením by bylo nainstalování terminálů do snadno přístupných prostor, kde by aplikaci mohli využívat i zaměstnanci bez stálého připojení k vnitropodnikové síti a zároveň by došlo k snížení nákladů na distribuci tištěné verze společně s menším zatížením životního prostředí.

Literatura :

[1] Císařová, K. : Studijní materiály k předmětu ŘD

[2] Riordan , R. M. : Vytváříme relační databázové aplikace

[3] Castagnetto, J. – Rawat, H. – Shumann, S. – Scyllo, C. – Valiath, D. : Programujeme PHP profesionálně

[4] Systémová analýza, studijní materiál společnosti LBMS.

[5] Příloha časocopisu Computerworld 45/97.

[6] SQL Server 7.0 implementace databází, Training Kit

[7] Lacko, L. : Web a databáze

Přílohy diplomové práce:

Příloha 1	Plán projektu
Příloha 2	Souhrn problémů a požadavků
Příloha 3	Data flow diagram (DFD)
Příloha 4	Entity relationship diagram (ERD)
Příloha 5	SQL skript
Příloha 6	Aplikace Query analyzer a Enterprise manager
Příloha 7	Příručka uživatele administračního modulu aplikace Telefonní seznam Peguform Bohemia, k.s.
Příloha 8	Rejstřík

Příloha 1 Plán projektu

Příloha 2 Souhrn problémů a požadavků

Příloha 3 Data flow diagram (DFD)

Příloha 4 Entity relationship diagram (ERD)

Příloha 5 SQL Skript

Příloha 6 Query analyzer a Enterprise manager

Příloha 7 Příručka administrátora administračního modulu aplikace Telefonní seznam Peguform Bohemia, k.s.

Příloha 8 Rejstřík