UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Plánování a sledování prací v lesních porostech od těžby až po jejich ukončení

Švec Ondřej

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej ŠVEC**
Osobní číslo:	**I07972**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Plánování a sledování prací v lesních porostech od těžby až po jejich ukončení**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvoření www aplikace postavené na databázi Oracle. Aplikace by měla postupně sledovat přeměnu starého lesa na les nový pro potřeby hajného. Hajný by měl tak kompletní přehled o stavu jednotlivých porostů, zásobách vytěženého dřeva v lese, jeho dovozu a prácích, které jsou ještě potřeba splnit.

Aplikace měla umožňovat následující práce:
* naplánování těžby lesních porostů pro následující roky
* vyhotovení číselníků vytěženého dřeva v jednotlivých porostech a jejich sumáře
* přibližování dříví v porostech
* odvoz dřeva z lesa
* výkaz skladu
* odstraňování klestu
* zalesnění vytěžených ploch
* vyžínání proti buřeni
* nátěry stromků proti okusu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

***James R. Groff, Paul N. Weinberg. SQL Kompletní průvodce. 1.vydání. Brno Computer Press, 2005 ISBN 80-251-0369-2**

***CASTAGNETTO, J. a kol. Programujeme PHP profesionálně. Computer Press, 2001. 676 s. ISBN 8072263102**

***DRUSKA, P. CSS a XHTML - tvorba dokonalých webových stránek krok za krokem. Grada 2006.**

***Jak psát web, návod na html stránky [online]. Dostupný z WWW: http://www.jakpsatweb.cz**

***Dílna dobrého stylu [online]. Dostupný z WWW: http://wellstyled.com**

***Seriál o PHP na linuxsoft.cz [online]. Dostupný z WWW: http://www.linuxsoft.cz/article_list.php?id_kategory=181**

***PHP manuál [online]. Dostupný z WWW: http://www.php.net/manual/cs**

Vedoucí bakalářské práce: **prof. Ing. Karel Šotek, CSc.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 9. 5. 2010 Ondřej Švec

Anotace

Cílem projektu je vytvoření webové aplikace postavené na databázi Oracle a s použitím jazyků XHTML, PHP a JavaScript. Aplikace by měla sledovat postupnou přeměnu starého lesa na les nový pro potřeby hajného. Hajný by měl tak kompletní přehled o stavu pracích v jednotlivých porostech, zásobách vytěženého dřeva v lese, jeho odvozu a prací, které jsou ještě potřeba splnit.

Klíčová slova

Webová aplikace, Oracle, XHTML, PHP, CSS, SQL, JavaScript, lesní práce, těžba dřeva

Title

Planning and Monitoring of Processing Forest crops from Cutting to Re-planting new trees

Annotation

The project aims to create a web application on Oracle databse using XHTML, PHP and Javascript. The application should monitor the gradual conversion of an old forests to a new forest for the needs of the gamekeepers. The gamekeepers would have a complete overview of the different crops, stocks of timbers in the forest, its transport and other operations required therein

Keywords

Web application, Oracle, XHTML, PHP, CSS, SQL, JavaScript, forest operations, logging

Obsah

Seznam zkratek

LČR	Lesy České Republiky
PHP	PHP: Hypertext Preprocessor, původně: Personal Home Page
SHA1	Secure Hash Algorithm 1
DOM	Document Object Model
(X)HTML	(eXtensible) Hypertext Markup Language
CSS	Cascading Style Sheets
XML	Xtensible Markup Language
AJAX	Asynchronous JavaScript And XML
DTD	Document Type Definition
API	Application Programming Interface
URL	Uniform Resource Locator
GPL	General Public License
ASP.NET	Active Server Pages .Network
SQL	Structured Query Language
PL/SQL	Procedural Language/Structured Query Language
DBMS	DataBase Management Systém

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem bakalářské práce je vytvoření webové aplikace pro plánování a evidování prací v lesních porostech od těžby až po zajištění kultur. Tím jsou myšleny základní úkony hajného při přeměně starého lesa na les nový. Aplikace sleduje pro každý naplánovaný porost aktuální stav těžby, zásobu dřeví v lese, odstraňování klestu, zalesnění vytěžených ploch, vyžínání proti buřeni, nátěry stromků proti okusu a jednotlivé probírky. Díky tomu má hajný potřebný přehled o stavu rozpracování jednotlivých činností v porostech. Aplikace je připravena jak pro jednotlivce, tak pro nasazení do firem, kde umožňuje celkový přehled prací všech hajných.

Nejprve v bakalářské práci vysvětluji, jaká je náplň práce hajného a co vůbec obnáší. Dále seznamuji s jednotlivými pracemi v lesním hospodářství.

Poté vysvětluji použité technologie, které byly použity na tvorbu aplikace. Tedy jednotlivé programovací jazyky a aplikace, v kterých jsem práci prováděl.

Jako poslední část je vysvětlení návrhu, popisu a konstrukce samotné aplikace. Na závěr je několik úvah na její možné rozšíření.

2 Práce v lesních porostech

Práce hajného není chodit s puškou po lese, jak se mnoho lidí domnívá. To je činnost myslivců, členů mysliveckých sdružení. Ve skutečnosti se jedná o komplexní péči o les – to znamená provádět veškeré výchovné zásahy, které zkvalitňují lesní porosty a těžit dříví, což přináší ekonomický efekt. Dále je to starost o lesní dělníky – především zadávání práce, kontrola množství a kvality vykonané práce, výpočet jejich mzdy a kontrola dodržování předpisů bezpečnosti práce.

Napřed je třeba vysvětlit základní pojmy lesního hospodářství a strukturu organizací, které v něm pracují.

Vlastníkem většiny lesů v České republice jsou Lesy České republiky – dále jen LČR. LČR sami nezaměstnávají žádné dělníky na práci v lese. Na tu si najímají podnikatelské subjekty, většinou jde o lesní společnosti. Ty pracují pro LČR a vykonávají pro ně veškeré práce. Pro výběr společnosti, která na jednotlivých lesních celcích bude pracovat, vyhlašují LČR veřejná výběrová řízení.

LČR vypracovávají na každý rok:

- Projekty těžebních prací, které obsahují úmyslné těžby rozdělené na mýtní (holoseče) a předmýtní (probírky).
- Projekty pěstebních prací, které obsahují veškeré výchovné zásahy.

Podle těchto projektů následně soukromé společnosti pracují. Společnosti jsou za ně od LČR placené podle ceníku, který je sestavován na základě nabídky při výběrovém řízení. Tyto společnosti zase naopak od LČR nakupují dříví dle různých druhů dřevin, průměrné hmotnatosti, druhu těžby a nutné technologie přibližování dřeva.

LČR mají svůj vlastní počítačový program, kde jejich zaměstnanci plánují a evidují jednotlivé práce. Program je zaměřen na sumarizaci těchto činností v celé republice. Firmy pracující v lese takový komplexní program, který by sledoval jejich celou činnost, nemají a ani jej nepotřebují mít tak podrobný.

Hajní pracují na velikých výměrách lesů, kde vykonávají množství velmi rozdílných prací. Mají většinou k dispozici pouze dílčí programy, jako třeba tvorbu číselníků či výkaz skladu. Ostatní data mají v lepším případě v excelovských tabulkách, nebo jen ručně psané poznámky na papíru. Tyto programy často mezi sebou nekomunikují, neřeší vazby mezi pracemi nebo mají jiné nedostatky, chybí jednotný formát. Přitom firmy potřebují veškeré činnosti sledovat a evidovat (výroba, výkaz skladu, aj.), kontrolovat, jestli je vše provedeno a následně zaplaceno od LČR. Při neuhlídání těchto plateb mohou vznikat lesním společnostem veliké finanční ztráty. Dále potřebují mít neustále přehled o množství vytěženého dřeva v lese pro přesné plánování odvozu dřeva odběratelům. Prodej dřeva totiž probíhá většinou v jejich režii a zisky z něj jsou důležitějším příjmem nežli platby za provedené práce.

Lesní porosty většiny dřevin se mýtí ve věku okolo sta let (výjimkou je například dub a buk, který má mýtní dobu 120 let), převážně holosečným způsobem. Při tom vznikají paseky, na kterých probíhají následné práce. V mladších porostech se provádí většinou 1x za 10 let probírky, při kterých se porosty prořeďují a odstraňují z nich nežádoucí dřeviny (bříza, jeřáb,…) a nekvalitní nebo poškození jedinci. Stromy se těží buď pomocí motorových pil a následně přibližují koňmi a traktory nebo speciálními těžebními stroji – harvestory a přibližuje se vyvážecími soupravami. Jednotlivé stromy se pokácí, odvětví a rozřežou na požadované délky (tzv. sortimenty) dle potřeb odběratelů. Každému tak vzniklému výřezu stromu se přiřadí evidenční a hmotové číslo, které se na něj vyrazí číslovačkou a následně se zaeviduje do číselníku. Hmotové číslo se najde v tabulkách podle délky a průměru výřezu a druhu dřeviny. Každá dřevina má při stejném středovém průměru a stejné délce jinou hmotu v $m^{3.}$ Hmota je vypočítaná podle takzvané výtvarnice – tvaru kmene, který není ideálním válcem. Proto jsou vypracované kubírovací tabulky podle jednotlivých dřevin, délky a průměru, které se pro zjištění hmotového čísla běžně používají.

Výřezy pokácených stromů se evidují ve výkazu skladu a to na lokalitě „P“ a „V“ podle porostů a na lokalitě „OM“ podle místních názvů skládek. Dokud se výřezy nepřiblíží, vedou se ve výkazu, na lokalitě „P“ - leží na místě, kde byly pokáceny (u pařezu). Pokud se k jednotlivým kusům z terénních důvodů nemůže dostat traktor, který by je odtáhl, přibližuje se koněm, na takzvanou lokalitu „V“, kam již může zajet traktor. Ten nadále přiblíží kusy na odvozní místo – lokalita „OM“. Těchto lokalit bývá na lesnických úsecích velké množství. Zde se dříví roztřídí podle jednotlivých sortimentů na návaly.

Výkaz skladu slouží ke konci roku i při provádění inventury dřeva. Množství vytěženého dříví musí souhlasit na zásoby v lese a množství odvezeného dříví. Hajný je za zásoby dřeva zodpovědný.

Správná manipulace (rozřezání) a následné zatřídění jednotlivých výřezů má rozhodující vliv na finanční efekt celého podnikání. Následně z odvozních míst probíhají dodávky k různým odběratelům podle kvality dřeva. Nejkvalitnější kusy se odvážejí na výrobu dýh a hudebních nástrojů. Další kvalitní kusy se odvážejí na truhlářské řezivo a méně kvalitní na stavební řezivo. Ještě horší kusy se používají na výrobu papíru a dřevotřískových desek. Nejhorší sortiment, to znamená slabé špičky nebo nahnilé kusy, se prodávají lidem jako palivové dříví nebo se drtí ve štěpkovačích na štěpku, která se následně spaluje v elektrárnách jako biomasa.

Pro větší přehlednost a lepší evidenci je les rozdělený na menší celky, na oddělení, dílce a porosty (např. u porostu *815B12* je *815* oddělení, *B* dílec a *12* vlastní číslo porostu, představující zároveň věkový stupeň porostu). Porost je základní hospodářská jednotka. Lesní porosty mají různé velikosti plochy (od desetin až po desítky hektarů). Důležité je, aby se dalo na nich hospodařit stejným způsobem.

Všechny porosty jsou zapsané v hospodářské knize, která se tvoří současně s lesnickými mapami při výrobě desetiletého lesního hospodářského plánu. Ten není plně

závazný a hajný má možnost některé činnosti provést podle svého názoru. Pro větší přehlednost jsou porosty na lesnických mapách rozlišeny různými barvami podle stáří. V plánu je uveden základní předpis prací v jednotlivých porostech (těžby mýtní, předmýtní a prořezávky). Z něj se následně vytvářejí projekty pro jednotlivé roky. Dříve se do hospodářských knih provedené práce zapisovaly ručně, nyní již pomocí počítačových programů. Důležité údaje o jednotlivých porostech je tak možno dohledat desítky let dozadu a zjistit třeba původ osiva, ze kterého byly porosty obnoveny, jejich přesný věk, kolik se v kterém porostu již vytěžilo dřeva…

Při těžbě se musí v číselníku evidovat každý strom i výřez z tohoto stromu podle porostů. Dřevaři si píší číselník na papír, kde si zaznamenávají dřevinu, délku, průměr, vyznačí oddenkové kusy (1 oddenek = 1 strom) a následně jej předají hajnému, který číselník zpracuje na počítači. Po těžbě se pro každý porost v číselníku vytváří sumář podle dřevin s výpočtem množství oddenků a průměrných hmotnatostí. Na základě číselníku společnosti účtují LČR cenu za těžební činnost, kterou pro ně vykonali. Dále z něj LČR vypočítají cenu za dříví, které si od nich společnosti kupují. Číselník je tak prvotní a nejdůležitější doklad evidence dřeva. Podle něj jsou také placeni dělníci, a je to vstupní údaj při tvorbě výkazu skladu.

Po vytěžení a vytahání dřeva zbývá na pasece klest, který je potřeba odstranit. To se provádí buď pálením, kupením klestu na kupy nebo drcením drtičem. Množství klestu se platí podle vytěžených m^3 dřeva, které se počítá z číselníku. Proto je důležité propojení těžby dřeva s odstraněním klestu.

Na vzniklých holinách následně probíhá výsadba stromků různých dřevin. Plocha pasek vzniklých při těžbě se automaticky převádí do plochy zalesnění, vyžínání a nátěrů. Projekt zalesnění vypracovávají LČR na základě lesních typů (rozhoduje nadmořská výška, vlhkost, typ půdy,…) a místních specifických podmínek. Zákon o lesích také přikazuje, kolik procent z vysázených stromků musí být tzv. melioračních a zpevňujících dřevin (buk, dub, jedle,…), aby nevznikly smrkové monokultury. Ty jsou výrazně méně odolné proti škůdcům (přemnožení kůrovců, bekyně mnišky,…), proti živelným kalamitám – polomy, vývraty a hůře plní další funkce lesa – vodohospodářskou, půdoochrannou, rekreační, aj.

O tyto nově vysázené kultury je třeba se starat. Pravidelně je vyžínat – to znamená odstraňovat buřeň (trávu, keře, nežádoucí dřeviny) pomocí křovinořezu nebo kosy. Většinou se provádí jednou ročně v letním období po dobu přibližně pěti let, nebo podle potřeby i delší období.

Dále je potřeba před zimním obdobím stromky natřít repelenty proti okusu zvěří, kdy hlavně srnčí a jelení zvěř na mladých porostech škodí. Nátěry se provádí stejně jako vyžínání každoročně zhruba po dobu 5 let. Množství repelentů se objednává podle celkové plánované plochy nátěrů v množství 20 kg na hektar.

Zhruba ve věku 10 let porostu probíhá první prořezávka, další dle potřeby každých 5 až 10 let až do 30 let věku porostu. Při ní se porost prořeďuje, odstraňují se nežádoucí dřeviny, slabé či poškozené stromky, u kterých je pravděpodobné, že se z nich nevypěstuje kvalitní dříví. Toto dříví se dále nezpracovává, ale nechává se na místě zetlít. Případně se levně prodá zájemcům v takzvané samovýrobě, většinou na palivo.

Ve věku nad 30 let probíhají v porostu zhruba po každých deseti letech probírky, jejichž hlavní význam je výchova. Vytěžené dříví se již přibližuje, odváží a prodává. V mladších probírkách zisky za dříví sotva pokryjí cenu za práce, starší probírky jsou již výdělečné. To se provádí až do mýtního věku, kdy se celý porost opět vytěží holosečným způsobem.

Les si musí na sebe vydělat a ještě přinést státu a komerčním firmám v něm hospodařícím zisk. Rozhodující příjmy jsou z prodeje dřeva. Ostatní činnosti jsou z tohoto prodeje dotovány.

3 Analýza aplikace

3.1 Důvod vzniku aplikace

Jak již bylo řečeno, na trhu chybí aplikace podobného rozsahu, která by dokázala komplexně plánovat a evidovat popsané úkony a dávat tak hajnému nejen přehled o plánu a o pracích ve všech porostech, ale i o výkazu skladu a tím možnostech prodeje dřeva. Je potřeba, aby jednotlivé činnosti byly provázané a navazovaly na sebe. Tím by se mohla práce hajného výrazně zjednodušit, urychlit a zefektivnit. Zároveň aby aplikace měla jednotné grafické prostředí pro zadávání veškerých prací. Aplikace je tvořena nejen pro jednoho hajného, ale pro celou lesní společnost, která většinou obhospodařuje více lesních celků, na kterých pracuje více hajných. Díky tomu by měla společnost přehled o aktuálně prováděných pracích a jejich stavu, zásobách dřeva podle lokalit a tím měla možnost přesouvat lidi a prostředky podle potřeby odběratelů.

Aplikace, která by zvládla zaznamenat a zdokumentovat práci hajného, by byla příliš rozsáhlá v rámci bakalářské práce a zabrala by mnoho hodin programování. Proto jsem musel zúžit rozsah aplikace a celou práci hajného tak zjednodušit. Základní zjednodušení je zmenšení počtu vstupů. Dále byla zjednodušena práce s jednotlivými dělníky, nebyla zapracována návaznost na finanční odměny za jednotlivé práce.

Celá práce tak naznačuje, jakým směrem by se měl budoucí program ubírat. Dá se použít jako případně další informativní a kontrolní program pro hajného.

3.2 Cíle aplikace

Ukázat a zdokumentovat jednotlivé práce hajného. Zlepšit centrální správu takto nashromážděných dat v rámci jedné společnosti. Usnadnit evidenci výroby ve všech činnostech hajného a usnadnit porovnávání provedených prací se sestavami od LČR, podle kterých probíhají platby za práce a dříví.

Aplikace bude mít tři uživatelské role:

- *Administrátor* – jeho úkol je správa jednotlivých uživatelů.
- *Hajný* – sledování a dokumentování prací v lesních porostech a přehledného stavu zásob dřeva.
- *Šéf* – vedoucí může nahlížet na jednotlivé práce hajných a stavu rozpracovanosti jednotlivých porostů. Nemůže však nic upravovat a měnit jejich záznamy.

Aplikace by postupně měla sledovat a zaznamenávat v roli *hajný*:

- Plánování těžby v jednotlivých porostech.
- Provádění těžby - tedy zaevidování jednotlivých pokácených stromů a jejich výřezů do číselníku. Spočítání jejich hmotnatosti. Na základě těchto údajů vytvořit sumáře za porost.
- Přibližování dříví na všech lokalitách.

- Odvoz dříví z lesa - evidovat postupné vytahání dříví z lesa, zaevidování jednotlivých prací spojené s odvozem dříví.
- Výkaz skladu – přehled o zásobách dříví v lese. Kde a kolik dřeva se aktuálně nachází na jednotlivých lokalitách (na „P“, na „V“ na „OM“).
- Likvidace klestu – zaevidování kolik se na pasece vytěžilo dříví, kolik z toho uklidilo klestu a jakým způsobem. Kolik klestu zbývá ještě uklidit.
- Zalesnění – naplánování kolik se na dané pasece vysází jakých stromků a následně kolik stromků se vysázelo. Dále přehled potřeby jednotlivých druhů stromků na celém lesnickém úseku pro objednání v lesních školkách.
- Vyžínání – po dobu několika let se musí nový porost pravidelně vyžínat proti buřeni. Je třeba mít přehled, které porosty jsou již vyžnuté a kolik ještě zbývá v daném roce vyžnout. Předem se přesně neví, kolik let se bude daná paseka vyžínat.
- Nátěry stromků – stejně jako u vyžínání se několik let stromky natírají proti okusu zvěří. Je zde vidět plán, skutečnost a zbývající úkol.
- Prořezávky – zhruba od 10 let probíhají prořezávky, každých 5 až 10 let, do 30 let stáří porostu.

V roli *šéf* by aplikace měla umožňovat přístup k veškerým záznamům, ale bez možnosti jejich editace. Podporovat různá filtrování a vyhledávání. Tímto bude mít vedoucí přehled o zbývajících úkolech a má možnost podle potřeby přesouvat pracovníky a výrobní prostředky mezi jednotlivými lesnickými úseky.

4 Použité technologie

Jelikož se jedná o webovou aplikaci, byl zvolen jazyk na tvorbu webových stránek XHTML formátovaný CSS styly.

Výběr programovacího jazyka na straně serveru taktéž nepředstavoval problém. Rozhodl jsem se pro jazyk PHP, jelikož se jedná o nejrozšířenější jazyk pro tvorbu webových aplikací a také je jediný, který jsem doposud používal. Místo něj je možno použít např. ASP.NET, s kterým jsem se zatím v praxi nesetkal. Po delším uvážení jsem se rozhodl navíc ještě obohatit stránky programovacím jazykem na straně klienta. Jelikož jsem nikdy programovací jazyky na straně klienta nepoužíval, zvolil jsem nejpoužívanější a nejznámější – JavaScript.

Z databázového pohledu jsem se rozhodoval mezi MySQL a Oracle. Nakonec jsem upřednostnil databázi od společnosti Oracle. Samotný SQL jazyk je v dnešní době standardizovaný a většina dotazů je pro tyto databáze stejná. Proto by neměl být problém po změně přihlašovacích funkcí k databázi a několika nestandardních prvků z PL/SQL, který není standardizovaný, aby aplikace pracovala s jinými databázemi.

4.1 (X)HTML

HTML (HyperText Markup Language) je značkovací jazyk pro tvorbu hypertextových dokumentů vyvinutý původně pro propojení informačních systémů v CERN. Dneska je spravován konsorciem W3C[1].

Jednoduchý popis jazyka: HTML dokument je textový soubor, který se skládá z textů a HTML značek (tagů). Tagy určují, jak bude daný text zobrazený nebo popisují jednotlivé prvky stránky. Tag se skládá z názvu uzavřeného do ostrých závorek. Případně můžou mít tagy ještě atributy s hodnotami oddělené mezerami, které dovysvětlují daný tag. Tagy jsou obvykle párové, tedy existuje otevírací tag *<otevírací_tag atribut="hodnota">*, a uzavírací tag *</uzavírací_tag>* . Otevírací a uzavírací tag spolu s obsahem tvoří element. Významy jednotlivých tagů lze vyčíst na stánkách konsorcia W3C (aktuální verze 4.01 [2])

Struktura dokumentu:

Jednotlivé elementy se postupně do sebe zanořují a vzniká tak stromová struktura dokumentu. Dokument začíná direktivou doctype, která určuje typ dokumentu. Následuje kořenový element *<html>*, obsahující element hlavičky *<head>*, kde jsou uloženy metadata o dokumentu a element *<body>*, který obsahuje vlastní text dokumentu.

Příklad HTML stánky

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

[1] http://www.w3.org
[2] http://www.w3.org/TR/REC-html40

```
    <!-- metadata -->
    <title>titulek HTML stánky</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
  </head>
  <body>
    <!--tělo dokumentu -->
    <h1 color=red class="nadpis">Nadpis 1. Úrovně</h1>
    <hr>
    <p><b>Tučný text</b>, normální text odstavce dokumentu.</p>
  </body>
</html>
```

XML (Extensible Markup Language) je obecný jazyk pro vytváření dokumentů obsahující strukturovaná data.

XML dokument se podobá HTML dokumentu. Jednotlivé prvky se obalují tagy, stejně jako v případě HTML. Veškeré tagy jsou však povinně párové. Tagy mohou též obsahovat atributy. Každý atribut pak musí mít přiřazenou hodnotu, která je uzavřena do uvozovek. Samotný XML definuje jen syntaktická pravidla a názvy tagů a atributů si tak musíme zvolit vlastní, které budou nejlépe vystihovat význam dané části textu. Tagy dokumentu obsahují významové informace a ne informace jak bude daný text prezentován. Dokumenty jsou tak informačně bohatší a mají tak širší možnosti uplatnění. Jejich styl zobrazení se definuje pomocí stylových jazyků. Díky tomu můžeme nejen jedním stylem zobrazit více dokumentů, ale i jeden dokument zobrazit pomocí více stylů a měnit tak jeho vzhled. Jedním ze stylových jazyků je i CSS popsaný níže.

Díky vlastním značkám budou dokumentu rozumět jen aplikace šité danému typu dokumentů na míru. Jednotlivé tagy je však možné definovat a popsat v DTD (*Definice typu dokumentu*). Ten vymezuje použití, způsob vnoření uspořádání tagů. Dále vymezuje povolené atributy pro daný tag. Pro některé standardní aplikace je již DTD vytvořen a popisuje význam jednotlivých značek. Jeho dodržováním je pak zajištěna správná interpretace dat v aplikaci.

XML dokument musí obsahovat prolog s XML verzí, kódováním dokumentu a jeden hlavní (kořenový) element.

Jednoduchý příklad XML dokumentu

```
<?xml version="1.0" encoding="UTF-8"?>
<dokument autor="Ondřej Švec">
  <nadpis>XML dokument</nadpis>
  <datum>1. 1. 2000</datum>
  <clanek>text článku v elementu clanek</clanek>
</dokument>
```

XHTML (eXtensible HyperText Markup Language) je rozšířený značkovací jazyk pro tvorbu hypertextových dokumentů vyvinutý konsorciem W3C. Jedná se o reformulaci jazyka HTML jako aplikaci XML. XHTML je tedy nástupce HTML založený na XML.

Jelikož se XML od HTML výrazně neliší, neliší se ani XHTML od HTML. XHTML žádné nové tagy nepřináší, obsahuje vlastně jenom omezení, které přináší XML.

Hlavní rozdíly oproti HTML:

- XML prolog s verzí a kódováním XML dokumentu
- nutný doctype dokumentu s typem XHTML a adresou DTD
- veškeré tagy psát pouze malými písmeny
- povinnost každý tag ukončit (i nepárový tag, ten poté končí lomítkem)
- všechny hodnoty atributů se uzavírají do uvozovek
- všechny atributy musí mít hodnotu
- jednotlivé tagy se nesmí křížit
- je upuštěno od veškerých formátovacích tagů, formátování se provádí pomocí CSS.

Podrobnější seznámení s (X)HTML můžeme nalézt na stránkách *jakpsátweb.cz* [3].

4.2 CSS

XML dokumenty a dokumenty od XML odvozené se zabývají popisem dat, ale nikoliv jejich vzhledem. Způsob zobrazení je definovaný pomocí kaskádových stylů, známých pod zkratkou CSS (Cascading Style Sheets).

Dříve se v HTML používaly k formátování textu HTML tagy (např. *tučný text*). Což ale znepřehledňovalo text dalšími informacemi, které s daným dokumentem přímo nesouvisely a velice komplikovaly případnou změnu designu stránek a celkově jejich jednotný design. Odtržením formátovacích pravidel od textu se dokument zpřehlednil a veškeré úpravy se dají měnit pro celý dokument/skupiny dokumentů jednotně z jednoho místa.

CSS dokument se většinou umísťuje do samostatného souboru s koncovkou *.css* a přilinkuje se do HTML kódem uloženém v hlavičce:

```
<link href="soubor.css" rel="stylesheet" type="text/css" />
```

Například pro formátování textu uvnitř *tagu text * na text zelený a podtržený, použijeme CSS styl:

Příklad CSS stylu

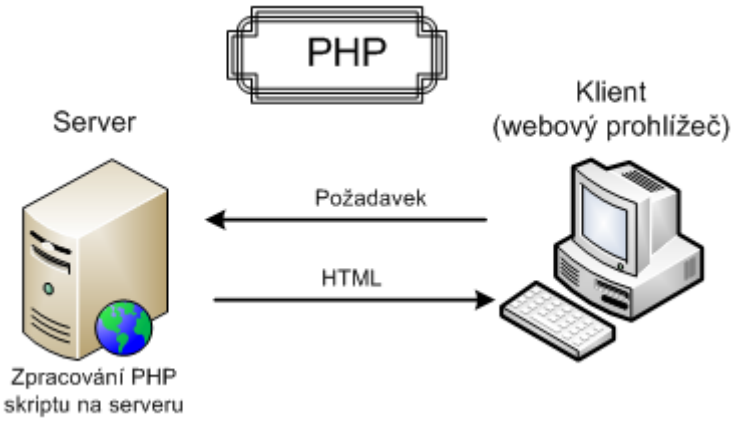
```
.nadpis {
    text-decoration: underline; /* podtržení */
    color: green;     /* zelená barva (případně #00FF00)*/
}
```

Více o CSS lze přečíst na stránkách projektu *jakpsátweb.cz* [4] nebo konsorcia W3C [5].

[3] www.jakpsatweb.cz/html
[4] http://www.jakpsatweb.cz/css

4.3 PHP

PHP (PHP: Hypertext Procesor) je skriptovací programovací jazyk, který se zpracovává na straně serveru.



Obrázek 1 - Princip zpracování PHP

PHP je specializován na tvorbu webových stránek, a tedy skripty se dají snadno začlenit přímo do HTML kódu a tím měnit strukturu stánek, zpracovávat formuláře, zabezpečit přístup na stránku, skládat stránku z dílčích souborů aj. Navíc výborně spolupracují s většinou databázových serverů.

Díky zpracování dat vzdáleně na serveru klient nevidí PHP skripty a ani k nim nemá přímý přístup. Jen odešle serveru požadavek (například vyplněný formulář), server jej zpracuje a vrátí odpověď.

Vzhledem k tomu, že PHP je zpracováván na straně serveru, nevystačíme si pouze s prohlížečem jako u HTML, ale je potřeba nainstalovat server s podporou PHP, který bude skripty zpracovávat. Obvykle se jako server používá Apache[6] s PHP modulem. Pro jeho snadnou instalaci a nastavení lze použít např. program XAMPP[7], který má vše již nastavené a není problém jej jednoduše nainstalovat a zprovoznit.

PHP skripty se vpisují přímo do HTML stánky mezi značky *<?php* a *?>*. HTML stránka pak mívá typicky koncovku *.php*, podle nastavení serveru. Tím server pozná, že stránka obsahuje PHP skript a ten zpracuje. Základní příkaz je příkaz *echo*, který pošle klientovi zadaný text.

Příklad PHP skriptu

```
<h1>HTML stránka</h1>
<?php echo "<p>text odeslaný pomocí PHP</p>"; ?>
```

[5] http://www.w3.org/Style/CSS
[6] www.apatche.org
[7] www.apachefriends.org/en/xampp.html

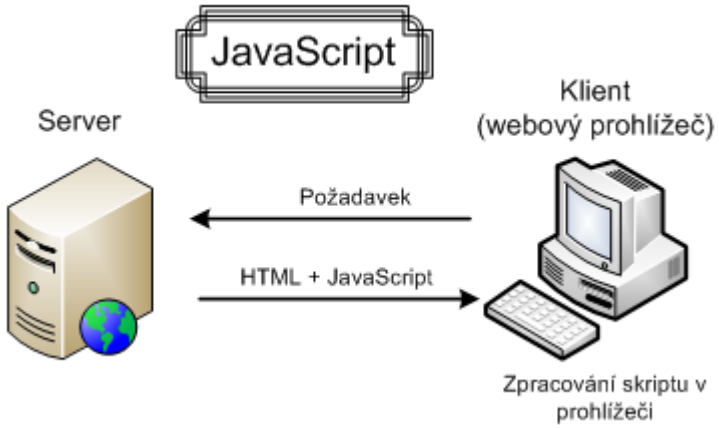
Server skript vyhodnotí a odešle klientovi již jen čistý HTML kód:

```
<h1>HTML stránka</h1>
<p>text odeslaný pomocí PHP</p>
```

Další příkazy a příklady PHP kódu lze nalézt na stránkách projektu PHP [8].

4.4 JavaScript

JavaScript je skriptovacím jazykem, který se používá na webových stránkách. JavaScript se vykonává na straně klienta přímo v prohlížeči. Klient zašle webserveru požadavek, server vrátí odpověď jako HTML kód spolu s kódem v JavaScriptu a ten se vykoná až na straně klienta buď přímo nebo na základě vyvolaných událostí. Tím se stávají stránky dynamické. Nevýhodou JavaScriptu je jeho závislost na podpoře v prohlížeči, který ho vykonává. Dnešní moderní prohlížeče jej zvládají bez větších problémů. Dále je třeba zmínit, že JavaScript se dá v prohlížeči vypnout a tím jeho funkce budou ignorovány.



Obrázek 2 - Princip zpracování JavaScriptu

Díky zpracování skriptu až u klienta nemůže JavaScript přistupovat k dokumentům na serveru ani do databáze, nezná práce se sítí. Z důvodu bezpečnosti nemůže ani přistupovat lokálně na disk. Tím je jeho rozsah dosti omezen. Jen může donutit prohlížeč načíst zadanou URL adresu.

JavaScript se používá pro oživení webových stránek, lepší komunikaci a interakci s uživatelem. Pomáhá tak zpříjemnit uživateli prohlížení webových stránek. Lze jej použít pro kontrolu formulářů, na akce vyvolané událostmi (např. najetí na prvek, zmáčknutí tlačítka, zatrhnutí zaškrtávacího políčka, opuštění vstupního pole, …)

JavaScript můžeme do dokumentu vložit buď jako speciální soubor s příponou *.js* a přilinkovat ho do HTML souboru pomocí zápisu:

[8] www.php.net

```
<script language="JavaScript" type="text/javascript"
src="soubor.js"></script>
```

Nebo jej napsat přímo do dokumentu mezi značky *<script></script>*. Jako příklad jsem si vybral jednoduchý skript pro vepsání textu do HTML stránky. Tedy stejnou akci, kterou jsem již psal pomocí jazyka PHP:

Příklad JavaScriptu

```
<h1>HTML stránka</h1>
<script>
    document.write("<p>text odeslaný pomocí PHP</p>");
</script>
```

Další seznámení s JavaScriptem můžeme nalézt na stránkách *javascript.cz* [9].

4.5 DOM

DOM (Document Object Model) je API, pomocí kterého můžeme pracovat se samotnou webovou stránkou a přistupovat přímo k jednotlivým objektům XHTML (XML) dokumentu a dále s nimi pracovat. Objektem může být *dokument*, který představuje dokument jako celek, objekt *tabulka*, který představuje XHTML prvky tabulky atd. Pomocí již popsaného JavaScriptu a DOM je možné dynamicky přidávat, mazat či měnit jakýkoliv obsah na XHTML dokumentu, aniž by se musela daná stránka odeslat zpracovat na server a změněná znovu načíst. Tím je zpříjemněna práce s webovou aplikací. Díky stromové struktuře dokumentu dovedeme rozlišit nadřazené nebo podřazené elementy a následně k nim pak přistupovat. Např. *document.form[1].name* vrací jméno druhého formuláře a *dokument.form[1].submit()* tento formulář odešle ke zpracování na server. Následné zpracování se již provádí programovacími jazyky, jako je PHP, na serveru.

DOM je stejně jako u JavaScriptu funkcí webového prohlížeče a ne všechny jeho funkce jsou podporovány na všech prohlížečích. Většina moderních prohlížečů se již snaží dodržovat doporučené standarty vydané pracovní skupinou W3C DOM [10].

4.6 Databáze

4.6.1 Popis databáze

Databáze je určitá uspořádaná množina dat, uložená na paměťovém médiu. Databázi v počítačovém světě si můžeme představit taktéž jako softwarové vybavení, které umožňuje efektivní a spolehlivé ukládání těchto dat, přístup k datům a i jejich manipulaci. Tvoří tedy rozhraní mezi aplikačními programy a uloženými daty. Tento software se sice odborně nazývá DBMS (Database Management Systém neboli systém řízení báze dat), ale v běžné terminologii se obecně používá jen slova databáze. V této práci je slovem databáze myšleno DBMS.

[9] http://www.javascript.cz
[10] http://www.w3.org/DOM

Většina moderních databází jsou relační databáze založených na teorii množin a predikátové logice. Na data nahlížíme jako na tabulky. Každý sloupec (atribut) má svůj název a obsahuje data stejného typu. Na řádky tabulky nahlížíme jako na záznamy a sloupce chápeme tak, že uchovávají informace o relacích mezi jednotlivými záznamy. Jednotlivé tabulky můžeme podle klíčů propojovat a získat tak záznamy, které potřebujeme.

Tabulková data pro využití v relační databázi mají několik základních pravidel:

- Každá tabulka má svůj jedinečný název.
- Každý řádek (záznam) odpovídá jedné entitě relace.
- Každý řádek by měl mít jedinečný identifikátor (klíč), který jednoznačně určí příslušný záznam.
- Žádné záznamy nebudou stejné, duplicita je nežádoucí.
- Každý sloupec (atribut) má jedinečný název.
- Hodnoty ve sloupcích jsou atomické a už je tedy nelze dále logicky rozdělit.
- Sloupec je vždy jednoho konkrétního datového typu.
- Pořadí atributů a záznamů je nepodstatné.

Příklad uložení dat v relační databázi, tabulky se můžou následně propojit pomocí sloupce *ID adresy*:

Tabulka 1 - Ukázková databázová tabulka Lidé

ID	Jmeno	Prijmeni	Pohlavi	Vek	ID_adresy
35534	*Jan*	*Pavel*	*m*	*12*	*235*
56721	*Tomáš*	*Knot*	*m*	*45*	*100*
59087	*Jana*	*Knotová*	*ž*	*40*	*100*

Tabulka 2 - Ukázková databázová tabulka Adresy

ID_adresy	Ulice	CP	Obec	PSC
100	*Nová*	*56*	*Nové Hory*	*20560*
235	*Stará*	*120*	*Neznámá*	*23231*

Pro většinu dnešních databází se používá jazyk SQL, pomocí kterého pracujeme s daty v databázi. Ty se dělí do tří základních skupin:

1. Příkazy pro definici dat (DDL) – těmito příkazy vytváříme, měníme a rušíme databázové objekty (tabulky, pohledy, indexy,…)

Příklad DDL příkazu: vytvoření tabulky lidé

```
CREATE TABLE lide(
id Number(5,0) NOT NULL, jmeno NVarchar2(20), prijmeni NVarchar2(20),
pohlavi char, vek Number(3,0), id adresy Number(5,0)
)
```

2. Příkazy pro řízení dat (DCL) – těmito příkazy nastavujeme přístupová práva, řízení provozu a údržbu databáze.

Příklad DCL příkazu: uživateli *uzivatel* přidělena práva pro vytváření tabulek

```
GRANT CREATE TABLE TO uzivatel;
```

3. Příkazy pro manipulaci s daty (DML) – sem patří příkazy pro vkládání, aktualizaci, mazání a výběr dat z tabulek.

Příklad DML příkazu: výběr dat z tabulek spojených pomocí klíče id_adresy

```
SELECT jmeno, prijmenni, ulice cp, obec FROM lide, adresy WHERE
lide.id_adresy=adresy.id_adresy
```

4.7 Normalizace databázových tabulek

Normalizace tabulek je proces, při kterém se tabulky a jejich atributy rozkládají za účelem jednodušší a přesnější práce s daty, jejich lepší manipulací a zabránění redundance dat (opakování). Normalizace vede k efektivnějšímu ukládání dat, ale neznamená zvýšení výkonu databáze.

Normalizace se provádí pomocí takzvaných normálních forem. Platí, že čím je tabulka ve vyšší normální formě, tím kvalitněji je tabulka navržena.

Nultá normální forma

Tabulka je v nulté normální formě, existuje-li pole, které obsahuje více než jednu hodnotu.

Tato forma odpovídá nenormalizovanému modelu. Pokud tabulka není v nulté normální formě, je alespoň v první normální formě.

První normální forma

Tabulka je v první normální formě, jestliže v každém poli je pouze jednoduchý datový typ, tedy pole je atomické (dále již nedělitelné) a jednotlivé atributy se neopakují.

Druhá normální forma

Tabulka je ve druhé normální formě, jestliže je v první normální formě, existuje primární klíč a každý neklíčový atribut je plně závislý na primárním klíči.

Třetí normální forma

Tabulka je ve třetí normální formě, jestliže je ve druhé normální formě a všechny neklíčové atributy jsou vzájemně závislé.

Boyce-Coddova normální forma

Tabulka je v Boyce-Coddově normální formě, když pro dvě množiny atributů A a B platí: A –> B a současně B není podmnožinou A, pak množina A obsahuje primární klíč tabulky

Čtvrtá normální forma

Tabulka je ve čtvrté normální formě, je-li ve třetí normální formě a pokud atributy v ní obsažené popisují pouze jeden fakt nebo jednu souvislost.

Pátá normální forma

Tabulka je v páté normální formě, pokud je ve čtvrté normální formě a pokud by se přidáním libovolného nového atributu rozpadla na více tabulek.

4.8 Použité aplikace

NetBeans IDE [11]

Vývojové prostředí NetBeans IDE 6.8 od společnosti Sun Microsystems jsem zvolil jako hlavní editor pro tvorbu XHTML dokumentu, zápis CSS stylů a tvorbu PHP a JavaScript kódů. Jeho hlavní výhodou je open source licence GPL v2 a CDDL [12], multiplatformnost, jednoduché a přehledné rozhraní, výborná práce s technologiemi HTML/PHP/CSS jako je zvýrazňování kódu, doplňování kódu, kontrola chyb.

Toad Data Modeler [13]

Toad Data Modeler 3 posloužil pro návrh a přehlednou vizualizaci databázových tabulek. Následně jsem z něj vygeneroval SQL dotazy, které mi posloužily k vytvoření tabulek, jejich atributů a vztahů mezi tabulkami.

Sqldeveloper [14]

Sqldeveloper vyvinutý firmou Oracle je vývojové prostředí pro databáze Oracle. Slouží pro vývoj, testování a ladění SQL dotazů a PL/SQL objektů.

[11] http://netbeans.org
[12] http://netbeans.org/cddl-gplv2.html
[13] http://www.toadsoft.com/toaddm/toad_data_modeler.htm
[14] http://www.oracle.com/technology/products/database/sql_developer

XAMPP [15]

XAMPP je balík aplikací pro webový server, který lze velice jednoduše nainstalovat. Nepotřebuje žádné dodatkové konfigurace. Obsahuje Apache server s PHP modulem a další programy pro tento projekt nepotřebné. XAMPP je šířen pod svobodnou GPL licencí.

Oracle databáze [16]

Databáze Oracle 10g byl použit jako hlavní databázový server, uchovávající databázové informace aplikace.

PSPad [17]

PSPad je jednoduchý, ale robustní textový editor používaný pro rychlé a jednoduché úpravy krátkých textů. Navíc podporuje zvýrazňování syntaxe jazyků HTML, PHP tak i SQL.

Gimp [18]

Gimp je grafický editor sloužící k úpravě bitmapových obrázků. Jedná se o zdarma dostupnou variantu komerčního programu Photoshop. Gimp je k dispozici pod svobodnou licencí GPL. V aplikaci byl použit pro návrh jednoduchého loga a základní grafické prvky.

Webový prohlížeč

Nejdůležitější aplikací je samotný webový prohlížeč, v kterém se samotná aplikace vykresluje a přes který se ovládá.

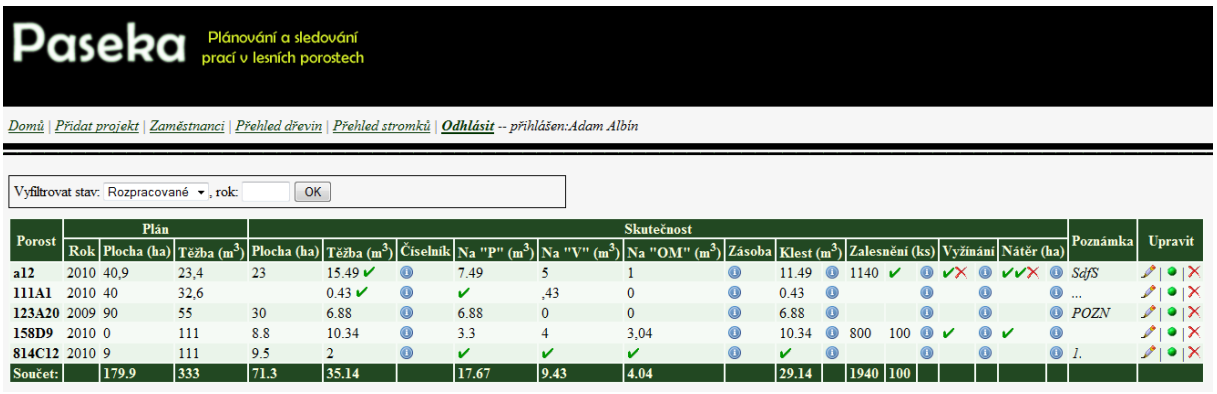
Aplikace byla zkoušena a testována na nejpoužívatelnějších prohlížečích: Firefox 3, Chrome 4 a Internet Explorer 8. Na těchto prohlížečích pracuje aplikace korektně, v nižších verzích nebo jiných prohlížečích nemusí být aplikace plně funkční.

[15] http://www.apachefriends.org/en/xampp.html
[16] http://www.oracle.com/us/products/database
[17] http://www.pspad.com
[18] http://www.gimp.org

5 Návrh aplikace

5.1 Vzhled a rozvržení aplikace

Aplikace má velice jednoduchý vzhled. Použito je jen základní podbarvení bez větších grafických prvků, které by aplikaci jistě oživily. V aplikaci jde spíš o samotnou funkčnost. Pro případný komerční úspěch aplikace by bylo potřeba grafickou úpravu stránek výrazně přehodnotit a značně ji přepracovat.

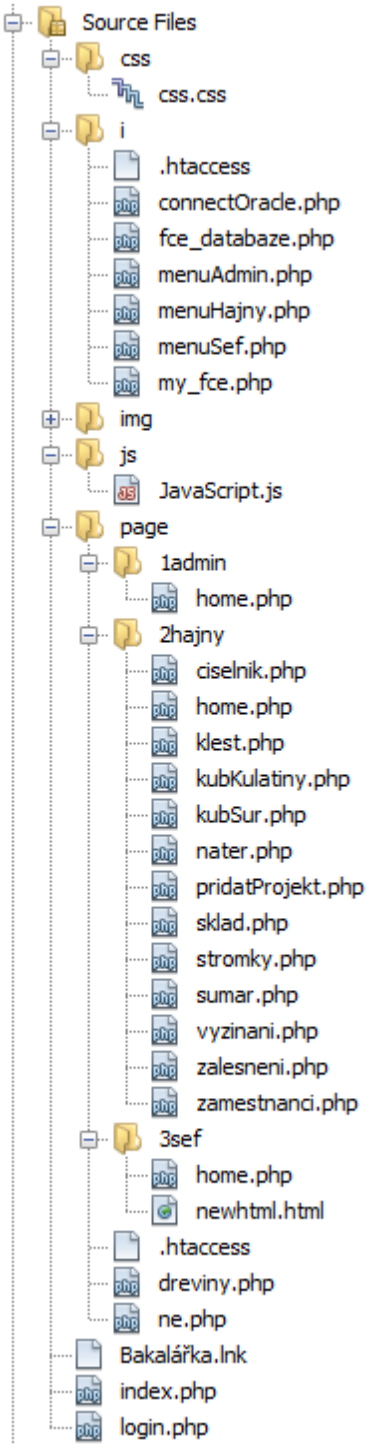


Porost	Plán			Skutečnost												Poznámka	Upravit
	Rok	Plocha (ha)	Těžba (m^3)	Plocha (ha)	Těžba (m^3)	Číselník	Na "P" (m^3)	Na "V" (m^3)	Na "OM" (m^3)	Zásoba	Klest (m^3)	Zalesnění (ks)		Vyžínání	Nátěr (ha)		
a12	2010	40,9	23,4	23	15.49 ✔		7.49	5	1		11.49	1140	✔	✔✕	✔✔✕	*SdfS*	
111A1	2010	40	32,6		0.43 ✔		✔	,43	0		0.43					...	
123A20	2009	90	55	30	6.88		6.88	0	0		6.88					*POZN*	
158D9	2010	0	111	8.8	10.34		3.3	4	3,04		10.34	800	100	✔	✔		
814C12	2010	9	111	9.5	2		✔	✔	✔		✔					*1.*	
Součet:		**179.9**	**333**	**71.3**	**35.14**		**17.67**	**9.43**	**4.04**		**29.14**	**1940**	**100**				

Obrázek 3 - Základní stránka aplikace

Aplikace má jednoduchý layout. Hlavičku tvoří jednoduché logo aplikace, pod ní je horizontální menu, které se liší pro každou uživatelskou roli. Pod horizontální čárou je již vsazeno tělo samotné aplikace, které se mění podle aktuálního požadavku.

5.2 Adresářová struktura



Obrázek 4 - Adresářová struktura

Jednotlivé soubory jsou logicky členěny do adresářové struktury, která pomáhá nejen k přehlednosti, ale i k bezpečnosti aplikace. Pomocí PHP skriptů kontroluji, zda jsou soubory načtené ze správných složek. Dále je možné zabránit přístupu do určitých složek pomocí souboru *.thaccess*. Jedná se o zvláštní soubor, který v určitém webovém adresáři dovoluje autoru webu upravit chování daného adresáře, například zabránění přímého přístupu a zabránění tak čtení těchto souborů přímo v prohlížeči. Přistupovat pak k daným souborům můžeme pouze pomocí funkcí na straně serveru.

V kořenovém adresáři se nachází jen dva soubory. *Login.php*, který se stará o přihlášení do aplikace a hlavní soubor celé aplikace, *index.php*.

V adresáři **css** se nacházejí veškeré soubory obsahující CSS styly, které jsou tak pohromadě a jejich editací lze jednotně měnit grafický styl celé aplikace.

Adresář **i** obsahuje veškeré pomocné funkce a části HTML kódu, ke kterým by uživatel neměl mít přístup a které se následně pomocí PHP funkce *include();* připojují do HTML stránek. Nejdůležitější jsou soubory *connetOracle.php*, kde jsou uloženy přístupové údaje do databáze a *fce_databaze.php*, kde jsou funkce zajišťující komunikaci s databází. Celá tato složka je pro uživatele znepřístupněna pomocí souboru *.htaccess*. Díky tomu se uživatel nemůže dostat k uloženým heslům pro přístup do databáze a k dalším funkcím, které pro něj nejsou užitečné a představovaly by případné bezpečnostní riziko. Dále jsou zde uloženy soubory obsahující menu aplikace k jednotlivým uživatelským rolím.

Adresář **img** ukrývá veškeré grafické prvky používané v aplikaci. Nejčastěji v grafických formátech *.gif* či *.png*.

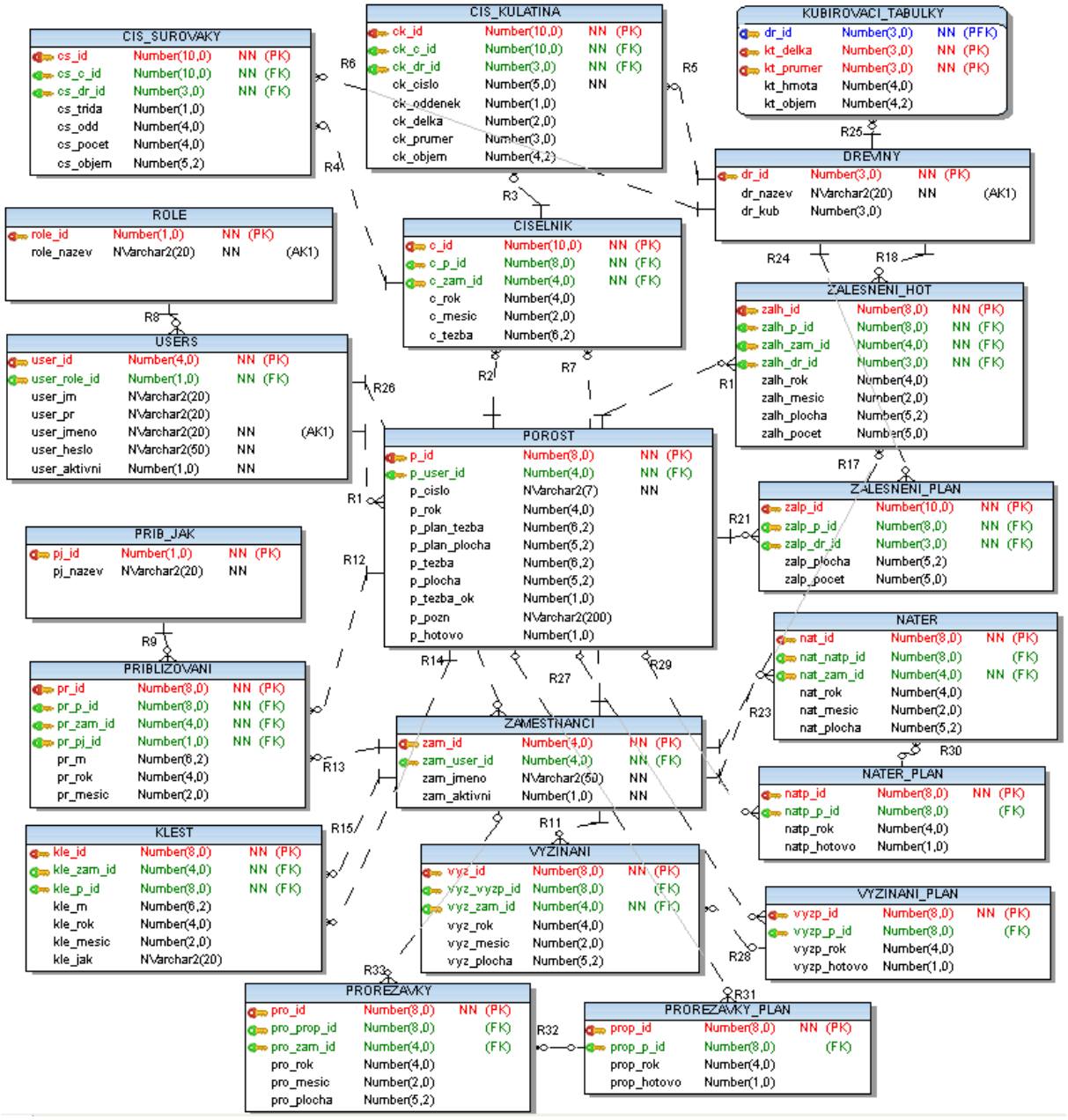
Dále se v adresářové struktuře nachází adresář **js**, který uchovává JavaScriptové soubory s příponou *.js*.

Poslední adresář **page** v sobě uzavírá hlavní soubory aplikace. Tyto soubory jsou načítány do souboru *index.php* a tvoří tak jednotlivé stránky aplikace. Tyto soubory obsahují veškerý HTML kód, tak i funkční logiku dané stránky. Adresář je rozdělen na další podadresáře podle jednotlivých rolí aplikace. Aplikace si hlídá, aby každá role načítala soubory pouze ze svého adresáře nebo soubory uložené přímo v adresáři page, které jsou přístupné

všem rolím. Jednotlivé soubory se nahrávají do hlavního souboru *index.php*. Přímý přístup do této složky je opět zakázán pomocí souboru *.htaccess*.

5.3 Návrh databáze

5.3.1 ER diagram



Obrázek 5 - Entity-relationship diagram

5.3.2 Stručný popis tabulek

Atributy (jména sloupců) jednotlivých tabulek vždy začínají zkratkou jména tabulky. Dále každá tabulka obsahuje identifikační číslo názvu *zkratkaTabulky_ID*, které jednoznačně identifikuje každý řádek tabulky a je tak primárním klíčem dané tabulky. Tyto id jsou vkládány pomocí sekvencí pro každou tabulku, čímž se zamezí jejich opakování.

Role – uchovává názvy přípustných uživatelských rolí.

Users – tabulka obsahuje veškeré informace o jednotlivých uživatelích: Jméno a příjmení, přihlašovací jméno a heslo, typ role uživatele a jestli je aktivní. V kladném případě se může uživatel přihlásit do systému. Jistě by se daly přidat další osobní údaje, ale prozatím by neměly uplatnění.

Zamestanci – zde se zaznamenávají informace o jednotlivých zaměstnancích pracujících pro hajného. Zaměstnanec je charakterizován jen číslem hajného, pod kterým pracuje, jménem a zda je zaměstnanec stále aktivní a pracuje ještě v lese, nebo již ne. Zaměstnanci, kteří již v lese pracovali, se nedají z projektu vymazat, protože je k nim stále přiřazena práce, kterou již vykonali.

Dreviny – tabulka obsahuje jednotlivé dřeviny. Jejich jméno ve zkratce, identifikační číslo, a id dřeviny, podle které se kubíruje v kubírovacích tabulkách.

Kubirovaci_tabulky – pomocí typu dřeviny, délky a průměru zobrazují hmotu a objem daného kusu.

Porost – tabulka shromažďuje informace o jednotlivých porostech. Hlavně číslo hajného, který se o porost stará, číslo porostu, plocha (ha), rok zadání, zda je výchova porostu již ukončena a prostor pro poznámku. Dále se tu vyskytují informace týkající se těžby, které by měly být již ve vlastní tabulce, což jsem si uvědomil bohužel pozdě, kdy oprava by si vyžádala velkou změnu aplikace. Těžba obsahuje plánovanou těžbu (m^3), plochu (ha), skutečnou plochu, skutečnou těžbu, která je automaticky dopočítávána pomocí triggerů z tabulky *cislenik* a informaci o ukončené těžbě.

Ciselnik – Uchovává údaje o jednotlivých číselnících, tedy ke kterému porostu se váže, jaký zaměstnanec porost těží, rok a měsíc těžby a objem těžby, který se automaticky dopočítává z tabulek *cis_kulatiny* a *cis_surovaky* pomocí triggerů.

Cis_kulatina – uchovává informace o jednotlivých kusech (sortimentech). Jako je číslo daného kusu, jestli se jedná o oddenek, délku, průměr a objem vypočítaný pomocí funkce *fce_kubirovani()* z tabulky *Kubirovaci_tabulky*.

Cis_surovaky – údaje o surových kmenech v porostu pro daný číselník. Tedy jejich třídu, typ dřeviny, celkový počet, počet oddenků a objem vypočítaný pomocí funkce *fce_kubirovani_sur()*.

Priblizovani – Ukazuje pohyb dřeva v lese. Zaznamenává id paseky, v které se přibližuje, zaměstnance, typ práce, kde jednotlivé práce jsou uloženy v tabulce **prib_jak**. Dále ještě rok a měsíc vykonání dané práce a objem dřeva, kolik bylo přiblíženo.

Klest – zde se zaznamenávají údaje o uklizeném klestu. O jakou paseku se jedná, jméno zaměstnance, rok a měsíc úklidu, jakým způsobem byl klest uklizen a kolik ho bylo uklizeno.

Zalesneni_plan – zde se plánuje zalesnění paseky, obsahuje id paseky, typ dřeviny, počet stromků a plánovaná plocha, na kterou se mají nasázet.

Zalesneni_hot – sem patří záznamy o již zasazených stromcích. Obsahují stejné záznamy jako tabulka *zalesneni_plan* a navíc informace, kdo je vysázel, rok a měsíc sazby.

Vyzinani_plan – zaznamenává rok vyžínání paseky a zda je již vyžínání ukončeno.

Vyzinani – tabulka uchovává údaje o provedených pracích vyžínání. S údaji id plánu vyžínání, rok a měsíc vyžínání, id zaměstnance a vyžnutou plochu.

Nater_plan – zaznamenává rok, kdy bude třeba natírat stromky na pasece, a zda je již natírání ukončeno.

Nater – tabulka uchovává údaje o provedení nátěru stromků. S údaji id plánu nátěru, rok a měsíc natírání, id zaměstnance a plochu, na které se natíralo.

Prorezavky_plan – plánování, který rok se bude prořezávat a zda je již provedeno.

Prorezavky – údaje o provedení prořezávky, tedy id plánu prořezávky, rok a měsíc vykonání práce, id zaměstnance a plochu, na které se prořezávalo.

6 Popis aplikace

6.1 Přihlášení

Po zadání adresy do webového prohlížeče, kde je aplikace uložena, nás přivítá přihlašovací obrazovka (Obrázek 6). Pro přístup do aplikace je potřeba se přihlásit pomocí uživatelského jména a hesla, přihlašovací údaje nám přidělí administrátor. Sám uživatel si uživatelské jméno a heslo nevybírá a nemůže ani změnit. Bez přihlášení není možné s aplikací nijak pracovat.

Obrázek 6 - Přihlašovací stránka

Z pohledu programátora

O přihlášení se stará jednoduchý kód uložený v přihlašovacím souboru *login.php*, který nás po úspěšném ověření přihlašovacích údajů přesměruje na *index.php*, který se stará o hlavní aplikaci. Přihlašovací heslo je z bezpečnostních důvodů v databázi zakódováno jednosměrnou hashovací funkcí sha1, díky které není možné zpětně získat heslo uživatele. Při případném nabourání do databáze a úniku přihlašovacích údajů útočník nebude mít skutečná přihlašovací hesla a nebude tak moci se přihlásit. V případně zapomenutí hesla není možné zpětně heslo zjistit a je tak nutné, aby administrátor vytvořil heslo nové.

Při přihlašování je kontrolováno přihlašovací jméno, otisk hesla a zda je uživateli stále povolen přístup, tedy je-li aktivní. Po úspěšném ověření se důležité údaje o uživateli přidají do trvalé proměnné *$_SESSION[]*, kde zůstanou uchovány, dokud se uživatel neodhlásí nebo se neuzavře prohlížeč.

Skript pro přihlášení uživatele

```
if (isset($_POST['prihlasit']) && isset($_POST['user']) &&
isset($_POST['pass']))
{
    $user = $_POST['user'];
    $pass = sha1($_POST['pass']); //vytvoření otistku hesla pomocí
sha1 algoritmu
```

```
    if (ereg("^[a-zA-Z0-9 .,-_]{1,20}$", $user)) // kontrola na
povolené znaky
    {
        $sql = "SELECT user_id, user_jmeno, user_role_id, user_jm,
user_pr FROM users WHERE user_jmeno='".$user."' AND
user_heslo='".$pass."' AND user_aktivni=1";
        $vysledek = dotaz($sql);
        if(count($vysledek['USER_ID'])==1)
        {
            $_SESSION['prihlasen']=true;
            $_SESSION['user_id']=$vysledek['USER_ID'][0];
            $_SESSION['user_jmeno']=$vysledek['USER_JMENO'][0];
            $_SESSION['user_jm']=$vysledek['USER_JM'][0];
            $_SESSION['user_pr']=$vysledek['USER_PR'][0];
            $_SESSION['user_role_id']=$vysledek['USER_ROLE_ID'][0];
            header('Location: index.php');
        }
        else
            $zprava ='Neplatné přihlášení';
    }
    else
        $zprava ='Neplatné přihlášení';
}
```

Kódem popsaným níže, uloženým v souboru *index.php*, je zabráněno nepřihlášeným uživatelům v přístupu na tuto stránku. Kód přesměrovává všechny nepřihlášené uživatele zpět na přihlašovací stánku *login.php*.

Skript vracející nepřihlášené uživatele na *login.php*

```
if (isset ($_SESSION['prihlasen']))
{
    if($_SESSION['prihlasen']!=true)
        header('Location: login.php');
}
else
    header('Location: login.php');
```

Soubor *index.php* se nadále stará o správné stránkování souboru a kontrolu přístupových práv. Pro každou roli jsou jednotlivé soubory uloženy ve vlastním adresáři a není tedy možné načítat stránku, která je určena pro jinou uživatelskou roli. Existují ještě sdílené stránky, které jsou pro všechny role přístupné. Ty lze najít přímo v adresáři */page* bez jakéhokoliv dalšího vnoření. Pokud nebude stránka v našem domovském nebo sdíleném adresáři nalezena, načte se chybová stránka *ne.php*.

Načítání jednotlivých stránek do *index.php*

```
if (isset($_GET['id'])) {
    $strana = urlencode($_GET['id']);  //přečte číslo strany z url
adresy
}
else {
    $strana = "home"; // pokud stránka není zadána, bude výchozí
"home"
}
if (ereg("^[a-zA-Z0-9_]*$", $strana)) //kontrola na povolené znaky
```

```
{
    $_SESSION["strana"] = $strana;
    $soubor = "page/".$strana.".php";

    switch($_SESSION['user_role_id']) //jednotlivé přístupné adresáře
podle rolí
    {
        case "1": $cesta="1admin"; break;
        case "2": $cesta="2hajny"; break;
        case "3": $cesta="3sef"; break;
    }
    if (!file_exists($soubor))
    {
        $soubor =  "page/".$cesta."/".$strana.".php";
        if (!file_exists($soubor))
        {
            $soubor = "page/ne.php";
        }
    }
}
    else $soubor = "page/ne.php";
```

6.2 Role administrátor

Administrační sekce umožňuje přidávání nových uživatelů a jejich jednoduchou správu.

Pro přidávání nového uživatele stačí jen zadat jeho jméno, příjmení, uživatelské jméno, heslo a roli, kterou bude uživatel disponovat. Nyní jsou aktuální tři role a každý z nich má přístup pouze do své sekce.

- Admin – umožňuje správu všech uživatelů
- Hajný – plánování a správa jednotlivých porostů
- Šéf – má přístup k celkovému přehledu prací všech hajných v databázi

Dále administrační účet zobrazuje přehled veškerých uživatelů. Jednotliví uživatelé se zde dají jednoduše deaktivovat, čímž se docílí toho, že uživatel s tímto účtem se již nebude moci přihlásit do aplikace, ale jeho účet a veškeré jeho práce zůstanou zachovány. Tento účet se dá zase lehce zaktivnit. Dále administrátor může změnit uživateli heslo, pomocí ikony . Což je třeba v případě, když uživatel zapomene heslo. Jelikož je heslo v databázi zakódováno jednosměrnou hashovací funkcí sha1, nelze již zpětně staré heslo získat. Je tedy potřeba zadat heslo nové. Poslední funkcí je smazání uživatele, přičemž dojde ke kompletnímu vymazání jeho účtu včetně všech uložených záznamů. Je tedy potřeba s touto funkcí nakládat nanejvýš opatrně. Smazání se provádí klikem na ikonu a potvrzením informativní hlášky o smazání uživatele.

Administrátor může upravovat veškeré záznamy až na svůj vlastní, který ale může případně upravit jiný administrátor.

6.3 Role hajný

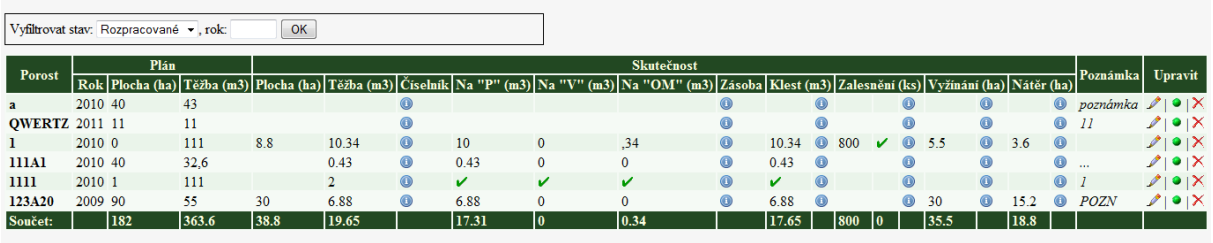
6.3.1 Základní přehled

Základní přehled plní funkci domovské stránky a případně se dá na něj dostat z *Menu -> Domů* či klikem na hlavní logo aplikace. Ukazuje přehled veškerých zadaných porostů uživatele (nebo pasek vyfiltrovaných podle filtru). Je zde přehledně vidět aktuální stav jednotlivých porostů. Ukazuje stav těžby, funguje jako výkaz skladu, dále ukazuje zbývající klest k úklidu, plán a skutečnost zalesnění, přehled zda bylo již vyžínání a natírání ukončeno nebo zda doposud ještě nebylo. Už dokončené činnosti jsou označeny zelenou fajfkou ✔. U naplánovaného, ale ještě nedokončeného vyžínání a nátěru, je značka ✘.

Přes tuto stránku se také dostáváme k jednotlivým pracím pro jednotlivé porosty. Stačí kliknout na ikonu ⓘ, která nás přesměruje na další stránky aplikace.

Dále zde můžeme editovat jednotlivé buňky porostu, pro které má editace význam (jméno porostu, plánovaný rok, plocha a těžba, skutečná plocha porostu, kde se bude těžit a poznámka). Můžeme zde také porost označit za hotový, čímž porost přesuneme mezi porosty hotové a již by se s ním nemělo dále pracovat. I když v aplikaci tomu zabráněno není. Lze též vrátit porost zpět mezi rozpracované pomocí tlačítka v podobě červeného puntíku. Porost můžeme také nenávratně smazat, čímž dojde ke kompletnímu smazání i všech činností prováděných na daném porostu. Proto je třeba s touto funkcí pracovat velice opatrně.

Položky lze filtrovat jednoduchým filtrem, který by jistě bylo vhodné rozšířit, ale pro prozatímní účely je tento filtr dostatečný. Vyfiltrovat lze jen porosty hotové, rozpracované nebo všechny a dále porosty těžené v zadaný rok těžby (prázdné pole rok znamená všechny roky).


Vyfiltrovat stav: Rozpracované , rok: OK

Porost	Plán			Skutečnost												Poznámka	Upravit
	Rok	Plocha (ha)	Těžba (m3)	Plocha (ha)	Těžba (m3)	Číselník	Na "P" (m3)	Na "V" (m3)	Na "OM" (m3)	Zásoba	Klest (m3)	Zalesnění (ks)		Vyžínání (ha)	Nátěr (ha)		
a	2010	40	43													*poznámka*	
QWERTZ	2011	11	11													*11*	
1	2010	0	111	8.8	10.34		10	0	,34		10.34	800	✔	5.5	3.6		
111A1	2010	40	32,6		0.43		0.43	0	0		0.43					...	
1111	2010	1	111		2		✔	✔	✔		✔					*1*	
123A20	2009	90	55	30	6.88		6.88	0	0		6.88			30	15.2	*POZN*	
Součet:		182	363.6	38.8	19.65		17.31	0	0.34		17.65	800	0	35.5	18.8		

Obrázek 7 - Základní přehled prací v porostu

Z pohledu programátora

Filtrování je provedeno jednoduchým formulářem, který pravidla, podle kterých chceme filtrovat, zpracuje a uloží do trvalé proměnné $_SESSION[] a pravidla filtrování tak zůstanou i při opuštění domovské stánky a opětovnému navrácení.

Zpracování formuláře pro filtrování

```
if (isset ($_POST['filtr']))
{
    $_SESSION['home_filtr_rozprac'] = $_POST['filtr_stav'];
    if($_POST['filtr_rok'])
        if (ereg("^[0-9]{4}$",  $_POST['filtr_rok']))
            $_SESSION['home_filtr_rok'] = $_POST['filtr_rok'];
        else  $_SESSION['home_filtr_rok'] = '';
    else  $_SESSION['home_filtr_rok'] = '';
}
```

Následně seskupí SQL dotaz pro výpis uložených porostů, přesně podle zadaných požadavků z filtrování pomocí několika kroků:

SQL dotaz pro výběr porostů podle nastaveného filtru

```
$sql = "SELECT p_id, p_cislo, p_plan_tezba, p_plan_plocha, p_tezba,
p_plocha, p_hotovo, p_rok, p_pozn FROM porost
    WHERE p_user_id = ".$_SESSION['user_id']." AND ";
// výběr stavu
switch ($_SESSION['home_filtr_rozprac']) {
    case 'hot': $sql .= 'p_hotovo=1'; break;
    case 'all': $sql .= '(p_hotovo=1 OR p_hotovo=0)'; break;
    default: $sql .= 'p_hotovo=0';
}
// výběr roku
if ($_SESSION['home_filtr_rok'])
    $sql.=" AND p_rok=".$_SESSION['home_filtr_rok']." ";
// seřazení výsledků
$sql.=" ORDER BY p_cislo";
```

Jeho následným provedením dostaneme veškeré porosty, které chceme zobrazit. Pomocí dalších SQL příkazů následně vybereme z databáze i další data o daném porostu.

6.3.2 Přidat nový projekt

Před samotnou prací na porostech je potřeba nejprve naplánovat nový projekt. Přidáním nového projektu najdeme v *Menu -> Přidat projekt*. Tlačítky *Přidej řádek* / O*deber řádek* nastavíme počet projektů, které hodláme v jednom kroku přidat. Následně vyplníme tabulku: číslo porostu, plánovaná plocha porostu, objem plánované těžby, rok, kdy se bude porost těžit (automaticky je předvyplněn aktuální rok) a krátká poznámka s dodatečnými informacemi a odsouhlasíme tlačítkem *Ulož*. Aplikace zkontroluje vstupní data. Pokud nebudou přijata, upozorní nás na chybu a vypíše, v kterém řádku a sloupci chyba nastala. Pokud data projdou úspěšně kontrolou, informačním boxem informuje o úspěšném uložení. Nyní se můžeme vrátit zpět na základní přehled, kde již najdeme nově zadané projekty.

Z pohledu programátora

Přidávání jednotlivých řádků v tabulce se provádí pomocí JavaScriptu a technologie DOM. Konkrétně se jedná o funkci v JavaScriptu *addRowToTable(),* uloženou v souboru *js/JavaScript.js,* která je volaná při stisku tlačítka *Přidej řádek.*

Funkce přidávající nový řádek do tabulky

```
function addRowToTable()
{
// zjištění id místa kam budeme objekty vkládat – tabulka
'tblPridatProjekt'
  var tbl = document.getElementById('tblPridatProjekt');
// zjištění aktuálního počtu řádků tabuky
  var lastRow = tbl.rows.length;
  var iteration = lastRow;
// přidání nového řádku na pozici podposledním řádkem a nastavení
atributů
  var row = tbl.insertRow(lastRow);
  row.setAttribute("class", lastRow%2==0 ? "sudyRadek aktual" :
"lichyRadek aktual" );

// vložení prvního sloupce s pořadovým číslem sloupce
  var cellLeft = row.insertCell(0);
  var textNode = document.createTextNode(iteration);
  cellLeft.appendChild(textNode);

// vložení HTML prvku input – vstupní textové pole pro číslo porostu
  var cellRight = row.insertCell(1);
  var el = document.createElement('input');
  el.type = 'text';
  el.name = 'cisloProjektu' + iteration;
  el.id = 'cisloProjektu' + iteration;
  el.size = 7;
  el.maxLength = 7;
  cellRight.appendChild(el);


… // postupné vkládání dalších objektů do řádku tabulky

// aktualizace počtu řádků v tabulce
  document.getElementById("radku").value =
parseInt(document.getElementById("radku").value, 10) + 1;
}
```

Výše napsaný kód je inspirovaný tutoriálem z mredkj.com – *DOM Table Add Row*[19].

Pomocí stejné technologie je dosaženo i odstranění vždy posledního řádku v tabulce, pokud se tedy nejedná o řádek číslo 1, u kterého již logicky odstranění není umožněno. Konkrétně jde o funkci *removeRowFromTable()*:

Funkce odstraňující řádek tabulky

```
function removeRowFromTable()
{
  var tbl = document.getElementById(tblPridatProjekt);
  var lastRow = tbl.rows.length;

// pokud je v tabulce více nežli jeden řádek
  if (lastRow > 2)
  {
```

[19] http://www.mredkj.com/tutorials/tableaddrow.html

```
    // smazání řádku
      tbl.deleteRow(lastRow - 1);

    // aktualizace počtu řádků v tabulce
  document.getElementById("radku").value =
parseInt(document.getElementById("radku").value, 10) - 1;
  }
}
```

Při ukládání dat je kontrola správnosti dat řešena pomocí JavaScriptu a regulárních výrazů pro každé textové pole v tabulce funkcí *validateRow(frm)*, kde argumentem *frm* je odesílaný formulář.

Kontrola dat pomocí JavaScriptu

```
function validateRow(frm)
{
    var tbl = document.getElementById('tblPridatProjekt');
    var Row = tbl.rows.length - 1;
    var i;
// zajištění procházení všech řádků tabulky
    for (i=1; i<=Row; i++) {

// kontrola čísla projektu
      var aRow = document.getElementById('cisloProjektu' + i);
      var reg = /^[0-9a-zA-Z]{3,7}$/;
       if (reg.test(aRow.value)==false) {
        alert('Číslo projektu na řádku č.'+i+' není korektně vyplněno
(např.123A10)');
        return false;
      }

// kontrola plochy - test desetinného čísla
      var aRow2 = document.getElementById('planPlocha' + i);
      reg = /^[0-9]{1,4}(\.[0-9][0-9]?)?$/;
      if (reg.test(aRow2.value)==false) {
        alert('Planovaná plocha na řádku č.'+i+' není korektně
vyplněna (např.2.34)');
        return false;
      }

// kontrola těžby - test desetinného čísla
      var aRow3 = document.getElementById('planTezba' + i);
      reg = /^[0-9]{1,3}(\.[0-9][0-9]?)?$/;
        if (reg.test(aRow3.value)==false) {
        alert('Plánovaná těžba na řádku č.'+i+' není korektně vyplněna
(např.2.34)');
        return false;
      }
    } // for
    return true;
}
```

Po úspěšné kontrole zadaných dat je již provedeno samotné uložení dat do databáze, konkrétně do tabulky *porost*. Ukládání se řeší pomocí transakce, kde se postupně ukládají jednotlivé porosty (řádky tabulky) do databáze, ale teprve po vložení posledního záznamu se příkazem *commit* definitivně zviditelní a korektně uloží do databáze. V případě

neúspěchu, který by z důvodu dobré kontroly dat nikdy neměl nastat, se veškeré záznamy odvolají příkazem *rollback*. Data se poté musí zapsat znovu. Viz ukázka zdrojového kódu ukládání dat do databáze:

Uložení nového porostu do databáze

```
if (isset($_POST['uloz']))
{
    $radku = $_POST['radku'];
    for ($i=1; $i<=$radku; $i++)
    {
        if (empty($_POST['cisloProjektu'.$i]))
            break;

        $sql[$i-1] = "INSERT INTO porost (p_id, p_user_id, p_cislo,
p_plan_tezba, p_plan_plocha, p_rok, p_pozn, p_hotovo ) VALUES
sq_porost.nextval, ".$_SESSION['user_id'].",
'".$_POST['cisloProjektu'.$i]."', ".$_POST['planPlocha'.$i].",
".$_POST['planTezba'.$i].", ".$_POST['rok'.$i].",
'".$_POST['poznamka'.$i]."', 0)";
    }
    $i--;
    if (dotaz_t($sql))
        $hlaska='Uloženo '.$i.' řádků.';
    else
         $hlaskaE='Uložení se nezdařilo!';
}
```

Samotná funkce *dotaz_t($sql)* pak řeší přímé uložení dat v poli *$sql[]* do databáze pomocí transakcí a v případě výskytu chyby stornování příkazem rollback.

6.3.3 Zaměstnanci

V sekci *Zaměstnanci*, nacházející se v *Menu -> Zaměstnanci*, je přehled veškerých zaměstnanců, kterým se následně přiřazují jednotlivé lesní práce. Zaměstnanec je zde prezentován jen unikátním jménem, které může představovat jak jméno konkrétního zaměstnance, tak i soukromníka, další firmu vykonávající lesní práce, brigádníky,… Popisovat korektně každého zaměstnance či firmu celou adresou, číslem IČO, číslem účtu a dalšími informacemi nemá zatím v této aplikaci význam. Proto jsem se rozhodl prozatím zjednodušit zadávání jen pomocí jednoho vstupního pole se jménem.

Tato sekce umožňuje přidávání nových zaměstnanců zadáním jejich jména. Dále v přehledu můžeme u jednotlivých zaměstnanců editovat jejich jméno, nebo jej můžeme rovnou smazat. Díky použitému návrhu databáze nelze odstranit zaměstnance, který již někdy pracoval a jeho záznam práce je stále v databázi. Proto zde bylo použito podobné technologie aktivování/deaktivování zaměstnance jakou je možno vidět v administrační sekci u přehledu uživatelů. Je zde tedy možno zaměstnance deaktivovat tlačítkem  označujícím, že je uživatel stále aktivní, čímž již nebude nabízen v seznamu zaměstnanců u jednotlivých prací a zbytečně tak zabírat místo v seznamu. Nebo deaktivované zaměstnance zpět aktivovat tlačítkem  .

Z pohledu programátora

Text zadaný do vstupního pole, které se následně ukládá do databáze, je kontrolován regulérním výrazem na nepovolené znaky. Kontrola nám zabrání případnému nepovolenému průniku do databáze zvaném SQL injection. Jméno nového zaměstnance je kontrolováno konkrétně regulérním výrazem, přes který neprojdou znaky: ^ " ' \ / ; a text kratší nežli jeden znak a delší nežli 20 znaků. Pokud data projdou kontrolou, následně se formulář odešle ke zpracování na server.

Kontrolní regulérní výraz

```
reg = /^[^"'\/;]{1,20}$/;
```

Problém aktivování a deaktivování zaměstnanců je v databázi řešeno v tabulce *zaměstnanci* ve sloupci *zam_aktualni* indexy *1* – pro aktivního, *0* – pro neaktivního a následně ověřován kontrolováním této hodnoty.

6.3.4 Přehled dřevin

Přehled dřevin lze nalézt v *Menu -> Přehled dřevin*. Zobrazuje jednotlivé kódy dřevin a jejich jméno ve zkratce. Tyto dřeviny je následně možné zadávat do aplikace. Editace ani jejich úprava není umožněna, neboť by neměla velký význam. Jednotlivé dřeviny jsou napojené na kubírovací tabulky, které by se musely případně upravovat spolu s dřevinami. Navíc se počítá s tím, že aplikace se naplní těmito potřebnými daty před spuštěním a dále by již nemělo být nutné další typy stromů zadávat. Případnou změnu musí provádět administrátor ručně přímo v databázi.

6.3.5 Těžba dřeva - Číselník

Kliknutím na ikonu v sloupci *Číselník* v základním přehledu se dostáváme na přehled jednotlivých číselníků daného porostu seřazených podle roku a měsíce. Pro každý číselník můžeme přidávat nové sortimenty ke kubírování a zobrazit sumář vytěženého dřeva.

K přidání nového číselníku stačí zadat rok a měsíc těžby, kde automaticky je již předvyplněn aktuální údaj a vybrat dřevaře z roletkové nabídky všech zaměstnanců.

Po ukončení těžby v porostu zaškrtneme v sekci *těžba ukončena* přepínač *Ano* a tím se v základním přehledu porostů objeví v sloupci *Těžba (m^3)* ikona ✔, která dává na vědomí, že je již porost kompletně vytěžený. U takto označených porostů se v číselníku neustále objevuje oznamující hláška: *Těžba byla již ukončena!* a již nejde vytvořit další číselník.

Pomocí číselníků se také dostaneme k sumářům buďto pro jednotlivé číselníky, nebo celkový sumář za celý porost.

6.3.6 Kubírování kulatiny

Na kubírování kulatiny se dostaneme přes základní přehled -> *Číselník -> kulatina.* Do boxu *Nové kusy* se zapisují jednotlivé kusy kulatiny. Vyplněním čísla kusu, oddenku

(pokud se sem napíše jakýkoliv znak, bude bráno, že tento kus je oddenek), číslo dřeviny, délka a průměr. Počet řádků lze nastavit pomocí tlačítek *Přidej/Odeber řádek*. Číslo kusu se vyplňuje jen u prvního řádku, další čísla se vyplňují automaticky zvýšením hodnoty předchozího čísla o jedničku. Po vyplnění všech hodnot u všech řádků tlačítkem *Ulož* se odešlou data k uložení. Nejprve dojde k automatické kontrole zadaných údajů, následné uložení hodnot do databáze a u každého kusu dojde k vypočtení jeho objemu. Pokud se objem automaticky nevypočítá, bude u objemu *0*. To znamená, že pro danou dřevinu a rozměry není záznam v databázi u kubírovacích tabulek. V tomto případně je potřeba buďto záznam doplnit do kubírovacích tabulek, nebo záznam opravit.

V boxu *Kusy* vidíme přehled všech kusů (sortimentů), které jsou v daném číselníku uloženy. Jednotlivé záznamy se dají jednotlivě odstranit. Případně tlačítkem *Změnit hodnoty* se jednotlivé hodnoty zobrazí jako editovatelné a můžeme hodnoty změnit a následně uložit. Červeným křížkem můžeme také jednotlivé záznamy z databáze odstranit.

Z pohledu programátora

Přidávání a odebírání posledního řádku je zajištěno pomocí technologie DOM + JavaScript. Hlavní rozdíl je zde v automatickém vkládání čísla kusu, které se počítá přečtením čísla o řádek výš a zvětšením tohoto čísla o 1. Viz ukázka kódu

Část kódu zvyšující číslo předešlé buňky o jedna

```
  var cell1 = row.insertCell(1);
  var el = document.createElement('input');
  el.type = 'text';
  el.name = 'cislo' + iteration;
  el.id = 'cislo' + iteration;
// přečtení předchozího čísla a jeho zvětšení o 1
  el.value = parseInt(document.getElementById("cislo"+(iteration-
1)).value, 10) + 1;
  el.size = 4;
  el.maxLength = 4;
  cell1.appendChild(el);
```

Před uložením nových kusů dochází k automatické kontrole vstupních údajů pomocí funkce *validateRowKubKulatinyZmena(frm, dreviny),* kde argument *frm* je vlastní formulář a argument *dreviny* je řetězec s ID čísly dřevin, které mohu do databáze uložit a pro které se nachází v databázi záznam, protože JavaScript nemá přístup do databáze a nemůže tak zjistit jaké dřeviny má kontrolovat. Argument *dreviny* je vkládán pomocí PHP funkce *stringDrevin()* ze souboru i*/my_fce.php*.

Před přidáním řádku do databáze se vykoná trigger *tr_cis_kulatina* pro příkaz insert, který zajistí kubírování kusu pomocí volané funkce *fce_kubirovani(id_dreviny, delka, prumer),* která zajistí správné kubírování a vrátí objem zadaného kusu. V případě nenalezení shody vrátí *0*. Dále trigger *tr_cis_kulatina* zajistí aktualizaci objemu v tabulce *ciselnik* a pomocí dalšího triggeru následně i v tabulce *porost*. Toto řešení není nejlepší, protože může nastat chyba v případě nějaké výjimky nebo porušení jednotlivých funkcí

a tato chyba již nejde jednoduše opravit. Ale protože přidávání nových kusů do číselníku je práce výrazně méně frekventovaná nežli zobrazování hlavní tabulky s porosty, přistoupil jsem na toto řešení, které ušetří nemalý databázový čas, protože nemusím neustále zpracovávat ohromné množství dat. Navíc je tento objem používán dále i v jiných místech aplikace. Podobné triggery musejí samozřejmě existovat i pro příkazy delete a update nad tabulkou *cis_kulatina*, aby byla veškerá data vždy konzistentní.

Insert trigger nad tabulkou *cis_kulatina*

```
CREATE OR REPLACE
TRIGGER tr_cis_kulatina
  BEFORE INSERT ON cis_kulatina
  FOR EACH ROW
DECLARE
  oldObjem NUMBER;
BEGIN
-- zjištění objemu nového kusu
  :NEW.ck_objem := FCE_KUBIROVANI(:NEW.CK_DR_ID, :NEW.CK_DELKA,
:NEW.CK_PRUMER);

-- zjištění stávajícího objemu v tabulce cislenik
  SELECT c_tezba INTO oldObjem  FROM ciselnik WHERE c_id =
:NEW.CK_C_ID;

-- vypočítání nového objemu
  IF (oldObjem is NULL) THEN
    oldobjem := 0;
  END IF;
  oldObjem := oldObjem + :NEW.ck_objem;

-- aktualizace nového objmeu v tabulce ciselnik
  UPDATE ciselnik SET C_TEZBA=oldObjem WHERE c_id = :NEW.CK_C_ID;
END;
```

Funkce pro kubírování kulatiny

```
CREATE OR REPLACE
FUNCTION fce_kubirovani (aDR_ID IN NUMBER, aKT_DELKA IN NUMBER,
aKT_PRUMER IN NUMBER )
RETURN NUMBER
AS
    objem NUMBER;
BEGIN
  SELECT kt_objem INTO objem FROM kubirovaci_tabulky WHERE DR_ID =
(SELECT dr_kub FROM dreviny WHERE dr_id=aDR_ID) AND KT_DELKA =
aKT_DELKA AND KT_PRUMER = aKT_PRUMER;

  RETURN objem;

  EXCEPTION
-- když řádek nenalezen
    WHEN NO_DATA_FOUND THEN RETURN 0;
END;
```

6.3.7 Kubírování surových kmenů

Na kubírování surových kmenů se dostaneme přes základní přehled -> *Číselník* -> *surové kmeny*. Do boxu *Nové kusy* se zapisují nové surové kmeny. Pro každou dřevinu a třídu (0 - 5), kterou potřebujeme, vyplníme počet oddenků a celkový počet kusů. Opět pomocí tlačítek *Přidej/Odeber řádek* můžeme nastavit počet řádků, které potřebujeme. Po vyplnění všech hodnot tlačítkem *Ulož* dojde k automatické kontrole zadaných údajů, následně se vypočte objem a dojde k uložení dat do databáze. Pro jeden číselník lze uložit vždy jen jednu kombinaci dřeviny a třídy. Nemůžou se tedy v záznamu vyskytovat dva řádky se stejnou dřevinou a třídou.

Uložená data se nacházejí v boxu kusy. Pokud dojde při zadávání k překlepu, nebo je potřeba záznamy upravit, lze úpravu jednoduše udělat pomocí tlačítka změnit hodnoty, který nám zaktivní hodnoty pro editaci a po úpravě data znovu uložíme. Případně můžeme celý řádek vymazat v daném číselníku. Jednotlivé záznamy se též dají odstranit tlačítkem pro smazání.

Z pohledu programátora

Programátorské řešení je hodně podobné jako u kubírování kulatin.

Při vkládání záznamu do databáze je na každý řádek použit trriger *cis_surovaky*, který pomocí funkce na kubírování surových kmenů *fce_kubirovani_sur(trida, pocet)*, zajišťuje správné spočítání objemu. Další úlohou trrigeru *cis_surovaky* je změna příslušného objemu v tabulce *ciselnik* a tedy následně i v tabulce *porost*. Obdobný trigger existuje samozřejmě i pro příkazy k editaci a mazání záznamu.

Funkce pro kubírování surových kmenů

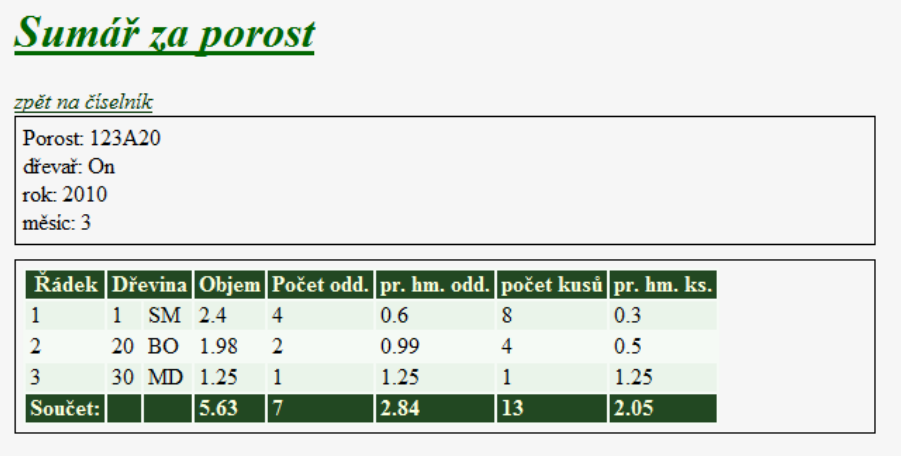
```
CREATE OR REPLACE
FUNCTION fce_kubirovani_sur (trida IN NUMBER, pocet IN NUMBER)
RETURN NUMBER
AS
    objem NUMBER;
BEGIN
  CASE
    WHEN trida = 0 THEN objem := 0.05*pocet;
    WHEN trida = 1 THEN objem := 0.1*pocet;
    WHEN trida = 2 THEN objem := 0.2*pocet;
    WHEN trida = 3 THEN objem := 0.3*pocet;
    WHEN trida = 4 THEN objem := 0.4*pocet;
    WHEN trida = 5 THEN objem := 0.5*pocet;
    ELSE objem := 0;
  END CASE;

  RETURN objem ;
END;
```

6.3.8 Sumář k číselníku

Na sumář se dostaneme ze základního přehledu -> *Číselník* -> *Zobraz sumář*. Sumář k číselníku je přehled vytěženého dřeva jedním dřevařem za daný měsíc zesumírovaný podle dřevin. Sumář udává přehled jednotlivých vytěžených dřevin a jejich

celkový objem, počet oddenků a jejich průměrnou hmotnatost, celkový počet kusů a jejich průměrnou hmotnatost.



Sumář za porost

zpět na číselník

Porost: 123A20
dřevař: On
rok: 2010
měsíc: 3

Řádek	Dřevina	Objem	Počet odd.	pr. hm. odd.	počet kusů	pr. hm. ks.
1	1 SM	2.4	4	0.6	8	0.3
2	20 BO	1.98	2	0.99	4	0.5
3	30 MD	1.25	1	1.25	1	1.25
Součet:		5.63	7	2.84	13	2.05

Obrázek 8 - Sumář k číselníku

6.3.9 Sumář za celý porost

Na sumář se dostaneme ze základního přehledu -> *Číselník* -> *Sumář za celý porost*. Sumář za celý porost je přehled vytěženého dřeva všemi dřevaři za celé období zesumírovaný podle dřevin. Sumář udává přehled jednotlivých vytěžených dřevin a jejich celkový objem, počet oddenků a jejich průměrnou hmotnatost, celkový počet kusů a jejich průměrnou hmotnatost.

Z pohledu programátora

Z programátorského hlediska zde stojí za zmínku databázový dotaz, kterým spojuji dvě nestejnorodé tabulky *cis_kulatina* a *cis_surovaky* a následně na výsledné tabulce provádím jejich součty podle jednotlivých dřevin.

Select vracející tabulku sumáře

```
SELECT sum(objem) AS objem, sum(pocet) AS kusu, sum(odd) AS odd,
sum(objem)/sum(objem) AS prumer, dr, dr_nazev
FROM (
  SELECT ck_objem AS objem, ck_oddenek AS odd, 1 AS pocet, ck_dr_id AS
dr FROM cis_kulatina WHERE ck_c_id=
    ANY(SELECT c_id FROM ciselnik LEFT JOIN porost ON c_p_id=p_id
WHERE p_id=$porostID)
  UNION
  SELECT cs_objem AS objem, cs_odd AS odd, cs_pocet AS pocet, cs_dr_id
AS dr FROM cis_surovaky WHERE cs_c_id=
    ANY(SELECT c_id FROM ciselnik LEFT JOIN porost ON c_p_id=p_id
WHERE p_id=$porostID)
)
LEFT JOIN dreviny ON dr=dr_id GROUP BY dr, dr_nazev ORDER BY dr
```

6.3.10 Výkaz skladu

Výkaz skladu se nachází v základním přehledu, v sloupci *Zásoba* a rozkliknutím ikony u správného porostu. Dělí se na tři části. *Zásoba (m^3)* udává objem zásoby dřeva

v lese. Ukazuje, kolik zásoby zůstává na pařezu „na P“ – toto číslo je zde automaticky přiřazeno z objemu těžby. Kolik zásoby je po přiblížení koňmi „na V“ a na odvozním místě „na OM“. Po odvezení veškerého dřeva zůstane tedy všude 0. Tento rychlý přehled se nachází i na základním přehledu.

Další sekce je *Přidat nový úkon*, kde zadáváme jednotlivé odvozy dřeva z předvolené nabídky: P-V, P-OM, V-OM a konečný odvoz dříví z lesa. K tomu se také pojí poslední sekce, kde je přehled veškerých vykonaných etap odvozu. V případě chyby se dá záznam editovat nebo případně vymazat z aplikace.

6.3.11 Klest

Na stránku *Klest* se dostaneme ze základního přehledu, výběrem správné ikony ve sloupci *Klest (m3)*. Sekce klest nám dává přehled o již spáleném či jinak uklizeném klestu po těžbě.

Můžeme jednoduše přidávat jednotlivé práce s úklidem klestu. Ve *vykonaných pracích* se nám ukazují jednotlivé uskutečněné práce. V *přehledu* vidíme aktuální stav paseky, tedy jak je paseka veliká, kolik bylo již klestu uklizeno a kolik ještě zbývá uklidit.

6.3.12 Zalesnění

V první části stránky zalesnění je přehledné shrnutí, které mapuje veškeré plány zalesnění, počet již zasázených stromků a počet stromků, které je třeba ještě zasadit pro danou paseku (*Rozdíl*). Taktéž se zde eviduje i plocha, která je pouze orientační a nemá prozatím dalšího uplatnění.

Ve druhé části přidáváme nové plány zalesnění paseky. Pro každou dřevinu naplánujme počet sazenic a velikost plochy, na kterou se mají vysázet. Každá dřevina může být plánována pouze jedenkrát, stejnou dřevinu se tedy podruhé přidat nepodaří. Pokud se plán změnil, může se plán s plochou a počtem sazenic editovat nebo úplně zrušit.

V poslední sekci zadáváme již vykonané práce – vysázené stromky. Eviduje se zde datum, zaměstnanec, který stromky sázel, dřevina, plocha a počet vysázených stromků.

6.3.13 Přehled zalesnění

K zalesnění náleží i sekce *Přehled zalesnění* nacházející se v horním menu. Zde je přehled veškerých stromků, které byly vysázeny a stromků, které je třeba ještě vysázet pro všechny aktuálně rozpracované paseky. Přehled je roztříděný podle dřevin.

Z pohledu programátora

Z programátorského hlediska bylo jednoznačně nejobtížnější vytvořit správný SQL dotaz, který by vrátil požadovaná data, jak spojit tabulky, správně je seskupit, následně z nich vypočítat správná statistická data a jak zpracovat záznamy vracející *NULL*.

Nakonec se vše podařilo pomocí trošku nepřehledného pohledu *prehledStromku* složeného ze tří selectů. Tento select patřil k nejnáročnějším selectům v celé aplikaci.

Databázový pohled vracející tabulku *prehledStromku*

```
CREATE OR REPLACE VIEW prehledStromku AS (
SELECT dr_id, dr_nazev, p_plocha, p_pocet, h_plocha, h_pocet,
NVL(p_plocha,0)-NVL(h_plocha,0) AS r_plocha, NVL(p_pocet,0)-
NVL(h_pocet,0) AS r_pocet
FROM (SELECT zalp_dr_id, sum(zalp_plocha) AS p_plocha, sum(zalp_pocet)
AS p_pocet FROM zalesneni_plan LEFT JOIN porost ON p_id=zalp_p_id
WHERE p_hotovo=0 GROUP BY zalp_dr_id) p
FULL JOIN (SELECT zalh_dr_id, sum(zalh_plocha) AS h_plocha,
sum(zalh_pocet) AS h_pocet FROM zalesneni_hot LEFT JOIN porost ON
zalh_p_id=p_id WHERE p_hotovo=0 GROUP BY zalh_dr_id) h
ON zalp_dr_id = zalh_dr_id LEFT JOIN dreviny ON dr_id=NVL(zalp_dr_id,
zalh_dr_id)
) WITH READ ONLY;
```

6.3.14 Vyžínání

Tato stránka ukazuje uskutečněné práce vyžínání daného porostu proti buřeni.

Nejprve je potřeba naplánovat, v kterých rocích se bude vyžínat. Tyto roky následně v sekci *Přidat rok vyžínání* zadat do aplikace.

Následně se přidávají uskutečněné práce. Pomocí jednoduchého formuláře se eviduje datum práce (kde rok se vybere z roletkového menu), zaměstnanec a vyžnutá plocha. Dále je zde celkový přehled, kolik zbývá vyžnout za jednotlivé roky. Po ukončení vyžínání v daném roce je potřeba to potvrdit tlačítkem *Hotovo*. Jako poslední sekce je přehled vykonaných prací.

6.3.15 Nátěr

Předposlední z hlavních činností aplikace je sledování nátěru stromků proti okusu. Tato stránka je funkčně shodná se stránkou *Vyžínání*. Opět je rozdělena na čtyři sekce:

- *Přidat rok nátěru* – naplánuje se natírání
- *Přidat nový úkon* - přidá zadanou práci do databáze
- *Přehled (ha)* – přehled o zbývající velikosti plochy, která se má ještě natřít pro jednotlivé roky
- *Vykonané práce* – přehled všech uskutečněných nátěrů na daném porostu

6.3.16 Prořezávky

Zde se plánují jednotlivé prořezávky, které budou probíhat. Můžou se naplánovat najednou nebo postupně v roce, kdy se bude prořezávka provádět. Po naplánování přidáváme jednotlivé hotové práce. Dále je zde přehled, který ukazuje, kolik zbývá práce ještě dodělat. Po dokončení prořezávek kliknutím na tlačítko *Hotovo* se prořezávky v daném roce ukončí.

Jako poslední část je přehled veškerých hotových prací s možností práci smazat, pokud se zadá špatně, nebo nastane jiný důvod k smazání.

6.4 Role šéf

Tato část slouží nadřízenému. Ten tak získává přehled o veškerých porostech, na kterých jednotliví hajní pracují nebo pracovali.

Přehled je shodný s přehledem jaký vidí každý hajný, jen zde přibyl sloupec *Zaměstnanec*, kde je jméno hajného, který má daný porost na starosti. Je tu také zrušena možnost jakékoliv editace a přístup na podrobnosti kromě číselníků. Ostatní jsou pro nadřízeného nepodstatné. Případně není problém je popsat. U číselníků je možné zobrazit jen základní přehled a veškeré sumáře.

Porosty se dají třídit podle rozpracovanosti, podle jednotlivých hajných nebo vypsat jen porosty naplánované na zadaný rok. V této části by se jistě hodilo rozšířené a komfortnější filtrování jednotlivých porostů. Ale pro základní přehled je připravené filtrování dostatečné.

7 Možnosti dalšího rozvoje aplikace

Vzhledem k tomu, že celá aplikace momentálně nemá určené komerční využití, ale spíše ukazuje, jakým směrem by se měl ubírat budoucí program, který by následně na trhu mohl uspět. Aplikaci by bylo vhodné rozšířit o další funkce:

- Rozšíření aplikace o možnosti exportu dat, tisk sestav a sumářů.
- Upravit sekci zaměstnanců, doplnit jejich údaje a přidělit jednotlivým zaměstnancům konkrétní možné práce.
- Vytvořit tabulku s evidencí všech prací. Vytvořit tak přehled všech uskutečněných prací v lese.
- Doplnit finanční záležitosti. Zohlednit jak ceny, které se platí lidem za práci, ale i ceny, které za jednotlivé práce platí LČR, ceny za dříví koupené od LČR a příjmy za odvezené dříví k odběratelům. Tím by byl vytvořen zajímavý přehled o hospodaření jednotlivých lesnických úseků.
- Zdokonalit, zmodernizovat design a zpřehlednit jednotlivé části aplikace.
- Rozšířit a zdokonalit možnosti filtrování základního přehledu.
- Důkladně zabezpečit veškeré vstupy a tím zamezit případnému průniku do databáze.

8 Závěr

Cílem bakalářské práce bylo vytvoření aplikace pro hajného, který by měl veškeré své administrativní činnosti uloženy v jednom programu. Práce by navazovali na sebe tak, jak se skutečně vykonávají. Všechny práce jsou zde provázány do jednoho celku a výstupy jednotlivých činností jsou zároveň vstupy do navazujících činností. Aplikace by měla hajnému zrychlit a zefektivnit práci.

Vytvoření aplikace se podařilo, ale její opravdový přínos je potřeba napřed řádně otestovat ve skutečném pracovním nasazení. Dále je třeba upozornit, že i když je aplikace plně funkční a použitelná, pro komerční nasazení do firemní sféry by bylo potřeba poupravit a doplnit o další funkce, čímž by se její hodnota výrazně navýšila.

Při vytváření aplikace jsem vycházel ze znalostí získaných nejen během studia, ale i ze znalostí získaných z odborných článků, které mé obzory v programování webových aplikacích značně rozšířily.

Literatura

ZAPLETAL, Lukáš. *Root.cz* [online]. 28. 1. 2003 [cit. 2010-05-06]. Regulární výrazy v příkladech. Dostupné z WWW: <http://www.root.cz/clanky/regularni-vyrazy-v-prikladech>.

JENCI, Keith. *Mredkj.com* [online]. 2005, 2006-02-21 [cit. 2010-05-06]. Tutorials - DOM Table Add Row. Dostupné z WWW: <http://www.mredkj.com/tutorials/tableaddrow.html>.

ZAJÍC, Petr. *Linuxsoft.cz* [seriál online]. 27. 5. 2004- [cit. 2010-05-06]. Seriál o PHP. Dostupné z WWW: <http://www.linuxsoft.cz/article_list.php?id_kategory=181>.

GRIMMICH, Šimon. *Tvorba-webu.cz* [online]. c2003-2008 [cit. 2010-05-06]. Dostupné z WWW: <http://www.tvorba-webu.cz>.

The PHP Group. *PHP: Hypertext Preprocessor* [online]. c2001-2010 [cit. 2010-05-06]. Dostupné z WWW: <http://cz.php.net>.

JANOVSKÝ, Dušan. *Jak psát web: o tvorbě, údržbě a zlepšování internetových stránek* [online]. 5. 1998, poslední aktualizace 02. května 2010 [cit. 2010-05-06]. Dostupné z WWW: <http://www.jakpsatweb.cz>.

KOSEK, Jiří. *VŠE O WWW* [online]. 1999 [cit. 2010-05-06]. Jak pracují databáze na Webu -- Co je to databáze. Dostupné z WWW: <http://www.kosek.cz/clanky/iweb/12.html>.

SKŘIVÁNEK, František. *Databázový svět: informační portál ze světa databázových technologií* [seriál online]. 01. 03. 2004- [cit. 2010-05-06]. 365 x SQL = praxe. Dostupné z WWW: <http://www.dbsvet.cz/view.php?cisloclanku=2004030102>.

Www.manualy.net: tvorba webu, tutoriály, PHP, MySQL [online]. 2. 8. 2007 [cit. 2010-05-06]. Teorie relačních databází: Normalizace. Dostupné z WWW: <http://www.manualy.net/article.php?articleID=13>.