Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Nové uživatelské rozhraní a navigace pro DCIx

Plzeň, 2013 Tomáš Časta

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne ....

Tomáš Časta

Poděkování

Děkuji doc. Heroutovi za připomínky k textu práce a všem ze společnosti Aimtec, kteří mi pomáhali s mojí diplomovou prací. Dále bych rád poděkoval rodině za finanční i morální podporu ve studiu, zejména pak mamince.

Abstract

The new user interface and navigation for DCIx
This thesis will familiarizes you with the user interface and navigation in DCIx. DCIx is a web application for managing inventory. Design and using of the application does not meet therequirements of today's users. On the ground of user requirements has been proposed a new more modern and interactive user interface and navigation. There are also introduced technologies which were used in the new version of application. Implementation of the selected solution is described in detail, together with solution of problems which have shown up during implementation. One of the requirements for the new application was ensuring testability. Nowadays the application contains a number of automatic tests and the resulting solution was chosen due to the fact that the tests could be used although with some modifications.

Nové uživatelské rozhraní a navigace pro DCIx
Tato diplomová práce vás seznámí s uživatelským rozhraním a navigací v aplikaci DCIx. DCIx je webová aplikace pro správu skladových zásob. Vzhled a ovládání aplikace již nevyhovuje dnešním požadkům uživatelů. Na základě uživatelských požadavků bylo navrženo nové interaktivní uživatelské rozhraní a navigace. Jsou zde také představeny technologie, které byly použity v nové verzi aplikace. Implemenetace zvoleného řešení je detailně popsána, společně s řešením problémů, které se během vývoje objevily. Jedním z požadavků bylo zajištění testovatelnosti. Současná aplikace obsahuje množství automatických testů a výsledné technické řešení bylo zvoleno tak, aby vaplikace byla testovatelná automatickými testy, byť s určitými změnami.

Obsah

1 Úvod

Firma Aimtec dlouhodobě vyvíjí webovou aplikaci *DCIx*, která slouží ke správě skladových zásob. S rozvojem technologií postupně přestával vyhovovat původní vzhled aplikace a bylo rozhodnuto o jeho zásadní změně. Ta samozřejmě zahrnuje i průběžně vzniklé požadavky na další funkcionalitu. Uživatelé dnes požadují interaktivní aplikaci moderního vzhledu, kterou budou moci snadno ovládat. Uživatelé jsou značně ovlivněni používáním aplikací, jako je Windows. Podobné ovládání je tak pro ně více intuitivní a snadněji se jim taková aplikace používá. Používají často možnosti, jako je práce s více otevřenými okny najednou a možná manipulace s nimi. Tyto možnosti současná verze aplikace *DCIx* neumožňuje.

Bylo tedy potřeba navrhnout takové uživatelské rozhraní, které bude splňovat požadavky uživatelů, usnadní jim práci v aplikaci *DCIx* a dá aplikaci modernější vzhled. Protože nebyl jednotný názor na vzhled nového rozhraní, bylo požadováno připravení několik návrhů *GUI*. Z těchto návrhů byl následně vybrán ten nejvhodnější. Současně bylo nutné provést průzkum moderních technologií používaných pro tento druh aplikací. Při průzkumu bylo ovšem nutné přihlédnout k tomu, jaké technologie již aplikace využívá. Následovala volbou nejvhodnějších technologií pro implementaci zvoleného řešení.

Další požadavek na práci byl, aby nové rozhraní bylo testovatelné automatickými testy. Aplikace obsahuje velké množství již hotových automatických testů, které testují správnou funkčnost aplikace. Je nutné, aby bylo zvolené takové technické řešení, které bude automatickými testy ověřitelné.

Cílem práce je tedy provést upgrade *GUI* existující, dlouhodobě používané aplikace *DCIx,* dle požadavků uživatelů a se zachováním testovatelnosti aplikace.

2 DCIx

2.1 Představení aplikace DCIx

Informační systém *DCIx* je řešení pro výrobní, logistické nebo distribuční společnosti hledající cesty k jednodušším, rychlejším a bezchybným činnostem v logistice. Je unikátní nejenom pro interní logistiku, ale pro celý dodavatelský řetězec od dodavatelů až k zákazníkům. Řeší to, co standardní celopodnikové informační systémy (*ERP*[1]) neumí nebo uživatelé nedokážou používat. Není rozhodující, jaké funkce logistický informační systém nabízí, ale zda je lze v praxi jednoduše využít. V této diplomové práci byla dále rozvíjena aplikace *DCIx Portal*. Informace o *DCIx* byly převzaty z [Aimtec].

2.2 DCIx Portal

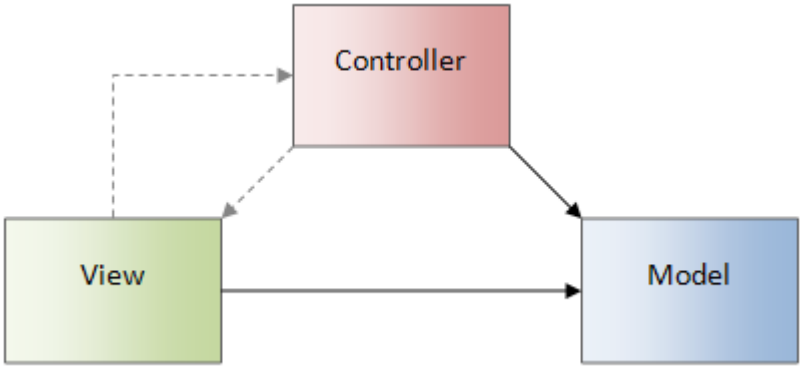
DCIx Portal je komunikační webový portál pro efektivní vazbu mezi odběrateli a dodavateli, je součástí integrovaného systému pro řízení logistiky *DCIx*. Poskytuje nástroje pro zapojení dodavatelů do štíhlého zásobování výroby s minimálním nebo nulovým skladovými zásobami materiálu. Informace o *DCIx portal* byly převzaty z [Aimtec].

2.3 Použité technologie

Aplikace je napsána v Javě, využívá framework *Struts* (viz 4.2.2). Kontejnerem pro běh *servletů*[2] je server *Apache Tomcat*. Jako databáze je použit *MS SQL Server*, pro komunikaci s ním se využívá framework *Alfa* (viz 4.2.4). Stránky jsou tvořeny pomocí *HTML* (viz 4.2.7) a *CSS* (viz 4.2.8) stylů, interaktivitu aplikace zajišťuje *Javascript*, konkrétně framework *JQuery*. Podrobný popis technologií je v kapitole 4.2.

2.4 Architektura aplikace

Aplikace používá architekturu *Model-view-controller (MVC)*. Architektura *MVC* dělí aplikaci na tři logické části tak, aby je šlo upravovat samostatně a dopad změn v jednotlivých částech byl na ostatní části co nejmenší. Tyto tři části jsou *Model*, *View* a *Controller*. *Model* reprezentuje data a *business logiku*[3] aplikace, *View* zobrazuje uživatelské rozhraní a *Controller* má na starosti tok a řízení událostí v aplikaci a obecně aplikační logiku. Informace o *MVC* byly čerpány z [MVC].

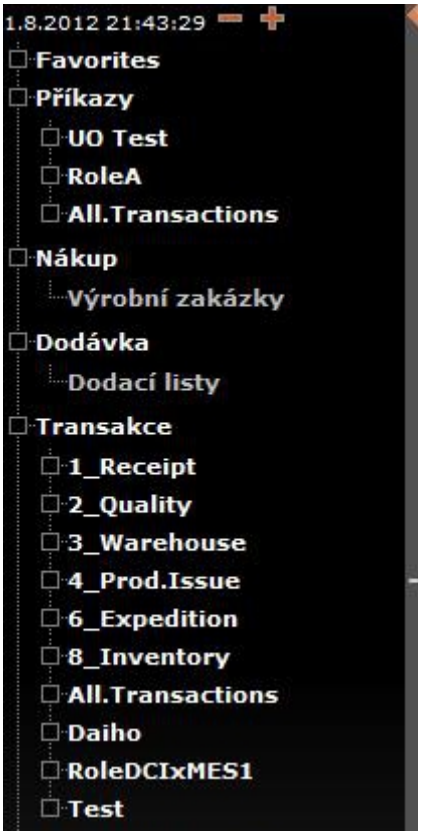


Obrázek 2.1: Architektura MVC (Převzato z [MVC])

1 *ERP* – Enterprise Resource Planning – podnikový informační systém
2 *Servlet* – Java třída, která umí obsloužit *HTTP* požadavek
3 *Business logika* – stanovuje množinu dostupných operací z pohledu klienta

2.5 Současné uživatelské rozhraní

Současné uživatelské rozhraní se skládá z levého menu, které je značně rozsáhlé a podle potřeb se zvětšuje, nebo zmenšuje. Je to základní ovládací prvek celé aplikace. Obrazovky s daty (formuláře), se zobrazují po kliknutí na vybranou položku menu. Vzhledem k tomu, že je položek v menu velké množství, není orientace v něm vždy snadná. Jako vylepšení rozsáhlosti menu, je v něm zahrnuta možnost, dát položku z menu do oblíbených, tyto položky jsou pak zobrazeny zvlášť. Menu dále obsahuje možnost pro zobrazení, nebo skrytí všech položek menu.

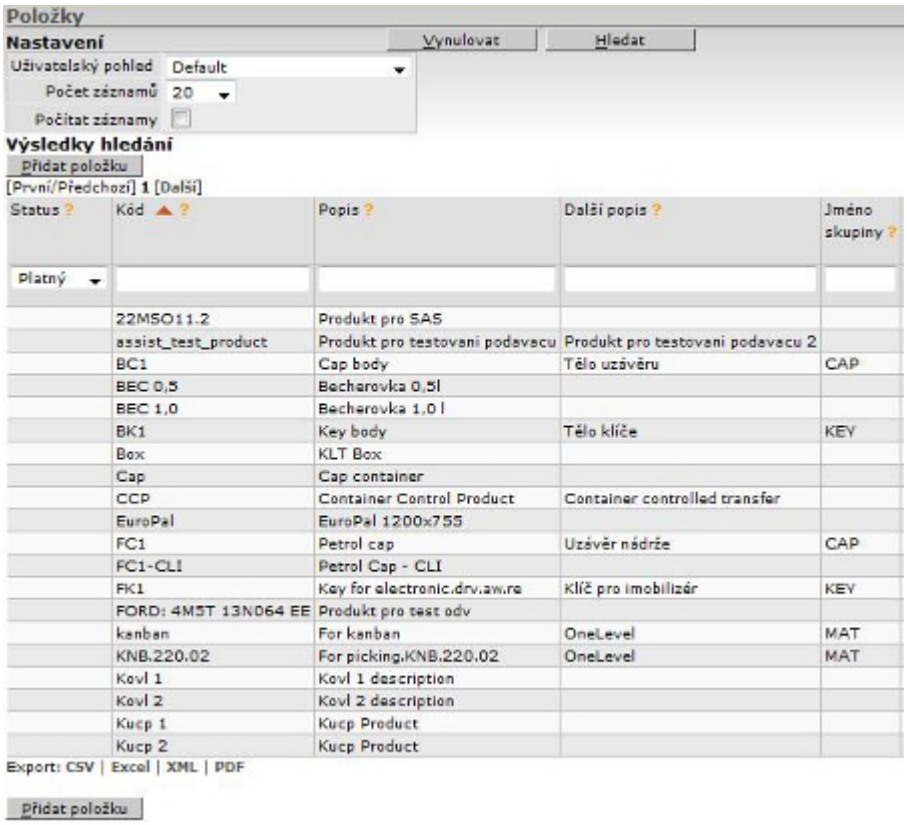


Obrázek 2.2: Menu aplikace

Horní menu nabízí uživateli informace o jeho účtu a informace o prostředí, do kterého je přihlášen. Následují odkazy na nápovědu, zákaznickou podporu, informace o aplikaci a možnost odhlášení se. Další ovládací prvky jsou čtyři ikony, symbolizující události pro práci s emaily, úkoly a zprávami, ty jsou prezentovány obrázky. Posledním ovládacím prvkem v horním menu je pole pro vyhledávání. I z něj se dá otevřít obrazovka s daty, podobně jako je tomu u levého menu.

Obrázek 2.3: Horní menu aplikace

Zbývající část aplikace tvoří místo, kam se načítají formuláře a obrazovky s daty. Je to stěžejní místo celé aplikace. V jednu chvíli může mít uživatel zobrazenou pouze jednu obrazovku s daty, pokud chce zobrazit novou, tak mu nová obrazovka přepíše předchozí. Na tomto místě může uživatel filtrovat, přidávat, upravovat, nebo mazat data zobrazená v tabulce formulářovými prvky.



Obrázek 2.4: Obrazovka s daty aplikace

2.6 Problémy současné uživatelského rozhraní

- rozsáhlé levé menu,
- problém rozsáhlosti menu řeší položka oblíbené v menu, ovšem menu je pak ještě zvětší,
- jediná možnost ovládání aplikace je přes levé menu, nelze ho tedy například skrýt pro větší plochu pro práci s daty
- zobrazení pouze jednoho formuláře,
- nutno mít otevřeno více záložek v prohlížeči, pro současné zobrazení různých dat,
- neinteraktivní aplikace,
- nutné načítání celé stránky při zobrazení formuláře,
- starý design.

3 Nové uživatelské rozhraní

Cíl práce bude řešen několika koncepčními návrhy, ze kterých pak šéfové společnosti Aimtec vybrali ten nejvhodnější.

3.1 Požadavky na nové uživatelské rozhraní

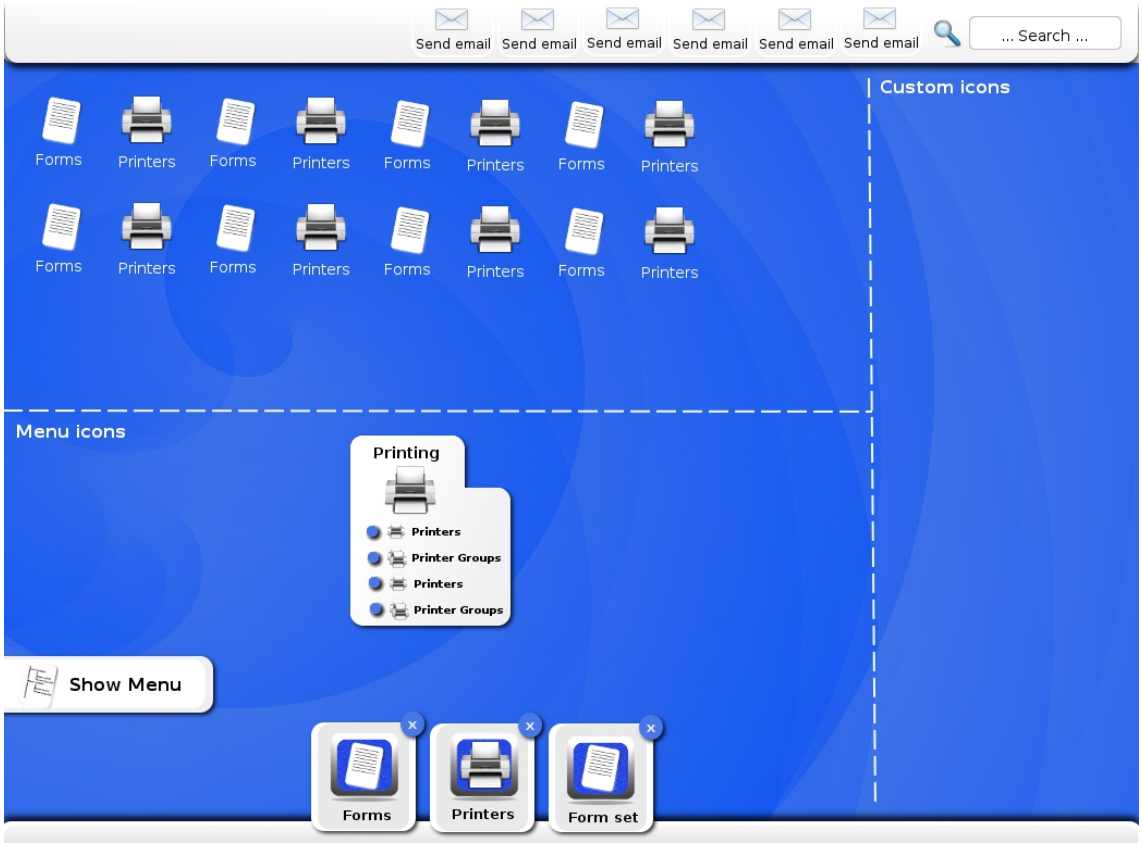
- nový design aplikace – pro modernější vzhled aplikace,
- interaktivita aplikace – pro snadnější užívání aplikace,
- podobná filosofie ovládání jako u Windows,
  - plocha s nejčastěji používanými položkami z menu – pro odstranění nutnosti volby z rozsáhlého menu,
  - práce s více okny najednou – pro možnost zobrazení více oken najednou,
  - minimalizace oken – pro zobrazení dalších oken, které by se nevešly na obrazovku,
  - možnost zmenšování a zvětšování oken – pro práci s okny, aby bylo možné mít zobrazeno více oken najednou,
  - možnost posunu oken – pro práci s okny, aby bylo možné okno posunout tak, aby bylo vidět na další okno.

3.2 První návrhy nového uživatelské rozhraní

První čtyři návrhy byly spíše ideou, jak by se mohla nová aplikace ovládat a jak by mohla vypadat. Nešlo o konkrétní grafický návrh. Pro návrhy byl použit grafický program *GIMP*.

3.2.1 Návrh číslo 1

Návrh, který si klade za cíl téměř odstranit potřebu levého menu.



Obrázek 3.1: Návrh číslo 1

Horní lišta

Položky jsou stejné jako v současné verzi, jen jsou přidány ikony ke všem položkám. Jsou použity zaoblené rohy, aby design působil moderněji.

Plocha

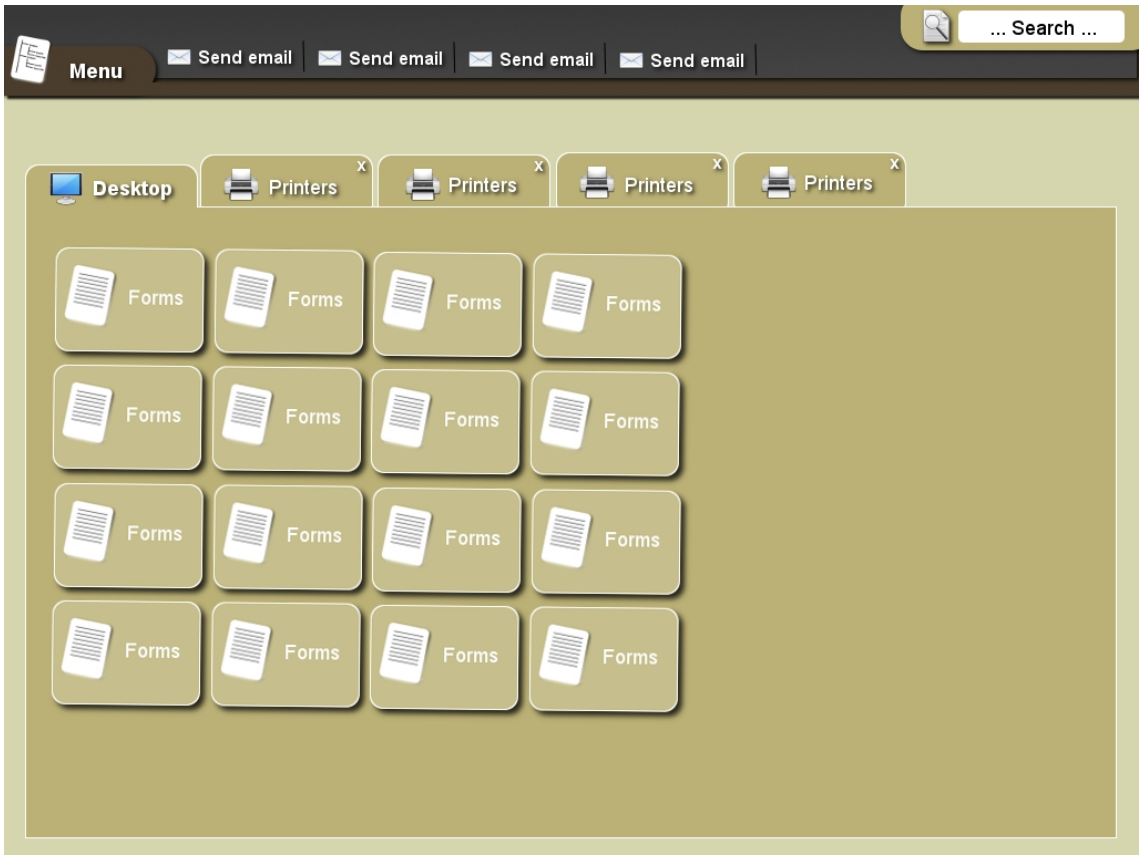
Plocha je rozdělená do tří částí. V horní části jsou ikony, které odpovídají nejpoužívanějším položkám v menu. Levá část plochy jsou zákaznické ikony, které by dovolily uživateli definovat ikony, které by byly odkazy na jiné stránky. Uživatel by tak mohl používat aplikaci *DCIx* jako portál. Spodní část plochy, jsou speciální rozbalovací ikony, které odpovídají určitě části menu. Menu by tak nemuselo být vůbec zobrazeno.

Dolní lišta

Lišta s již otevřenými okny, která je možné znovu otevřít, případně zavřít.

3.2.2 Návrh číslo 2

Návrh, který odstraňuje spodní lištu s otevřenými okny a vše řeší záložkami.



Obrázek 3.2: Návrh číslo 2

Horní lišta

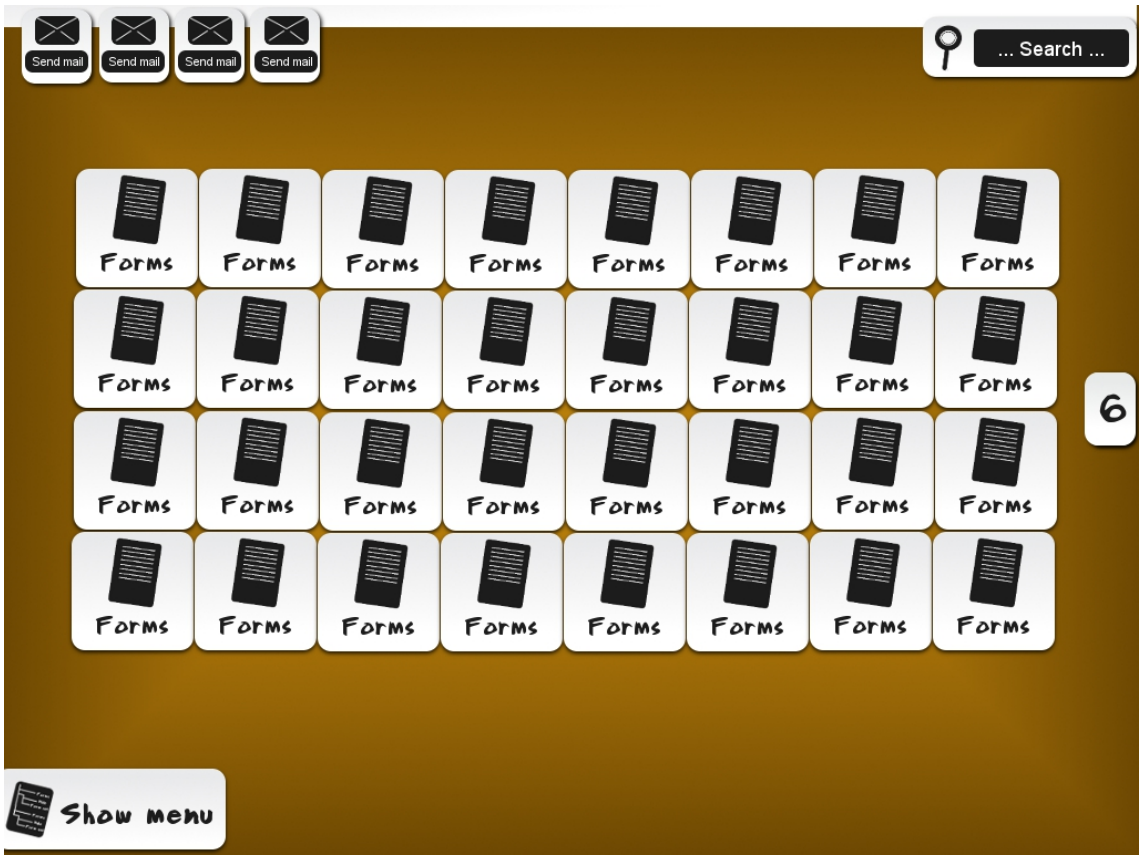
Položky jsou stejné jako v současné verzi, jsou přidány ikony ke všem položkám. Dále jsou použity zaoblené rohy, aby design působil moderněji. Do horního lišty je umístěno menu, které při otevření překryje položky plochy.

Plocha

Plocha obsahuje odkazy na nejpoužívanější položky menu, je to jedna ze záložek. Ostatní záložky prezentují otevřená okna a je možno na ně přepnout, po kliknutí na danou záložku, díky tomu není třeba existence dolní lišty. Přepnutí z okna na plochu se provede po volbě záložky s plochou.

3.2.3 Návrh číslo 3

Pokus o ovládání aplikace, které by bylo podobné systému *Metro*[4] ve *Windows 8*.



Obrázek 3.3: Návrh číslo 3

Horní lišta

Položky jsou stejné jako v současné verzi, jsou přidány ikony ke všem položkám. Použity zaoblené rohy, aby design působil moderněji. Vše vypadá jako „dlaždice“, inspirace *Windows 8*.

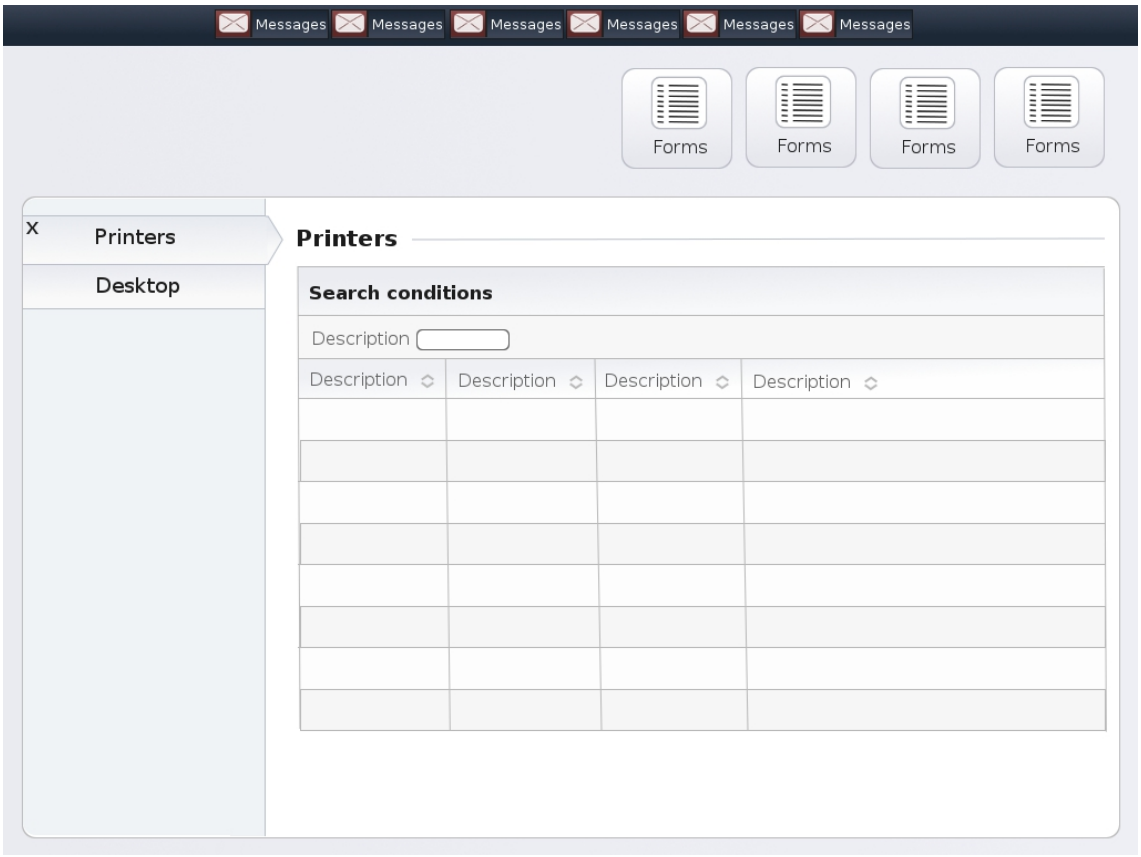
Plocha

Plocha obsahuje odkazy na nejpoužívanější položky menu. Ikony se tváří podobně jako ty v horní liště, podobné současné ploše u *Windows 8*. V pravé části plochy „dlaždice“ s číslicí šest, která prezentuje počet otevřený oken, po kliknutí na ní se zobrazí volba pro vybrání konkrétního okna.

4 *Metro* – uživatelské rozhraní *Windows 8*

3.2.4 Návrh číslo 4

Poslední návrh, který je do jisté míry kombinací těch předchozích.



Obrázek 3.4: Návrh číslo 4

Horní lišta
Položky stejné jako v současné verzi, jsou přidány ikony ke všem položkám. Použity zaoblené rohy, aby design působil moderněji. Pod horní lištou je prostor pro momentálně otevřená okna, místo dolní lišty z předchozích návrhů.

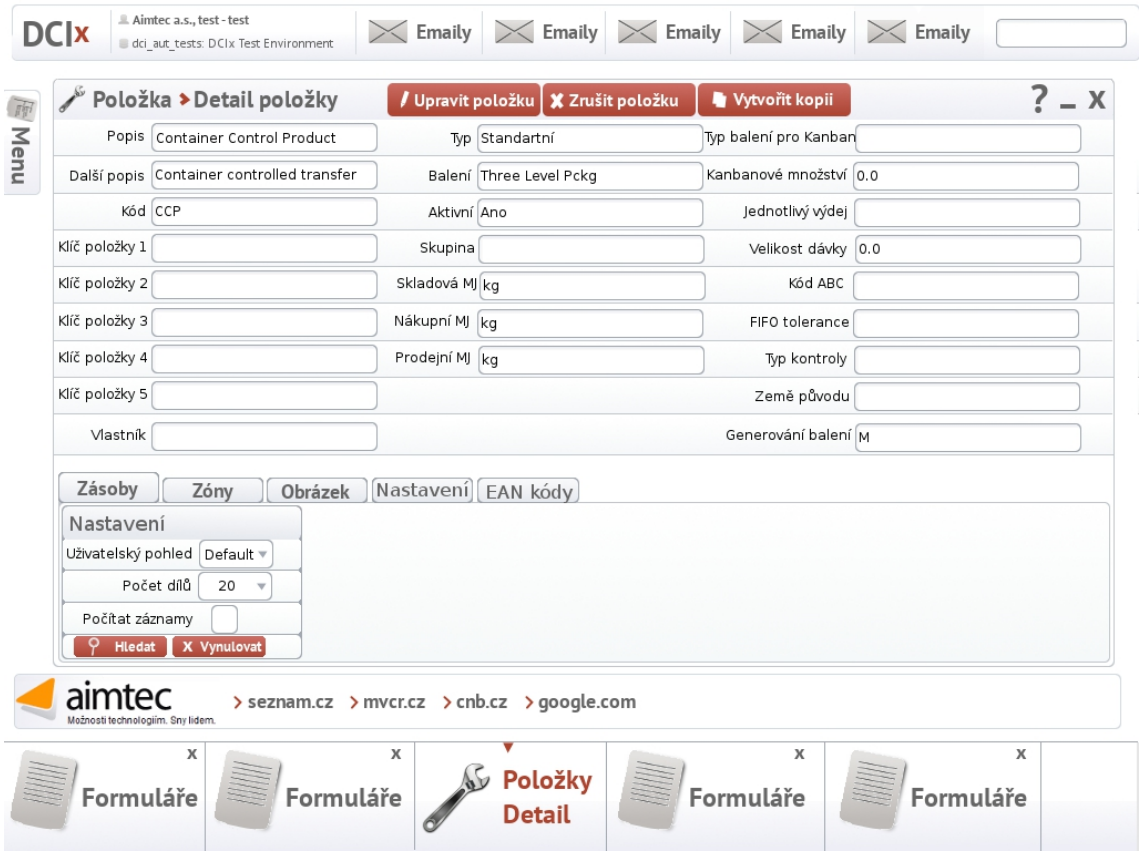
Plocha

Plocha obsahuje odkazy na nejpoužívanější položky menu. Je to podobná verze jako 3.2.2, jen záložky jsou umístěný nalevo.

3.3 Rozpracování dvou variant návrhu

Po diskuzi se zadavatelem práce byly vybrány varianty číslo 2 (viz 3.2.2) a 4 (viz 3.2.4), které byly dále podrobněji rozpracovány. S tím, že oba návrhy budou, co se týče barev laděny do šedé barvy. Oba návrhy také budou obsahovat animace, aby bylo možné demonstrovat zamýšlenou interaktivitu.

3.3.1 Doplněný návrh číslo 2



Obrázek 3.5: Doplněný návrh číslo 2

Horní lišta

Laděno do šedé barvy, s červenými prvky. V horní liště je umístěné logo aplikace a informace o uživateli. Vedle jsou ikony pro všechny události, více vypovídající o jejich významu.

Levé menu

Je zobrazené pouze na požádání, všechny položky mají ikony. Pomocí technologie *drag and drop*[5] je možné přetáhnout položku na plochu a k ní se zobrazí odpovídající ikona.

Plocha

Plocha obsahuje odkazy na nejpoužívanější položky menu. Každá ikona má svůj obrázek pro lepší přehlednost.

Dolní lišta

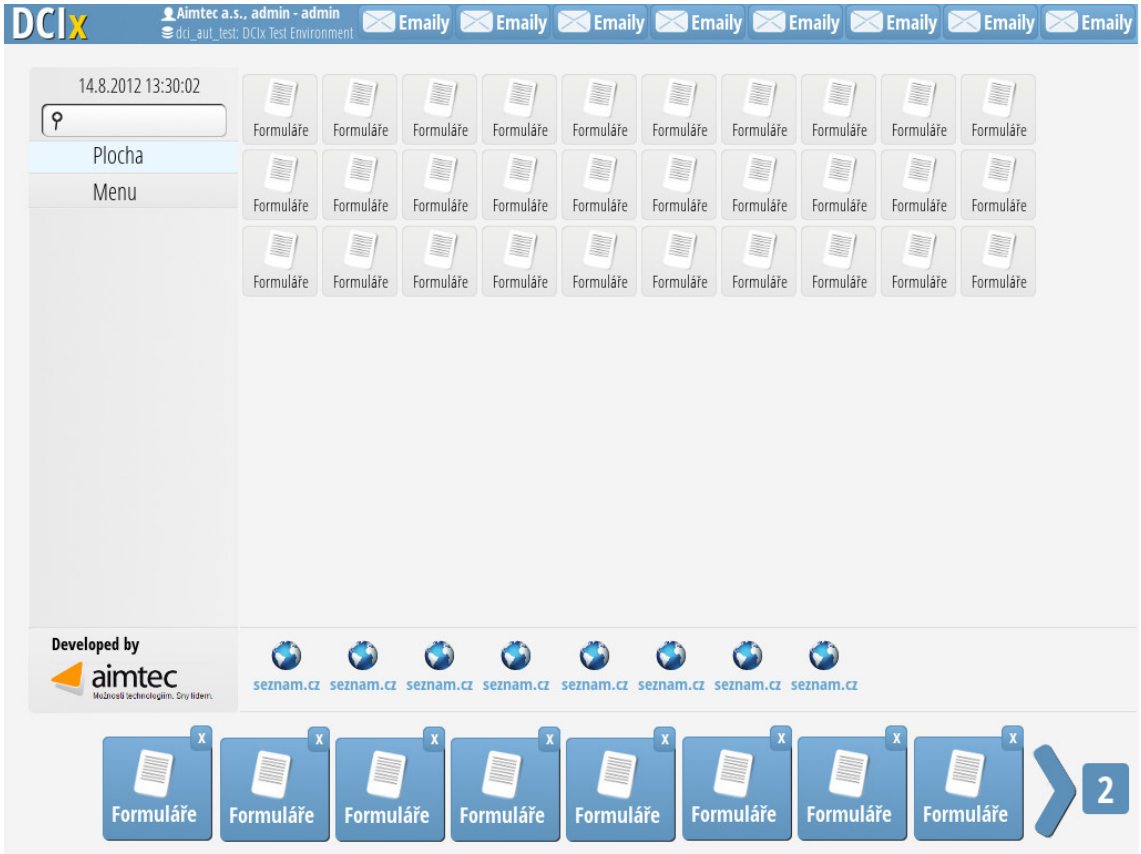
V horní části obsahuje zákaznické ikony (viz 3.2.1), pod nimi momentálně otevřená okna, problém této verze je v tom, kam umístit okna, pokud jich bude minimalizováno více.

5 *Drag and drop* – přetažení objektu z jednoho místa na druhé pomocí kurzoru myši

Animace

K návrhu jsou připojeny tři animace, které ukazují přetažení položky z menu na plochu, minimalizaci otevřeného okna a změnu podbarvení položek v horní liště.

3.3.2 Návrh číslo 4



Obrázek 3.6: Doplněný návrh číslo 4

Horní lišta

Je laděna do hnědé barvy, s bílými prvky. V horní liště je umístěné logo aplikace a informace o uživateli. Dále ikony pro všechny události, více vypovídající o významu.

Levé menu

Je zobrazené stále, je možné zobrazit plochu kliknutím na vybranou položku v menu. Pomocí technologie *drag and drop* je možné přetáhnout položku z menu na plochu. V menu nejsou oproti návrhu číslo 2 (viz 3.2.2)zapracovány ikony k položkám.

Plocha

Plocha obsahuje odkazy na nejpoužívanější položky menu. Každá ikona má vlastní obrázek pro lepší přehlednost. Pokud není žádný nastaven, je použit přednastavený.

Dolní lišta

V horní části obsahuje zákaznické ikony (viz 3.2.1). Pod nimi momentálně otevřená okna. Těch může být libovolný počet, když už není místo na obrazovce, je možné posunovat se dále šipkami

Animace

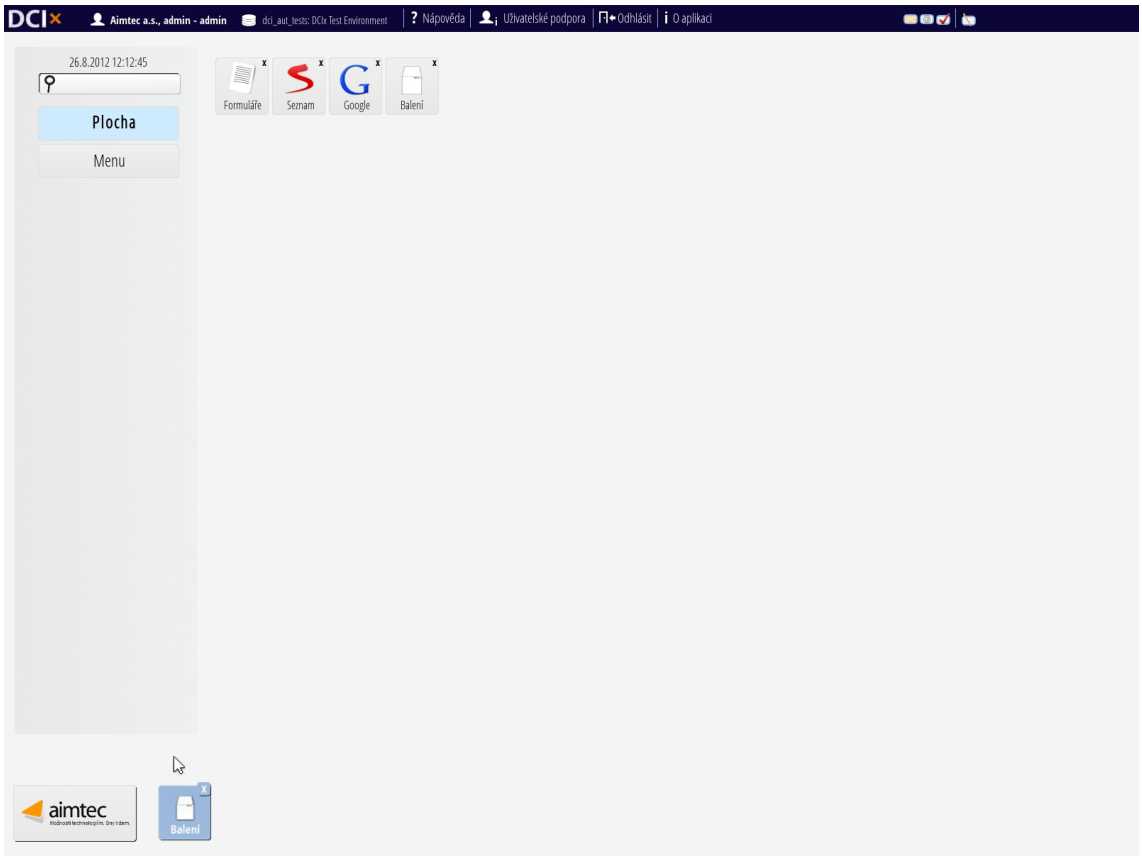
K návrhu jsou připojeny tři animace, které ukazují přetažení položky z menu na plochu, minimalizaci otevřeného okna a posun otevřených oken pomocí šipek.

3.4 Návrh nového uživatelské rozhraní

Tyto dvě varianty byly prezentovány vedoucímu diplomové práce a na základě jeho připomínek byly provedeny změny, které se však vesměs týkaly barev, které byly změněny na více pastelové.

3.4.1 Připomínky managementu

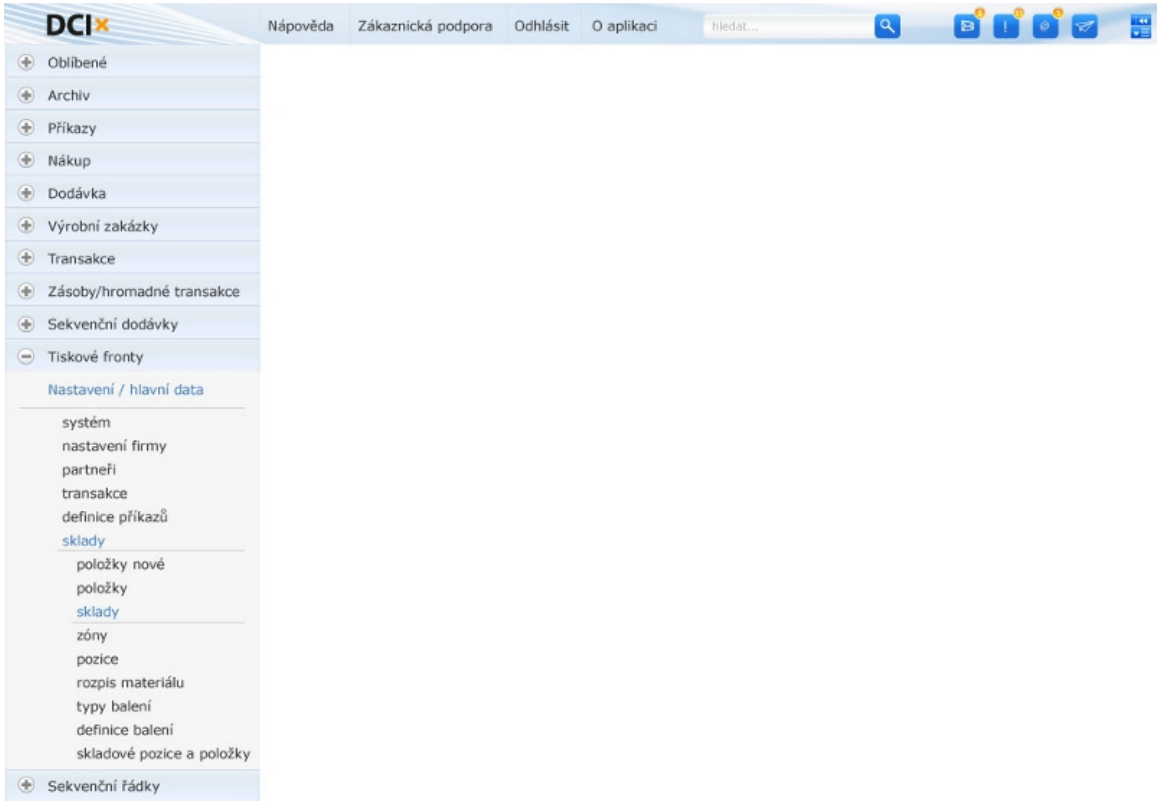
Dále byly varianty prezentovány zadavatelům práce, kteří jako výsledný návrh zvolili ten druhý. Dle jejich dalších připomínek byl tento návrh dále dopracován.



Obrázek 3.7: Návrh po připomínkách managementu

3.4.2 Konečný návrh uživatelského rozhraní

Konečný návrh, který byl realizován, co se týče barevné kombinace zabolení tlačítek a vzhledu ikon navrhl designer. Plocha a zobrazení minimalizovaných oken bylo převzato z návrhu 3.4.1 s drobnými barevnými úpravami.



Obrázek 3.8: Finální návrh

4 Realizace nového uživatelského rozhraní

Pro realizaci Byly použity technologie, které se v aplikaci již používají, aby nebylo nutné příliš zasahovat do současné podobny aplikace. Nově byla použita technologie *CSS3*, hojněji byl používán *Ajax* a došlo k výběru vhodného javascriptového frameworku.

4.1 Výběr JavaScriptového frameworku

Pro interaktivitu nového uživatelské rozhraní byl zvolen *JavaScript*[6], bylo ovšem nutné zvolit vhodný javascriptový framework, který by pokryl všechny potřeby na nové uživatelského rozhraní. Na základě dřívějších zkušeností a požadavků zadavatele, by měl framework poskytovat:

- Hledání ve struktuře stránky dle specifických selektorů
- *Ajax* (viz 4.2.5) – odesílání dat touto technologií pro interaktivitu aplikace
- *JSON* (popsáno v 3.2.5) – použití formátu *JSON* pro příjem dat
- Nástroje pro práci s *UI*[7] – posun elementů, změna velikosti elementů

Existuje řada frameworků, v rámci vhodnosti pro nové uživatelské rozhraní, byly zkoumány tři následující.

EXT JS

Tento framework nabízí všechny požadované dovednosti, navíc má opravdu bohatou knihovnu *UI* komponent, jako jsou kalendáře atd.

DOJO

Také nabízí všechny požadované funkčnosti, ovšem syntaxe se jeví jako komplikovaná. Hlavně u komponent pro posun a změnu velikosti.

JQuery

Nabízí všechnu požadovanou funkčnosti a ve společnosti Aimtec se již používá. Také při hledání řešení problémů, nebo při zjišťování, jak daná funkčnost frameworku funguje se o *JQuery* dá najít nejvíce. Pro vývoj byl tedy zvolen tento framework.

4.2 Použité technologie pro uživatelské rozhraní

4.2.1 JQuery

JavaScriptový framework, který používají například společnosti jako Google nebo Microsoft [JQuery]. Jedná se o multiplatformní knihovnu, která je zadarmo. Soubor obsahující kompletní *JQuery* má v současné verzi pouhých 32 kb. Připojení knihovny k webové stránce, je velmi

6 *JavaScript* – skriptovací jazyk, prováděný na straně klienta
7 *UI* – User Interface – uživatelské rozhraní

jednoduché, stačí připojit tento jeden javascriptový soubor, ve kterém je vše potřebné. Informace uvedené v kapitole o *JQuery* byly čerpány z [JQBook]

Syntaxe JQuery

Syntaxe *JQuery* je velmi jednoduchá, zjednodušeně se dá popsat takto.

```
$("prvek, se kterým chceme něco udělat").funkce("parametr/y funkce");
```

Knihovna JQuery nabízí tyto možnosti:

- Manipulace s *DOM*[8]*/HTML (viz 4.2.7)*– pro úpravu, nebo přidávání elementů

```
$('#icon').remove() // odstrani element s id icon
```

- Manipulace s *CSS* (viz 4.2.8) – pro změnu vlastností prvků

```
$('p').css('color', 'red') // elementu p nastavi cervene pismo
```

- Události – pro zajištění správné reakce na podněty v aplikaci

```
// po kliknuti na prvek s tridou hideIcon skryje prkven s id icon
$(document).on('click', '.hideIcon',
                    function() {
                      $('#icon').hide()
                    }
                   );
```

- Efekty a animace – pro lepší vzhled aplikace a její interaktivitu

```
// animuje posun objektu na pozici 200, 500
$(this).css('position', 'absolute').animate(
{
  left : 200,
  top : 500
});
```

- Ajax – použití ajaxového volání (viz 5.8.6)
- Utility – pro zajištění dalších požadavků a funkčností, které nejsou součástí *JQuery* (viz 5.7.2)

4.2.2 Java Framework Struts

Technologie *Java Struts* je open-source framework pro vytváření webových aplikací od společnosti Apache. Framework je založen na technologii *JSP (*viz 4.2.3*)* pro generování stránek, servletech a *JavaBeans*. Používán je návrhový vzor *MVC* (viz 2.4) Informace o frameworku *Struts* čerpány z [Struts MVC] a [StrutsBook].

8 *DOM* – Document Object Model

JavaBeans

JavaBeans jsou komponenty pro *Javu*, prakticky jsou to třídy, které zapouzdřují více objektů do jednoho a povolují přístup k atributům pomocí *getterů*[9] a *setterů*[10].

Základní terminologie

- *ActionForm* – jeho atributy odpovídají datům posílaným v *HTTP* požadavku
- *ActionMapping* – namapování konkrétní akce na dané *URL*
- *ActionForward* – mapování *JSP* stránky, která se zobrazí uživateli

MVC ve Struts

- *Model* obsahuje *business logiku* a interakci s persistentním úložištěm dat. Má také za úkol manipulovat s daty.

- *View* zobrazuje požadovaná data přímo uživateli. Ve *Struts* se k tomu používá technologie *JSP*.

- *Controller* zpracovává všechny žádost od uživatele a vybírá vhodný pohled, který mu vrátí zpátky. *Struts* používají pro práci *Controlleru* třídu *ActionServlet*.

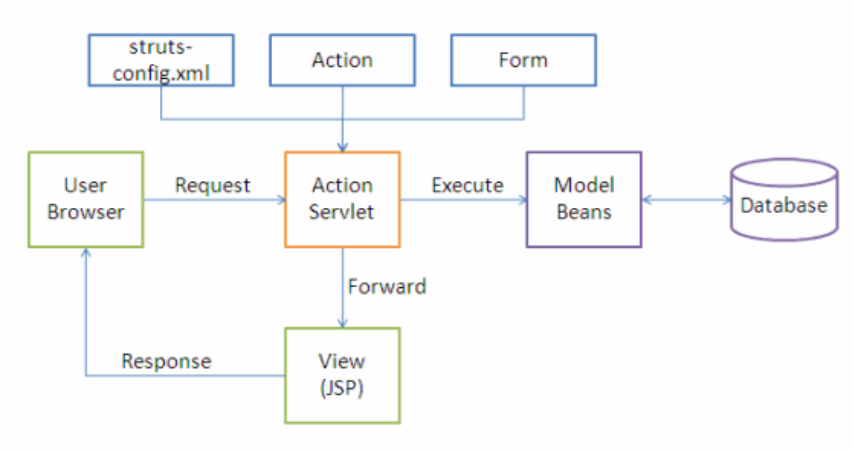
Struts-config.xml

Důležitý konfigurační soubor, jsou v něm nastaveny všechny potřebné informace pro *Struts*. V první řadě je třeba nastavit cestu k *ActionForm*, jehož atributy odpovídají datům, posílaným v *HTTP*[11] požadavku. Po zavolání *URL*[12], které je určeno atributem `path` tagu `action`, dojde ke spuštění třídy *ActionServlet*, cesta k ní je nastavena pomocí atributu `type` tagu `action`. Třída *ActionServlet* provede požadovanou manipulaci s daty a výsledky zobrazí uživateli na *JSP* stránce, definované v atributu `path` tagu `forward`.

```
<form-bean name="favoriteForm"
type="cz.aimtec.dci.menu.FavoriteMenuChangeForm" />

<action
      path="/changeFavoriteMenu"
      type="cz.aimtec.dci.menu.action.FavoriteMenuChangeAction"
      name="favoriteForm"
      scope="request">
<set-property property="authClass"
 value="cz.aimtec.dci.action.DciActionAuthentication"/>
<forward
      name="success"
      path="/template/page/default_leftMenuFavorites.jsp" />
</action>
```

9 *Getter* – metoda pro zjištění hodnoty atributu
10 *Setter* – metoda pro nastavení hodnoty atributu
11 *HTTP* – HyperText Transfer Protocol – protokol pro výměnu *HTML* dokumentů
12 *URL* – Uniform Resource Locator – identifikuje stránku

Podrobnější popis celé funkcionality *Struts* je na Obrázek 4.1



Obrázek 4.1: Princip funkce Struts (Převzato z [Struts MVC])

Následující akce nastanou, pokud klientský prohlížeč vyšle *HTTP* požadavek:

- *ActionServlet* obdrží požadavek od prohlížeče,
- *stuts-config.xml* obsahuje informace k odpovídající *Action* – *ActionForm, ActionMapping* a *ActionForward,*
- uloží hodnoty z požadavku do *JavaBean* třídy *ActionForm,*
- rozhodne, která *Action* třída má požadavek dále zpracovat,
- zkontroluje obdržená data,
- *Action* třída zpracuje data s pomocí komponenty *model,* která komunikuje s databází,
- po dokončení požadavku *Action* třída vrátí *ActionForward*, tedy odkaz na *JSP* stránku *controlleru,*
- pPodle *ActionForward* třídy *controller* vyvolá odpovídající *view,*
- *HTTP* odpověď je vrácena zpátky uživateli a vykreslena pomocí komponenty *view.*

Struts nabízejí vlastní *taglib*[13], které se používají v *JSP*

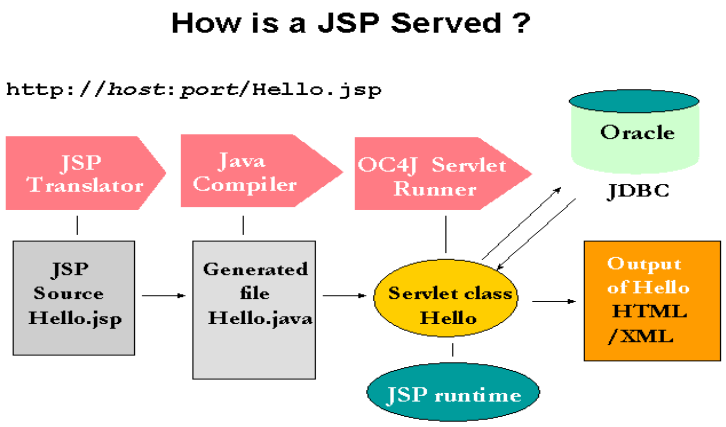
- *Bean* – pro práci s *Beans* – pro vypisování hodnot atributů objektů
- *HTML* – pro tvorbu *HTML* značek, například formulářů
- *Logic* – nabízí možnosti pro logické operace nebo procházení cyklů

4.2.3 JSP

Java Server Pages je technologie pro tvorbu dynamických stránek, založených na Javě. *JSP* je vlastně *HTML* stránka, do které se pomocí speciálních značek vkládá kód Javy. Na rozdíl od

13 *Taglib* – knihovna značek

PHP[14] nebo *ASP*[15] má *JSP* silnou typovou kontrolu z Javy. *JSP* stránky jsou kompilované. Takže při volání *PHP* stránky se stránka interpretuje vždy znovu, kdežto u *JSP* se kompiluje jen jednou a pak až při případné změně. Obsluha *JSP* stránky je znázorněna na obrázku Obrázek 4.2. Informace o JSP byly čerpány z [JSPBook].



Obrázek 4.2: Obsluha JSP stránky (Převzato z [OracleDoc])

Vložení Java kódu do JSP

```
<% ... Java kód ... %>
pro vypsání hodnoty
<%= ... Java výraz ... %>
<% String s = "text"; %>
<b>text:</b> <%= s %><br>
```

Vkládání jiných JSP stránek

JSP dovoluje vkládat jednotlivé stránky do sebe, také je možné si vytvořit hlavičku nebo patičku stránky pouze jednou a pak jí při potřebě vkládat. Navíc je možné *JSP* stránky parametrizovat.

```
<%@ include file="hlavni.jsp" %>
...
<jsp:include page="login.jsp">
  <jsp:param name="login" value="<%= register.getLogin()%>"/>
  <jsp:param name="heslo" value="<%= register.getHeslo()%>"/>
</jsp:include>
```

14 *PHP* – Hypertext Preprocessor – technologie pro tvorbu webových stránek
15 *ASP* – Active Server Pages – technologie pro tvorbu webových stránek

4.2.4 DB Framework Alfa

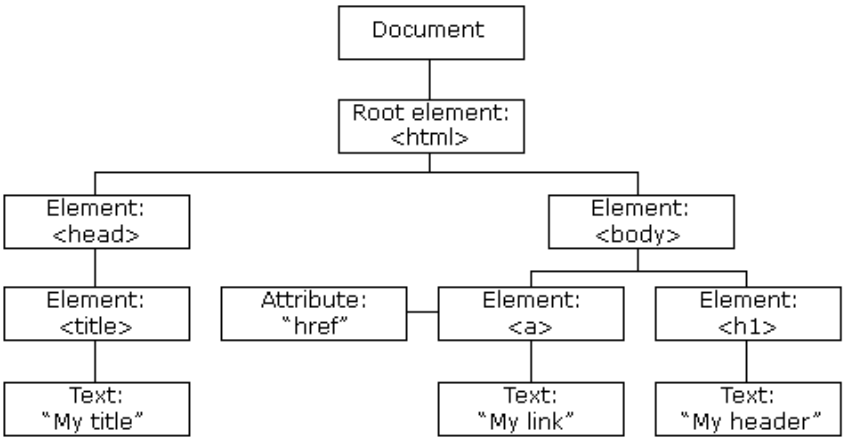
Framework vyvinutý ve společnosti Aimtec. Je napsán jazyce *Java* a pracuje nad databází *MS SQL*[16]. Stará se o komunikaci s databází a mapování tabulek na objekty. Poskytuje metody pro práci s daty, uloženými v tabulce. Pro práci s tabulkou se používají následující objekty.

- *Alfa* – předek všech Alfa objektů, stará se o logování
- *AlfaObject* – poskytuje metody pro práci s tabulkou – *read, update*, *delete*, *getTable*
- *AlfaCollection* – poskytuje metody pro čtení dat z tabulky – *readFirstPage*, *fetchRow*
- *Table* – ukládá data z dotazů do objektů a naopak
- *ColumnMapping* – nastavení mapování sloupců tabulky na objekty
- *table-mappings.xml* – nastavení cest k potřebným Java třídám

4.2.5 AJAX

Ajax je zkratka *Asynchronous JavaScript and XML*[17]. Je to obecné označení technologie vývoje interaktivních webových aplikací, které mění obsah stránek bez nutnosti jejich znovu načítání. *Ajax* umožňuje asynchronní[18] změnu obsahu, pomocí výměny malého množství dat se serverem. Interaktivní aplikace jsou vyvíjeny s využitím dalších technologií, které jsou popsány dále. Informace v této kapitole jsou čerpány z [Ajax] a [AjaxBooks].

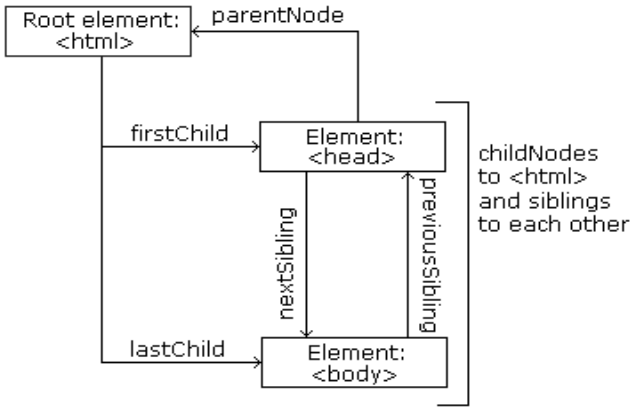
DOM
Document Object Model, jedná se objektovou reprezentaci *XML* nebo *HTML* dokumentu. *HTML DOM* je *API*[19] umožňující přístup a manipulaci s *HTML* dokumentem. *DOM* prezentuje *HTML* jako stromovou strukturu.



Obrázek 4.3: Stromová struktura HTML dokumentu (Převzato z [DOM])

16 *Microsoft SQL Server* – relační databázový server
17 *XML* – Extensible Markup Language – obecný značkovaní jazyk
18 *Asynchronní* – při komunikaci nečeká na odpověď a pracuje dále
19 *API* – Application Programming Interface

K jednotlivým uzlům stromu je možno přistupovat pomocí *JavaScriptu*. Všechny *HTML* uzly mohou být měněny, nebo také vytvářeny, či mazány. Uzly mají hierarchický vztah ke všem ostatním uzlům ve stromu. Termíny rodič, potomek a sourozenec jsou používány k popsání vztahů mezi uzly. Rodičovský uzel má potomky, potomci na stejné úrovni se nazývají sourozenci.



Obrázek 4.4: Vztahy mezi uzly (Převzato z [DOM])

- horní uzel se nazývá kořen,
- každý uzel má právě jednoho rodiče, kromě kořene, který nemá žádného rodiče,
- uzel může mít libovolný počet potomků,
- sourozenci jsou uzly, které mají stejného rodiče.

XMLHttpRequest

Je *API* dostupné ve skriptovacích jazycích, prováděných na straně klienta. Je používané k přímému posílání *HTTP* nebo *HTTPS*[20] požadavku webovému serveru a k načtení odpovědi od serveru zpátky přímo do skriptu. Data, která přichází zpátky od serveru, mohou být ve formátu *JSON*, *HTML*, *XML* nebo jako prostý text. Data lze technologií *DOM* načíst přímo do konkrétního uzlu, bez nutnosti znovu načtení stránky.

Použití Ajaxu

- **Ajax výhody**
  - Není nutné znovu načíst celou stránku
  - Uživatelská přívětivost – interaktivita
  - Snížení zátěže serveru – načítá se jen část stránky
- **Ajax nevýhody**
  - Uživatel nemusí poznat, že se jeho požadavek již zpracovává a pošle ho znovu – zvýšení zatížení.

20 *HTTPS* – Hypertext Transfer Protocol Secure – *HTTP* využívající asymetrickou šifru

- Nemění se *URL*, nelze používat například šipky zpět a dopředu v prohlížeči – lze to vyřešit.
- Používaný prohlížeč musí podporovat *AJAX* technologie – dnes už většinou bez problémů.

4.2.6 JSON

JavaScript Object Notation je způsob zápisu dat, který je textový a nezávislý na platformě. Nevýhodou je však to, že nedovoluje definovat použitou znakovou sadu. *JSON* používá syntaxi *JavaScriptu* pro popis datových objektů. Data ve formátu *JSON* mohou být využívána technologií *AJAX*, je to rychlejší, než v případě použití *XML*. [JSON]

Datové typy formátu JSON jsou:

- číslo – integer nebo float
- řetězec – v apostrofech
- logický datový typ – boolean – true nebo false
- pole – v hranatých závorkách
- objekt – ve složených závorkách

Data ve formátu *JSON* jsou zapsána jako dvojice klíč a odpovídající hodnota.

```
"jmeno":"Josef"
```

Objekt je zapsán ve složených závorkách a obsahuje více párů klíč a hodnota.

```
{ "jmeno":"Josef" , "prijmeni":"Novák" }
```

Pole je zapsáno v hranatých závorkách a obsahuje více objektů.

```
{
"zamestnanci": [
{ "jmeno":"Jan", "vek": 25 },
{ "jmeno":"Petr", "vek": 17 },
{ "jmeno":"Pavel", "vek": 44 }
]
}
```

4.2.7 HTML

HyperText Markup Language je značkovací jazyk, je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka *SGML*[21]. *HTML* využívá hypertext, což je nelineární způsob strukturování textu, které odkazuje na další informace. [PIA-HTML]

21 *SGML* – Standard Generalized Markup Language – metajazyk umožňující definovat značkovací jazyky

HTML používá tyto prvky pro popis obsahu:

- **Značky** – vyznačují elementy obsahu
  - Párové značky – `<p>paragraf</p>`
  - Nepárové značky – `<br />`
- **Atributy** – popisují vlastnosti elementu, jsou u otevírací značky
  - `<a href='odkaz.html'>klikni zde</a>`
- **Entity** – pro zápis speciálních znaků, které jsou například použity v samotném HTML
  - `&lt;` na stránce pak zobrazí `<`

HTML dokument má následující strukturu:

- **Preambule** – určuje verzi a kódování
- **Deklarace** – deklarace typu dokument, odkazuje na *DTD*, které určuje interpretaci dokumentu
- **Záhlaví** – `<head>` – obsahuje meta-informace o dokumentu, nezobrazuje se
  - **Značky v záhlaví**
    - *title* – nadpis dokumentu
    - *meta* – metainfo – autor, klíčová slova atd.
    - *link* – odkazy na úrovni souborů, připojení CSS souborů, nebo skripty
    - a další
- **Tělo dokumentu** – `<body>` – obsah dokumentu
  - **Obsahuje vnořené elementy**
    - *Blokové* – zalamují odstavec – `<h1>...<h1>, <p>...</p>` a další
    - *Textové* – uvnitř blokových – `<em>...</em>, <b>…</b>` a další
    - *Hypermediální elementy* – obrázky, odkazy – `<img>...</img>, <a>...</a>`
    - **Generické kontejnery**
      - *blokový* – `<div>...</div>`
      - *řádkový* – `<span>...</span>`
    - **Formulářové** – pro interakci s klientem `<form>...</form>`
      - **Metody odesílání**
        - *GET* – odesílaná data jsou vidět v *URL*
        - POST – odesílaná data nejsou vidět v *URL*
      - *Formulářové prvky* – input, radio button, check box atd
    - **Tabulky** – popis dat s tabelární strukturou – `<table>...</table>`

- **Atributy –** důležité jsou také atributy pro stylování, které mohou být u všech výše popsaný elementů
  - *id* – jednoznačně identifikuje element
  - *class* – více elementů může mít stejnou třídu
  - s*tyle* – zápis v syntaxi *CSS*
- **Rámce** – vložení jiné *HTML* stránky do té stávající, i dnes ještě používané, například komentáře k článkům přes facebook `<iframe>...</iframe>`

4.2.8 CSS

Cascading Style Sheets definuje, jak se budou zobrazovat *HTML* elementy. *CSS* je deklarativní, nestrukturovaný jazyk, který se skládá z pravidel. Slouží k oddělení obsahu (*HTML*) od prezentace/formátování (*CSS*). *HTML* dává značce význam, jestli se jedná o odstavec, nadpis atd. Ale o tom, jakou bude mít barvu, jaké bude použito písmo rozhoduje právě *CSS*. *HTML* značky, které tohle umožňovaly (jako `<font>`), se už dnes nepoužívají. [PIA-CSS]

Syntaxe:

- **selektor** – co se bude formátovat,
- **deklarace** – jak se to bude formátovat, skládá se z vlastnosti a hodnoty.

Způsoby zápisu CSS:

```
h1{
 font-size: 12px;
 color: red;
 font-weight: bold;
}
```

Text v nadpisech první úrovně `h1` bude o velikosti `12px`, červeně a tučně.

Připojení CSS k HTML:

- externí CSS soubor
  - přes tag link v záhlaví stránky `<link rel="stylesheet" type="text/css" href="style.css">`
  - @import deklarace – `@import url(http://style.com/basic)`
- interní styly v záhlaví stránky – element `<style>...</style>`
- atribut style u elementu.

Selektory:

- elementy – názvy elementů – `body, p, a, …`
- třídy – elementy s danou třídu – `.trida`
- id – element s konkrétním id – `#menu`
- pseudotřídy
  - po najetí kurzorem na odkaz – `a:hover`
  - prvním písmeno v odstavci – `p:first-letter`
  - a další
- kombinace selektorů – odstavec s nastavenou třídu horni – `p.horni`
- pokročilé selektory
  - libovolný element – `*`
  - přímý potomek – `p > em`
  - atribut má danou hodnotu – `p[title='titulek']`
  - a další

Vlastnosti:

- klasifikace druhu obsahu – `display: block`
- textové, barvy a pozadí – `color: blue`
- pozicování, stránkování – `float: left`

Dědění vlastností:

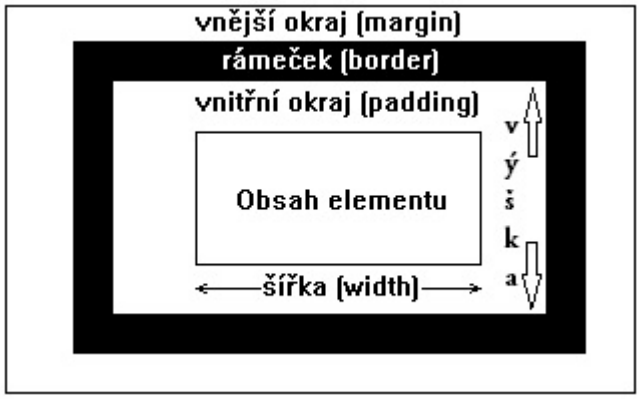
Většina vlastností definovaných pro určitý *HTML* uzel, se dědí na jeho potomky. To znamená, že pokud je nastaveno pro `<body>` zelené písmo, tak písmo bude zelené i ve všech dalších elementech uvnitř `<body>`, pokud to nebudou mít nastaveno jinak. Příkladem vlastností, které se nedědí, jsou například `border` a `margin` a další.

Je nutné vyřešit konflikty pro různé deklarace stejné vlastnosti. Používá se priorita hodnot:

- vlastnosti s označením *!important* se dají dopředu,
- seřazení podle původu *autor* > *čtenář* > *prohlížeč*,
- seřazení dle specifikčnosti selektoru *ID nebo atribut style* > *class* > *kontextový selektor* > *typový selektor.*

Formátovací model

Určuje velikost elementu, jeho rámeček a mezery mezi okraji.



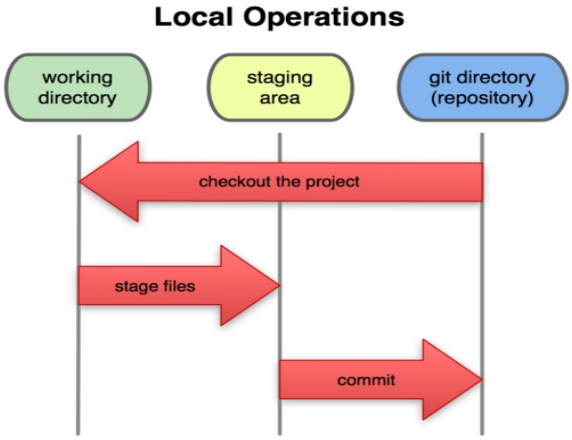
Obrázek 4.5: Formátovací model CSS (Převzato z [BOXMODEL])

4.3 Verzovací systém Git

Jedná se o distribuovaný verzovací systém, který každému vývojáři poskytuje lokální kopii celé historie vývoje. Oproti nástrojům jako je třeba *CVS*[22] a *SVN*[23] je *Git* jiný například v tom, že rozděluje *commit*, tedy uložení změn do repositáře do dvou kroků. Téměř všechny operace v *Gitu* jsou lokální, takže nejsou potřeba informace ze síťových uzlů. Vše se načte z lokální databáze, například historie změn je rychlejší než u jiných systémů.

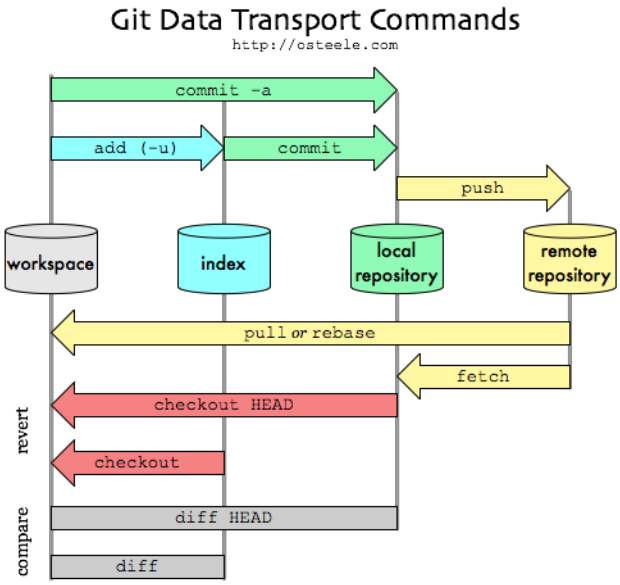
Projekt je v systému *Git* rozdělen do tří hlavních částí: adresář systému *Git* (*Git directory*), pracovní adresář (*working directory*) a oblast připravených změn (*staging area*). Git používá pro soubory tři základní stavy: zapsáno (*committed*), změněno (*modified*) a připraveno k zapsání (*staged*). Zapsáno znamená, že jsou data uložena v lokální databázi. Změněno znamená, že v souboru byly provedeny změny, avšak soubor ještě nebyl zapsán do databáze. Připraveno k zapsání znamená, že změněný soubor bude zapsán v další revizi. Informace v této kapitole byly čerpány z [GIT].

22 *CVS* – Concurrent Version System – systém pro správu projektů
23 *SVN* – Subversion – systém pro správu a verzování zdrojových kódů



Obrázek 4.6: Pracovní adresář, oblast připravených změn a adresář Git (Převzato z [GIT])

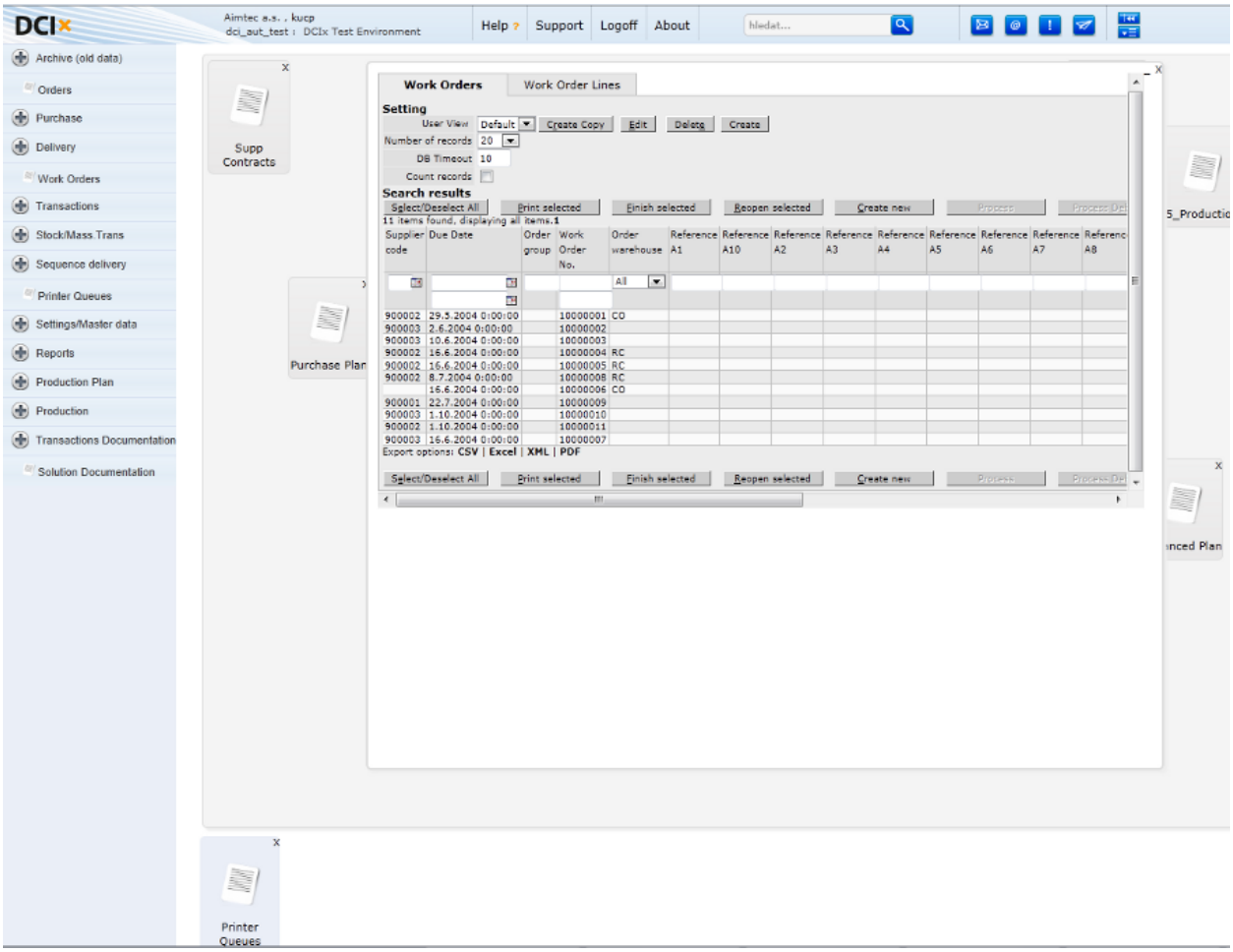
Pro uložení změn do vzdáleného repositáře je nutné nejprve udělat c*ommit*, který zapíše data do lokální databáze a pak *push*, který je zapíše do vzdáleného repositáře. Více možností je popsáno na následujícím obrázku.



Obrázek 4.7: Git příkazy (Převzato z [GitImage])

5 Implementace uživatelského rozhraní

Obrazovka je rozdělena do čtyř částí, které byly implementovány postupně. Vývoj probíhal postupně. Od nejjednodušších částí po ty nejsložitější. Ke správě verzí byl používán systém Git (viz 4.3). Současné *JSP* stránky byly zachovány, tak aby bylo možné vrátit se ke staré verzi aplikace. Proto všechny *JSP* stránky, které mají nahradit ty staré, mají postfix *new.* Názvy tříd již existujících elementů, které mají pouze jiný vzhled, či upravenou funkci, byly ponechány stejné. Tyto názvy se používají v automatických testech a bylo nutné je zachovat. Soubor *newApplication.css*, který určuje vzhled nové verze, je ke stránce připojen jako poslední, takže pravidla z něj mají nejvyšší prioritu.



Obrázek 5.1: Nová verze aplikace

5.1 Struktura aplikace

Aplikace *DCIx* je rozsáhlá, celkově obsahuje přes 17 000 souborů (zdrojové, konfigurační atd). Struktura aplikace je následující, popis je orientační a zjednodušený.

- **DCI**
  - **Accept Test Source** – *Jameleon*[24] tagy
  - **Common configuration** – připravené konfigurační soubory s nastavením
  - **DB** – databázové skripty
  - **JavaSource** – zdrojové soubory Javy
  - **JsTestSource** – testy pro *JavaScript*
  - **TestCases** – testovací scénáře v nástroji *Jameleon*, nastavení nástroje *Jameleon*
  - **Tools** – externí nástroje – *Tomcat*, *DocBook*[25] atd
  - **T-SQL Test Source** – testy pro *SQL*
  - **UnitTest lib** – knihovny pro unit testy
  - **UnitTest Source** – zdrojové soubory jednotkových testů
  - **WebContent** – obsahuje *JSP* šablony, *CSS* šablony, *JS* skripty, obrázky
  - **WebServices** – webové služby

5.2 Postup pro nahrazení JSP stránek

Ke zjištění toho, jaká *JSP* stránka, nebo *Java* třída se má upravit, aby bylo možné provést požadované změny, se používá následující postup.

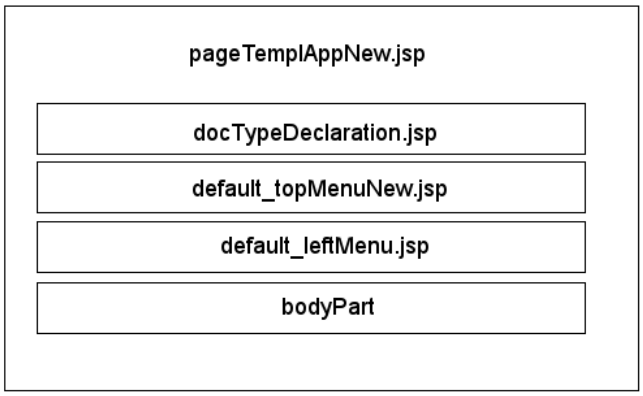
- ve zdrojovém kódu stránky ve formátu *HTML* se z tagu `<meta name="head-page-path" content="/DCI/startEntryNew.jsp">` z atributu `content` zjistí název *JSP* stránky,
- v konfiguračním souboru *struts-config.xml* se dohledá odpovídající akce ke zjištěné *JSP* stránce, akce se nachází v tagu `action` atributu `type`.

```
<action ... type="cz.aimtec.dci.action.StartPageAction" ...
```

5.3 Struktura JSP Stránek

V aplikaci se používá šablona *pageTemplAppNew.jsp*, kterou používá každá generovaná stránka, struktura je následující.

24 *Jameleon* – testovací framework
25 *DocBook* – značkovací jazyk pro tvorbu dokumentace



Obrázek 5.2: Základní šablona

- *docTypeDeclaration.jsp* – deklarace *Doctype*[26] pro *HTML* dokument
- *default_topMenuNew.jsp* – horní menu aplikace
- *default_leftMenu.jsp* – levé menu aplikace
- *bodyPart* – do této části se vkládá šablona s plochou a otevřenými okny, v předchozí verzi se do této části dynamicky vkládaly šablony odpovídající danému formuláři

5.4 Ukládání dat

V některých případech je potřeba uložit další informace o daném *HTML* elementu, například odpovídající *id* z databáze k položce v menu. V aplikaci *DCIx* se používá uložení do atributů s předponou *_dci*. Manipulace s nimi je díky *JQuery* jednoduchá.

```
$('#activeWindow1').attr('_dci_windowid', 12)
$('#activeWindow1').attr('_dci_windowid')
```

JQuery také nabízí uložení dat přímo do odpovídajícího elementu. Stará se o to funkce *data*, která umožňuje uložit i složitější struktury.

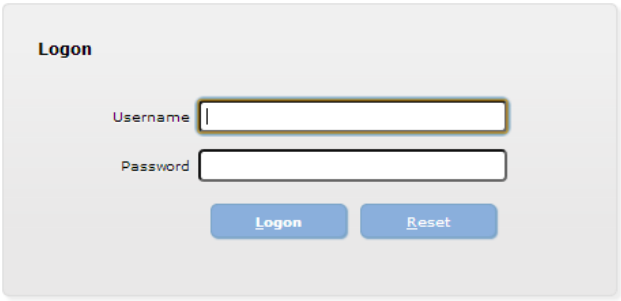
```
$('#activeWindow1'.data('isMinimized', 0);
$('#activeWindow'1).data('isMinimized');
```

V případě aplikace *DCIx* se ale vždy ukládá pouze řetězec nebo číslo, takže stačí využít možností uložení do atributů. Navíc tento způsob se již v aplikaci používá, možnosti funkce *data* tak nebyly použity.

26 *Doctype* – obsahuje informace o verzi použitého *HTML*

5.5 Přihlašovací stránka

Jako první byla změněna stránka pro přihlášení uživatele. Jedná se pouze o změnu *CSS* a jednoduchou úpravu *JSP* stránky, kdy byl odstraněn tabulkový l*ayout*[27] stránky.



Obrázek 5.3: Přihlašovací obrazovka

5.5.1 HTML

Byl přidán obalující `<div class="login">`, aby došlo k odstranění tabulky, jinak je kód převzatý ze staré verze.

```
<div class="login">
 <h1 class="formTitle"> Logon </h1>
 <form name="logonForm" method="post" action="/DCI/logon.do">
...
</div>
```

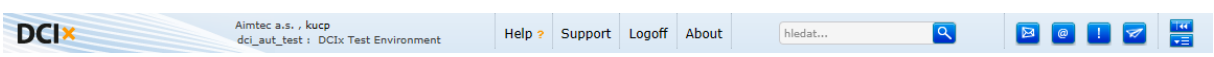
5.5.2 CSS

Byly použity vlastnosti z *CSS3* zaoblené rohy a barevný přechod (viz 5.8.2). Kolem obalujícího elementu je použita vlastnost, která definuje vržený stín.

```
box-shadow: 3px 3px 3px #EEEEEE;
```

5.6 Horní menu

Horní menu bylo na implementaci snadné. Nebylo třeba použít, ani změnit žádnou *Java* třídu, stačilo vytvořit novou *JSP* stránku a upravit vhodně kaskádový styl.



Obrázek 5.4: Nové horní menu

27 *Layout* – grafické rozvržení

5.6.1 HTML

Šablona pro horní menu má následující strukturu

```
<div class='topMenu'>
 <div class='logo'> Logo aplikace </div>
 <div class='userInfo environment'> Informace o uživateli  </div>
 <div class='menuTopLinks'> Odkazy na podporu atd. </div>
 <div class='searchTop'> Hledání </div>
 <div class='eventsTop'> Události – email, zpráva, úkol </div>
 <div class='hideButtons'> Tlačítka - skrytí menu, minimilizace oken  </div>
</div>
```

- `logo` – logo aplikace je vloženo pomocí elementu `<img>`, odkaz na něj je realizován přes element `<a>`,
- `userInfo environment` – obsahuje další `<div>` s potřebnými informacemi o uživateli a prostředí, do kterého je uživatel přihlášen,
- `menuTopLink` – obsahuje seznam, který je řešený pomocí `<ul> <li>`, jednotlivé položky jsou odděleny pomocí obrázku, umístěném v elementu `<img>`,
- `searchTop` – obsahuje formulář `<form>` s možností vyhledávání – vkládá se již hotový skript pro hledání,
- `eventsTop` – jednotlivé ikony jsou umístěny v kontejneru `<span>`, kterému je nastavena pevná velikost a pomocí *CSS* obrázek na pozadí.

5.6.2 CSS

Všechny kontejnery `<div>` mají nastavenu pozici `float: left`, která zajistí, že budou zobrazeny vedle sebe. Mezi nimi jsou nastaveny mezery pomocí vlastností `padding` a `margin`. Po najetí kurzorem myši na ikonu v pravé části, se ikona změní a indikuje tak, že se uživatel nachází nad ní. Umožňuje to *CSS* vlastnost `hover`.

```
.generalIcon {
      background: url(../img/generalMessage.png);
      background-repeat:no-repeat;
}

.generalIcon:hover {
      background: url(../img/generalMessageHover.png);
      background-repeat:no-repeat;
}
```

Vertikální zarovnání ikon v `<div class='eventsTop'>` je řešeno tímto nastavením, pro kontejner `<span>` s ikonou.

```
.messageIconWrapper {
      display: table-cell;
      vertical-align: middle;
}
```

Aby při změně velikosti okna prohlížeče nedošlo k překrývání jednotlivých částí, má `<div class='topMenu'>` nastavenou minimální šířku dle rozlišení obrazovky. To se děje v *JavaScriptové* funkci *onDomReady*[28], kdy po zjištění rozlišení obrazovky, je tato hodnota nastavena vlastnosti *min-width,* která určuje minimální šířku elementu. Pak nedojde k překrývání elementů při změně velikost okna prohlížeče, ale zobrazí se scrollbar pro posunutí stránky.

5.6.3 JavaScript

Tlačítko pro práci s menu umožňuje skrytí, nebo opětovné zobrazení menu.

Změna levého menu

- **Zjištění funkce tlačítka a jeho ikony –** v daném elementu je zobrazeno buď tlačítko pro skrytí, nebo zobrazení menu. O tom rozhoduje třída, kterou má element nastavenou. Pokud má nastavenou třídu *showMenu,* je zobrazena ikona pro zobrazení menu. Pokud má nastavenou třídu *hideMenu*, je zobrazena ikona pro skrytí menu. Funkce pro skrytí/zobrazení menu se volá dle nastavené třídy.
- **Skrytí levého menu –** Pomocí *JQuery* je spuštěna animace, které levé menu posune o jeho šířku doleva. To udělá tak, že mu nastaví zápornou vlastnost *margin-right*. Dále dojde k posunutí minimalizovaných oken o šířku menu doleva.
- **Zobrazení levého menu** – Je nastavena vlastnost *margin-left* na hodnotu 0 pro menu a minimalizovaná okna jsou posunuta doprava.

Pod tlačítkem pro práci s menu se nachází tlačítko pro minimalizaci všech otevřených oken.

Minimalizace všech otevřených oken

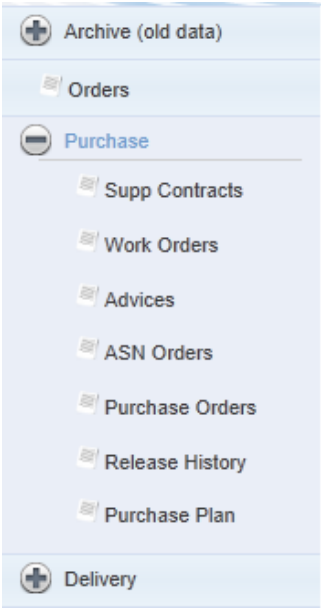
- iteruje přes všechna otevřená okna na ploše,
- uloží informace o pozici a rozměrech okna do atributů,
- posune všechna již minimalizovaná okna doprava,
- otevřená okna minimalizuje a přesune do lišty minimalizovaných oken,
- nastaví všem oknům *z-index*[29] na 0, aby se přes ně dané minimalizované okno zobrazilo správně,
- skryje formuláře v oknech,
- uloží nové informace o velikosti a pozici oken do databáze.

5.7 Levé menu

Jako základ pro levé menu byla využita současná funkcionalita, co se týče načítání položek menu z databáze a skriptu, který využívá utilitu *Treeview* od společnosti Yahoo. Došlo ke změně *CSS* stylu a k přidání nových funkcí. Naopak byla odstraněna sekce *Oblíbené*, kterou v nové verzi

28 *OnDomReady* – funkce je zavolána, když je *DOM* načten
29 *Z-index* – slouží k určení pořadí elementů na ose *z*, bez ohledu na jejich pořadí v *DOM*

převzala plocha. Všechny položky menu mají svojí ikonu. Položka v menu je buď rozbalovací a obsahuje další položky, jako *Purchase* (má u sebe ikonu plus), nebo naopak sbalí odpovídající část menu (má u sebe ikonu mínus). Nebo je odkazem na okno s formulářem a daty, pak má vedle sebe ikonu jako *Work Orders*.



Obrázek 5.5: Nové levé menu

5.7.1 HTML

Postup zobrazení menu je vysvětlen na kódu pro první dvě položky menu. *Archive (old data)* a položce *Orders Obrázek 5.5.*

```
<div class='leftMenu'>

<ul class='treeview-dci treeview' id='menuTreeview'>

  <!-- Položka Archive (old data) -->
  <li class="menuItemLevel2 expandable">
   <div class="hitarea menuItemLevel2-hitarea expandable-hitarea"/>
    <span class="" _dci_openWindow="false" enabled="true" menuKey="archive"
     expanded="false">
     Archive (old data)
    </span>

  <ul style="display: none;">

   <!-- Orders -->
   <li class=menuItemLevel2 context-menu-one menuItemLevel2 dragFromMenu
    openIframeFromMenu ui-draggable">
    <img class="menuLogo" alt="logo" src="img/menu/default_icon_small.png"
     border="0" _dci_srcBig="img/menu/default_icon.png"/>
    <span title="/changeMenu.do?id=420">
    <span _dci_openWindow="false" enabled="true" menuKey="universalOrders"
```

```
     expanded="false">
     Orders
    </span>
   <span title="" class="favorite" _dci_menuKey="universalOrders"
    _dci_menuId="420"/>
 </ul>
</div>
```

- Položka `Archive (old data)` je rozklikávací, má nastavenu třídu `expandable`. Dle této třídy se volí ikona před ní (plus nebo mínus), která je umístěna v elementu `<div>`, který má nastavený obrázek na pozadí pomocí vlastnosti `background-image`
  - Element `<ul>` s nastavením `display: none`, se nezobrazuje. Tyto nezobrazené položky představují položky menu, které se zobrazí po jeho rozbalení.
- Položka `Orders` slouží k zobrazení obrazovky s objednávkami.
  - Má nastavené třídy
    - `contex-menu-one` – je možné přidat ji na plochu pro kliknutí pravým tlačítkem,
    - `dragFromMenu` – je možné přetáhnout ji z menu na plochu,
    - `openWindowFromMenu` – je možné otevřít nové okno přímo z menu,
    - `ui-draggable` – nastaveno *JQuery* pro *draggable.*
  - Ikona před ní je načtena z databáze, pokud není v databázi záznam o ikoně k této položce, použije se defaultní. Tak jako v tomto případě.
  - Atributy
    - `dci_openWindow` – určuje, zda je položka otevírána v novém okně pomocí, *JavaScriptu*, nebo zda má dojít k načtení do elementu na stránce
    - `dci_menuId` – *id*, které má položka menu v databázi.

5.7.2 JavaScript

Kontextové menu

Po kliknutí pravým tlačítkem na položku menu, která není rozklikávací, se zobrazí menu s možností přidání ikonu na plochu. Menu je vytvořené pomocí rozšíření *JQuery context menu*. [JqCon]

Obrázek 5.6: Kontextové menu

```
$.contextMenu({
                        selector : '.context-menu-one',
                        callback : function(key, options) {
                              // telo funkce

                                    },

                              // nastaveni menu
                              items : {
                                    "add" : {
                                          name : "Add to desktop",
                                          icon : "add-to-desktop"
                                    }

                              }

                        });
```

- `selector` – určuje element, na kterém se má menu pro kliknutí zobrazit
- `callback` – funkce, která se volá po zvolení položky menu, parametr `key` obsahuje název `name` z části `items`, na který bylo v menu kliknuto
  - Načte informace z atributů položky, na kterou bylo kliknuto, důležitá je hlavně položka *menu id* viz další krok.
  - *Ajaxem* se dotáže, zda se položka s daným *menu id* již nachází na ploše. Je důležité nastavit při volání *Ajaxu* parametr `async: false`, to znamená, že skript se nebude provádět dále, ale bude se čekat na odpověď ve formátu *JSON*. Pokud se položka již nachází na ploše, zobrazí dialog o duplicitě, pokud ne viz další bod.
  - Vytvoří novou ikonu a přidá ji do levého horního rohu plochy.
  - Uloží informace o nové ikoně do databáze.
- `items` – položky které se zobrazí po rozbalení menu

Pro *Internet Explorer* musí být třída `contex-menu-one` zapsána jako první, jinak zobrazení menu nefunguje. Nastavení ikony v kontextovém menu se provede v souboru *jquery.ContextMenu.css*

```
.context-menu-item.icon-add-to-desktop
{
  background-image: url(../../../img/desktop-icon-16.png);
}
```

Přetažení z položky na plochu

Pro tuto funkci se používá komponenta `draggable`, kterou obsahuje *JQuery*.

```
$('.dragFromMenu').draggable({
                              containment : 'body',
                              opacity : 0.50,
                              appendTo : 'body',

            // ----- create new element (desktop icon) after start dragging
```

```
                            helper : dragFromMenuHelper,

          // ----- add element (desktop icon) to desktop after stop dragging
                            stop : addFavorite
                });
```

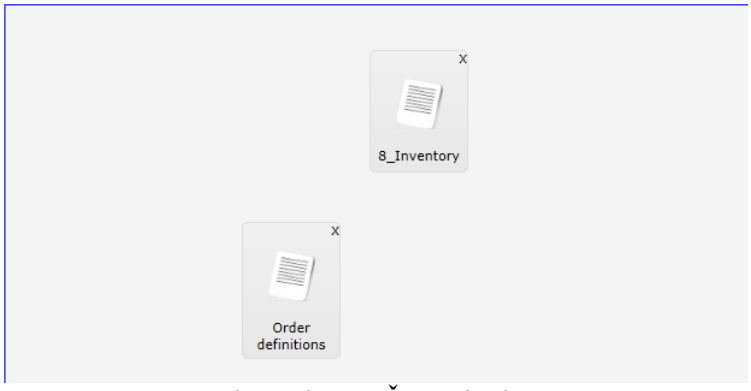
- `containment` – určuje, kam bude možné položku přetáhnout, jelikož je tažená z levého menu na plochu, není možné ji omezit třeba jen plochou, nebo levým menu, proto je nastaveno `body`
- `opacity` – určuje, že tažená položka bude mít nastavenou průhlednost, kde 1 značí neprůhlednost a 0 úplnou průhlednost
- `appendTo` – k jakému elementu bude položka přidána v průběhu tažení
- `helper : dragFromMenuHelper`
  - Funkce, která se stará o vykreslení položky v průběhu tažení.
    - Načte z dané položky menu potřebná data pro novou ikonu (text, odkaz na obrázek, atd.), která bude přidána na plochu.
    - Pokud je text položky moc dlouhý, přidá mezeru.
    - Vytvoří novou ikon zavoláním funkce `createDesktopIcon`, která vrátí správný typ ikony, pokud se jedná o ikonu, která se otevírá v novém okně *JavaScriptem*, nebo o ikonu, který načítá data do elementu na stránce.
- `stop : addFavorite`
  - Je zavolána v okamžiku, kdy je ukončeno tažení myší.
  - Nejprve zkontroluje, zda se tato položka na ploše již nenachází. To dělá voláním *Ajaxu*, je důležité nastavit při volání parametr `async: false,` to znamená, že skript se nebude provádět dále, ale bude se čekat na odpověď. Vrátí se odpověď ve formátu *JSON*, s informací jestli je ikona již na ploše.
  - Pokud ne, přidá taženou ikonu na plochu a pomocí *Ajaxu* ji uloží do databáze.
  - Pokud se ikona na ploše již nachází, zobrazí dialog o duplicitě ikon.

5.7.3 Databáze

Položky menu jsou uloženy v databázi v tabulkách *Menu*, *Menu_Items* a *Menu_Items_For_Role*, všechny potřebné třídy již byly hotové a pouze byly znovu použity, podobně jako v původním menu. Tabulka *Menu_Items* dokonce už obsahovala i položky pro ikony. Pro ikonu malou, která je umístěna v menu, i pro velkou, která je umístěna na ploše.

5.8 Plocha

Plocha obsahuje ikony, které je možné na ní libovolně posouvat, nebo mazat. V případě kliknutí na ikonu na ploše dojde k otevření obrazovky s daty.



Obrázek 5.7: Část plochy

5.8.1 HTML

Struktura plochy je jednoduchá, jedná se o dva `<div>` kontejnery, ve kterých jsou umístěny ikony.

```
<div id="desktop" style="width: 1690px; height: 772px;">
 <div id="icons">
   <div class="desktopIcon divDefaultSyle ui-draggable" style="left: 236px;
    top: 474px;">
   <div class="desktopIcon divDefaultSyle ui-draggable" style="left: 136px;
    top: 574px;">
 </div>
</div>
```

Ikony na ploše se liší dle toho, jestli se otevírají v novém okně pomocí *JavaScriptu*, nebo zda jsou načítány přímo do elementu.

- Ikona otevírající nové okno pomocí *JavaScriptu* (struktura je podrobnější popsána u dalšího typu ikony).

```
<div class="desktopIcon divDefaultSyle ui-draggable" style="left: 413px;
 top: 316px;">
 <div class="removeIcon">X</div>
 <a href="JavaScript:OpenActionWindow(childEditWindow,...");">
 <div class="iconText">Work Orders</div>
</div>
```

- Ikona otevírající nové okno do elementu na stránce.

```
<div class="desktopIcon divDefaultSyle ui-draggable" style="left: 413px;
 top: 316px;">
 <div class="removeIcon">X</div>
 <img title="http://localhost:8080/DCI//changeMenu.do?id=404"
  class="iconImagenewIframe"alt="logo"src="img/menu/default_icon.png"
  border="0" _dci_menuid="404" srcBig="img/menu/default_icon.png"/>
 <div class="iconText">
```

- `<div class="desktopIcon...">` v atributu style, je nastavena pozice, která určuje pozici ikony na ploše. Třída `divDefaultStyle` slouží pro zajištění jednotného stylu ikon. Třída `ui-draggable` je přidána *JQuery* a umožňuje tažení ikony na ploše.
- V `<div class="removeIcon">` je umístěn křížek, po kliknutí na něj dojde k zobrazení dialogu o vymazání ikony z plochy, v případě potvrzení bude ikona smazána jak z plochy, tak z databáze.
- Element `<img>` obsahuje všechny důležité informace pro vytvoření nového okna, nebo práci s ikonou v databázi.
- `<div class="iconText">` obsahuje text, který je zobrazený na ikoně.

5.8.2 CSS

Kontejner `icons` má nastavenou pozici jako relativní, takže souřadnice ikon na ploše nejsou počítány v rámci celého okna prohlížeče, ale pouze tohoto `divu`. Počítají se od levého horního rohu `<div id="icons">`.

Jak je vidět na Obrázek 5.7 ikony mají zaoblené rohy a navíc je na jejich pozadí použitý barevný přechod. Tyto možnosti nabízí *CSS3*. Pro staré prohlížeče je nastaveno jednobarevné pozadí a pak dle prohlížeče barevný přechod. Pro vygenerování zápisů přechodů pro všechny prohlížeče je možné použít nástroj http://www.colorzilla.com/gradient-editor/.

```
/* Old browsers */
background: rgb(240, 240, 240);

      /* IE9 SVG, needs conditional override of 'filter' to 'none' */
      background:
url(
JodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIgd2lkdGg9IjEwMCUiIGhlaWdodD0iMTAwJSIgdm
lld0JveD0iMCAwIDEgMSIgcHJlc2VydmVBc3BlY3RSYXRpbz0ibm9uZSI+CiAgPGxpbmVhckdy
YWRpZW50IGlkPSJncmFkLXVjZ2ctZ2VuZXJhdGVkIiBncmFkaWVudFVuaXRzPSJ1c2VyU3BhY2VPbl
VzZSIgeDE9IjAlIiB5MT0iMCUiHgyPSIwJSIgeTI9IjEwMCUiPgogICAgPHN0b3Agb2Zmc2V0PS
IwJSIgc3RvcC1jb2xvcj0iI2YwZjBmMCIgc3RvcC1vcGFjaXR5PSIxIi8+CiAgICA8c3RvcCBm
ZzZXQ9IjEwMCUiHN0b3AtY29sb3I9IiNlOWU4ZTgiHN0b3Atb3BhY2l0eT0iMSIvPgogIDwvbG
luZWFyR3JhZGllbnQ+CiAgPHJlY3QgeD0iMCIgeT0iMCIgd2lkdGg9IjEiIGhlaWdodD0iMSIgZm
lsbD0idXJsKCNncmFkLXVjZ2ctZ2VuZXJhdGVkKSIgLz4KPC9zdmc+);

      /* FF3.6+ */
      background: -moz-linear-gradient(top, rgba(240, 240, 240, 1) 0%,
            rgba(233, 232, 232, 1) 100% );

      /* Chrome,Safari4+ */
      background: -webkit-gradient(linear, left top, left bottom,
                  color-stop(0%, rgba(240, 240, 240, 1) ),
                  color-stop(100%, rgba(233, 232, 232, 1)));

      /* Chrome10+,Safari5.1+ */
      background: -webkit-linear-gradient(top, rgba(240, 240, 240, 1) 0%,
            rgba(233, 232, 232, 1) 100% );

      /* Opera 11.10+ */
      background: -o-linear-gradient(top, rgba(240, 240, 240, 1) 0%,
            rgba(233, 232, 232, 1) 100% );

     /* IE10+ */
```

```
      background: -ms-linear-gradient(top, rgba(240, 240, 240, 1) 0%,
            rgba(233, 232, 232, 1) 100% );

     /* W3C */
      background: linear-gradient(to bottom, rgba(240, 240, 240, 1) 0%,
            rgba(233, 232, 232, 1) 100% ); * W3C */

/* IE6-8 */
filter: progid : DXImageTransform.Microsoft.gradient
      (startColorstr ='#f0f0f0', endColorstr = '#e9e8e8', GradientType = 0 );

      border-radius: 5px 5px 5px 5px;
      -ms-border-radius: 5px;
```

5.8.3 Databáze

Pro uložení ikon z plochy byla použita již existující tabulka *Favorite_Menu_Item_For_Role*, která v původní verzi sloužila k uchovávání informací o oblíbených položkách. Jelikož tato volba z menu byla odstraněna a místo ní je připravena plocha, přímo se nabízelo tuto tabulku použít. Bylo tak možné využít řady hotových tříd.

Alfa Framework

- `FavoriteMenuItem` – obsahuje potřebné informace pro ikonu na ploše, je obrazem dat, jednoho řádku tabulky,

```
public class FavoriteMenuItem extends AlfaObject<String> implements Mutable {
      private Menu menu;
      private String menuKey;
      private Integer positionX;
      private Integer positionY;
```

- `FavoriteMenuItemTable` – provádí načítání dat z databáze do objektu a z objektu do databáze, metodami `fillData` a `fillColumnValues`,

```
public void fillData(ResultSet pResultSet, AlfaObject<?> pAlfa){

 FavoriteMenuItem item = (FavoriteMenuItem) pAlfa;

 item.setMenuKey(pResultSet.getString(FavoriteMenuItemMapping.MENU_KEY.
                 GetName()));

 item.setPositionX(Integer.valueOf(pResultSet.
                   getInt(FavoriteMenuItemMapping.POSITION_X.getName())));

 item.setPositionY(Integer.valueOf(pResultSet.
                  getInt(FavoriteMenuItemMapping.POSITION_Y.getName())));
}
```

- `FavoriteMenuItemMapping` – určuje názvy sloupců v tabulce, primární klíč a název tabulky v databázi,

```
public class FavoriteMenuItemMapping extends ColumnMapping {
      public static final ColumnDescriptor MENU_KEY =
      new ColumnDescriptor("Menu_Key", ColumnType.STRING);
```

```
      public static final ColumnDescriptor USER_ID =
      new ColumnDescriptor("User_ID", ColumnType.ID);

      public static final ColumnDescriptor POSITION_X =
      new ColumnDescriptor("Position_X", ColumnType.NUMBER);

      public static final ColumnDescriptor POSITION_Y =
      new ColumnDescriptor("Position_Y", ColumnType.NUMBER);
```

- `FavoriteMenuItemCollection` – přidává, maže nebo aktualizuje záznamy v kolekci.

5.8.4 Struts

Pro novou funkcionalitu byla upravena akce `FavoriteMenuChangeAction`. Na tuto akci je navázána třída `FavoriteMenuChangeForm`, jejíž atributy odpovídají položkám, posílaným v *URL*, při přidávání, mazání, nebo aktualizaci ikony.

- `FavoriteMenuChangeForm` – pro potřebu kontroly, zda se již ikona nachází na ploše, byl přidán atribut `added`

```
public class FavoriteMenuChangeForm extends AlfaActionForm<Object> {
      private String menuId;
      private String positionX;
      private String positionY;
      private String added;
```

- `FavoriteMenuChangeAction` – metoda získá data, která byla poslána *HTTP* požadavkem pomocí `FavoriteMenuChangeForm` a na základě požadované akce pracuje dále
  - přidání ikony – přidá ikonu do databáze i do kolekce
  - smazání ikony – vymaže ikonu z databáze i z kolekce
  - aktualizace ikony – změní pozici ikony na ploše a uloží nové hodnoty do databáze
  - kontrola, zda je ikona na ploše – zjistí, zda se ikona již nachází na ploše nebo ne

Kontrola unikátnosti ikony

- `struts-config.xml` – nutno nastavit nové přesměrování na odpovídající *JSP*

```
<forwardname="checkDesktopIcon"
 path="/template/getFavoriteMenuItemIsAdded.jsp" />
```

- `FavoriteMenuChangeAction` – kontrola unikátnosti ikony a přesměrování na nově přidané *JSP*

```
if (DciActionName.CHECK_FAVORITE.equals(xForm.getAction())) {
                  if (favorites.notContainsKey(favoriteKey)) {
                        xForm.setAdded("false");
                  }
                  else {
                        xForm.setAdded("true");
                  }
```

```
                return findForward(mapping,
                                   DciForwardKey.CHECK_DESKTOP_ICON);
            }
```

- `GetFavoriteMenuItemAdded.jsp` – slouží k zobrazení dat, poslaných třídou `FavoriteMenuChangeAction`

```
<bean:define id="added" name="favoriteForm" property="added"></bean:define>
{
 "added" : ${added}
}
```

- Ve skriptu se na základě zjištěné hodnoty `added` zobrazí hláška, že se ikona již nachází na ploše, nebo dojde k jejímu přidání na plochu (viz 5.8.6).

5.8.5 JSP

Ikony z databáze jsou na plochu vykresleny pomocí *JSP* stránky. Třída *DciWorkspace* obsahuje metodu `getFavorites()`, která vrátí seznam všech ikon, uložených v databázi. V *JSP* se pak pomocí tagu `<logic:iterate>` ikony zobrazí na plochu. Ukázka *startNewEntry.jsp*.

```
<logic:iterate id="favoriteMenu"
 name="wrk"property="currentMenu.favorites.list" type="FavoriteMenuItem">

 <bean:define id="menu" name="favoriteMenu" property="menu" type="Menu" />

 <div class="desktopIcon divDefaultSyle ui-draggable" style="left:
 ${favoriteMenu.positionX}px; top: ${favoriteMenu.positionY}px;">

 <div class="removeIcon">X</div>

 <!-- Open in current window  -->
 <logic:notEqual name="menu" property="menuItem.openInNewWindow"
  value="true">

  <!-- Icon is set -->
  <logic:notEmpty name="menu" property="iconExpanded">
   <img src="${menu.iconExpanded}" srcBig="${menu.iconExpanded}"
    border="0" title="http://localhost:8080/DCI/${menu.actionInWithType}"
    alt="logo" _dci_menuid="${menu.id}" class="iconImage newIframe" />
 </logic:notEmpty>
```

- `<logic:iterate>` definuje kolekci, ze které se budou číst data,
- `<bean:define>` definuje, z jaké proměnné a jakého typu se budou jednotlivé ikony vypisovat,
- `{favoriteMenu.positionX}` vypíše atribut `positionX`, zavolá nad třídou *Menu* metodu *getPositionX()*, pro odpovídající položku, stejně tomu děje i pro další atributy,

- `<logic:notEmpty name="menu" property="iconExpanded">` kontroluje, zda je v databázi uložena ikona k dané položce, pokud ne, nastaví defaultní,
- nastaví další atributů ikony, jako jsou třídy a *menu id*.

5.8.6 JavaScript

Uložení změněné pozice ikony na ploše do databáze Ajaxem.

```
// update positon in db
 $.ajax({
      type : 'GET',
      url : 'changeFavoriteMenu.do',
      data : {
            action : 'moveFavorite',
            menuId : menuId,

           // round is because of IE
            positionX : Math.round(off.left),
            positionY : Math.round(off.top)
            }
      });
```

- Ajaxové volání pomocí knihovny *JQuery*
  - `type` – *GET* – typ, jak se budou data serveru posílat,
  - `url` – *URL* adresa na serveru, kam se budou posílat data,
  - `data` – data posílaná serveru, která jsou přeformátována do *Query Stringu*[30],
    - `action` – `moveFavorite` – jaká akce se bude provádět, v tomto případě posun ikony,
    - `menuId` – id menu, podle kterého se bude hledat ikona z plochy v databázi,
    - `positionX` – nová pozice, na které se ikona nachází,
    - `positionY` – nová pozice, na které se ikona nachází.

Kontrola unikátnosti ikony

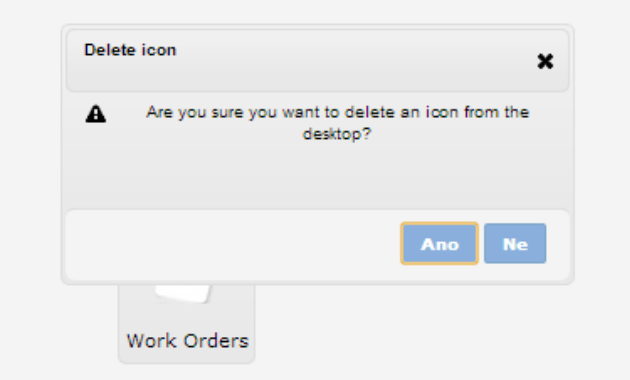
```
// ----- check if item isn't on desktop
 $.ajax({
      dataType: "json",
      url: 'changeFavoriteMenu.do?action=checkFavorite',
      data: {menuId : menuId},
      async : false,
      success: function(data) {
                alreadyAdded = data.added;
            }
      });

      // ----- item isn't on desktop
      if (!alreadyAdded){

            // ----- add icon to db
            addFavoriteToDb(off);
      }
```

30 *QueryString* – část *URL*, který obsahuje data

- Ajaxové volání pomocí knihovny *JQuery*
  - `dataType` – očekávají se data ve formátu *JSON,*
  - `url` – *URL* adresa na serveru, kam se budou posílat data,
  - `data` – data posílaná serveru, která jsou přeformátována do *QueryStringu,*
    - `menuId` – id menu, podle kterého se bude hledat ikona z plochy v databázi,
  - `async` – `false` – bude se čekat na odpověď, protože na jejím základě se rozhodne o tom, zda je možné ikonu přidat na plochu,
  - `success` – funkce která se provede, když přijdou úspěšně data.

Dialogy

Při mazání ikony z plochy, při zavírání otevřeného okna, či zobrazení zprávy o duplicitě ikon na ploše, se používají dialogy pro potvrzení volby uživatele.



Obrázek 5.8: Dialog pro potvrzení smazání ikony z plochy

Dialog je `<div>`, který je připravený na stránce a má nastaveno, že se nemá zobrazovat. Při kliknutí na křížek na ikoně, se dialog vyvolá pomocí *JQuery*.

```
$(document).on('click', '.removeIcon', function() {
            // set dialog visible because of IE
            $('#dialogRemoveIcon').css('visibility', 'visible');
            $('#dialogRemoveIcon').dialog('open');
 }
$('#dialogRemoveIcon').dialog({
            autoOpen : false,
            show : 'blind',
            hide : 'fade',
            resizable : false,
            buttons : {

                  // remove icon from desktop and db
                  Ano : function() {

                        ...
                  },

                  // close dialog
                  Ne : function() {
                        ...
                  }
```

- Otevření dialogu pomocí funkce dialog
  - `autoOpen – false` – dialog se nezobrazí při načtení stránky,
  - `show – blind` – efekt pro zavření dialogu,
  - `hide – fade` – efekt pro zobrazení dialogu,
  - `resizable – false` – dialogu není možné změnit velikost,
  - `buttons` – tlačítka, která dialog obsahuje.

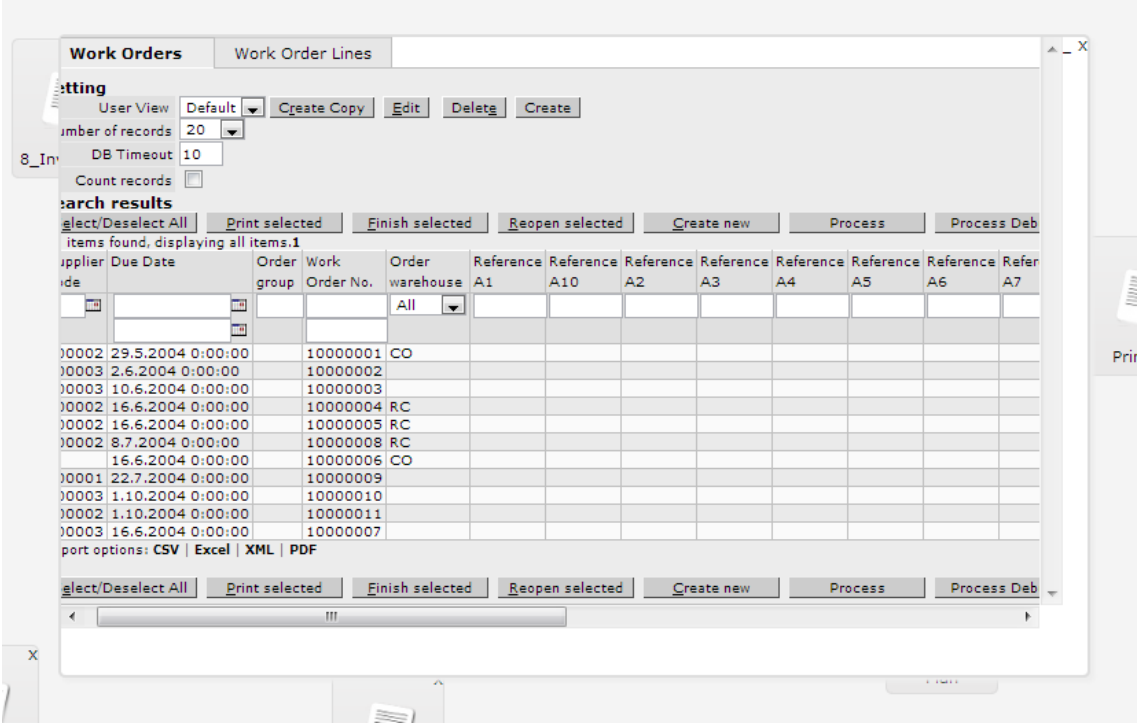
Aplikace je v české i anglické verzi, proto bylo třeba dialogy internacionalizovat, aby se zobrazily ve správném jazyce. Řešení pro internacionalizaci nabízí framework *Struts.*

```
<div id='dialogRemoveIcon' title='<bean:message
 key="desktop.dialog.removeIcon.title"/>'>
 <p>
  <span class='ui-icon ui-icon-alert'
   style='float: left; margin: 0 7px 20px 0;'> </span>
  <bean:message key="desktop.dialog.removeIcon.text"/>
 </p>
</div>
```

- Vypsání textu v odpovídajícím jazyce se provede tagem `<bean:message>`, který najde odpovídající hodnotu pro klíč v souboru *Application_Resources.properties,* který obsahuje dvojici hodnota, klíč pro odpovídající položku. Soubor pro českou verzi má název *Application_Resources_cs.properties,* ostatní jazykové verze se liší zkratkou jazyka za *Resources.* Ukázka z *Application_Resources_cs.properties.*

```
desktop.dialog.removeIcon.text = Opravdu chcete smazat ikonu z plochy?
```

5.9 Práce s okny



Obrázek 5.9: Okno s daty

Aplikace umožňuje zobrazení více oken najednou. Okna je možné zvětšovat, zmenšovat, posouvat a nebo minimalizovat. Je to stěžejní a také nejsložitější část celé aplikace.

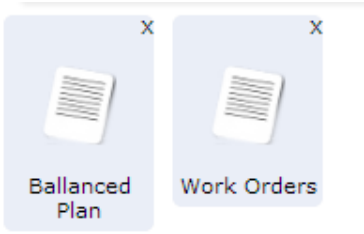
5.9.1 HTML

Okna v aplikaci mají následující strukturu.

```
<div id='activeWindow1'>
 <div class='closeWindow'> ... </div>
 <div class='minimizeWindow'> ... </div>
 <div id='activeWindow1Content'> ... </div>
</div>
```

- `closeWindow` – obsahuje křížek pro zavření okna, po kliknutí na něj se zobrazí dialog pro potvrzení volby,
- `minimizeWindow` – obsahuje ikonu pro minimalizaci okna, po kliknutí se okno minimalizuje do spodní části obrazovky,
- `activeWindowContent` – obsahuje formulář s daty.

Minimalizovaná okna



Obrázek 5.10:
Minimalizovaná okna

```
<div id='activeWindow1'>
 <div class='minimizedWindow'>...</div>
 <div class='removeIconActive'>...</div>
 <img class='activeWindowImage'>...</img>
 <div class='iconTextActiveWindow'>...</div>
</div>
```

- `minimizedWindow` – element, který je umístěn přes minimalizované okno,
- `removeIconActive` – umožňuje odstranění minimalizovaného okna,
- `activeWindowImage` – obsahuje obrázek minimalizovaného okna,
- `iconTextActiveWindow` – obsahuje text k danému oknu.

Otevřená okna se přidávají k ploše, tedy k elementu `<div id='desktop'>`, stejně tak se k němu přidávají okna, která se maximalizují. Minimalizovaná okna se přidávají k elementu `<div id='minimizedWindows'>`

5.9.2 CSS

Okno má nastavené zaoblené rohy a jeho pozice se počítá od levého horního rohu prohlížeče. V závislosti na tom, zda se daným oknem právě pracuje, má nastavený *z-index*, který určuje v jakém pořadí se elementy zobrazí. Okno, se kterým se právě pracuje, má nastavený *z-index* na hodnotu 100, všechny ostatní ho pak mají nastavený na 0.

Element `activeWindowContent`, který obsahuje formulář s daty, má nastavenou vlastnost `overflow:scroll`, která způsobí, že při změně šířky okna nedojde k „přetečení“ obsahu, ale objeví se scrollbar pro posunutí.

Při najetí kurzoru myši nad element pro minimalizaci nebo zavření okna, se změní kurzor myši.

```
.closeWindow {
      cursor: pointer;
}
```

5.9.3 JavaScript

Otevření nového okna

- Vytvoří se struktura elementů, jak je popsáno výše a okno je přidáno k ploše.
- Dojde k uložení okna do databáze, pomocí *Ajaxového* volání, které je nutné nastavit jako synchronní, aby se čekalo na odpověď. Zpátky přijde *JSON* odpověď s *id* okna v tabulce, což umožní jeho jednoznačnou identifikaci.
- Funkcí `load` je načten formulář do elementu s třídou `activeWindowContent`, když jsou všechna data načtená, pokračuje se dále.
- Novému oknu jsou nastaveny možnosti
  - *resizable* – umožňuje zvětšování a zmenšování okna tažením kurzoru myši. Nastavuje minimální výšku, na kterou je možné změnit velikost okna, dle výšky formuláře, který okno obsahuje. Při začátku měnění velikost je ve funkci *start* nastaven oknu *z-index* na hodnotu 100 a všem ostatním je nastaven na 0 (viz 5.9.2). Díky tomu je právě měněné okno vždy navrchu. V události *stop*, která se provede při ukončení změny velikosti, volá *Ajax*, aby upravil velikost okna uloženou v databázi.
  - *draggable* – umožňuje změnu pozice okna na ploše tažením kurzoru myši. Tuto možnost nepovolí pro element s třídou `activeWindowContent`, což je vnitřek okna s daty, nad kterým není možné posun okna provádět. Událost *stop*, která se volá při ukončení tažení objektu, volá *Ajax*, aby se upravila pozice okna v databázi.

Při minimalizaci okna se používá následující postup:

- do atributů se uloží velikost a pozice okna, které se bude minimalizovat,
- okno je přesunuto v *DOM* struktuře z plochy do minimalizovaných oken,
- v cyklu se iteruje přes všechna již minimalizovaná okna
  - je vypočtena pozice pro okno, které bude minimalizováno, to bude přidáno úplně vlevo,
  - je nastaven *z-index* na 0, protože na okno bylo kliknuto a má nastavený z-index na 100, takže by překrývalo minimalizované okno,
  - všechna minimalizovaná okna jsou posunuta doprava,
- přes zmenšené okno je přidán <*div*> symbolizující minimalizované okno,
- jsou skryty formuláře v minimalizovaném okně,
- jsou uloženy změny pozic oken do databáze.

Načtení oken

Při zobrazení plochy po přihlášení uživatele je nutno načíst z databáze uložená okna a to jak ta minimalizovaná, tak ta otevřená. To se děje v šabloně s*tartEntryNew.jsp*, která se stará i o vykreslení ikon na ploše.

Zobrazení aktivní oken

```
<logic:iterate id="window" name="wrk" property="windows.list"
 type="ActiveWindowsItem">

 <logic:notEmpty name="window">
  <logic:equal name="window" property="isMinimized" value="0">

   <div id="activeWindow${window.windowId}" _dci_windowid="
    {window.windowId}"  class="activeWindowDiv"
    style="left: ${window.positionX}px; top: ${window.positionY}px;
    width: ${window.width}px; height: ${window.height}px">

   <div class="closeWindow">X</div>
   <div class="minimizeWindow">_</div>

   <logic:notEmpty name="${wrk.menuItemMap[window.menuKey].iconExpanded}">

    <div id="activeWindowContent${window.windowId}"
      class="activeWindowContent"
      _dci_menuid="{wrk.menuMap[window.menuKey].id}"
      src="http://${pageContext.request.serverName}:
      ${pageContext.request.serverPort}${pageContext.request.contextPath}
      ${wrk.menuMap[window.menuKey].actionInWithType}"
      title="${window.menuKey}"
      srcbig="${wrk.menuItemMap[window.menuKey].iconExpanded}">
     </div>

   </logic:notEmpty>
  </logic:equal>
 </logic:notEmpty>
</logic:iterate>
```

- iteruje přes všechna aktivní okna, která nejsou minimalizovaná,
- nastaví do atributu *id* okna z databáze a pozici a velikost okna,
- přidá elementy pro minimalizaci a zavření okna,
- kontroluje, zda je pro dané okno uložena informace o ikoně,
- nastaví, jaká akce se má při načtení okna provést.

Zobrazení minimalizovaných oken

```
<logic:iterate id="window" name="wrk" property="windows.list"
 type="ActiveWindowsItem">

 <logic:notEmpty name="window">
  <logic:equal name="window" property="isMinimized" value="1">
```

```
   <div id="activeWindow${window.windowId}"
    _dci_windowid="{window.windowId}"_dci_positionx="${window.positionX}"
    _dci_positiony="${window.positionY}"_dci_windowwidth="${window.width}"
    _dci_windowheight ="${window.height}" class="activeWindowDiv"
    style="left:${window.minimizedPositionX}px;
    top: ${window.minimizedPositionY}px;
    width: 60px; height: 80px; opacity: 0.5;">

   <div class="closeWindow">X</div>
   <div class="minimizeWindow">_</div>

   <c:if test="${not empty wrk.menuItemMap[window.menuKey].iconExpanded}">

    <div id="activeWindowContent${window.windowId}"
     class="activeWindowContent" style="visibility: hidden;"
     _dci_menuid="${wrk.menuMap[window.menuKey].id}"
     src="http://${pageContext.request.serverName}:
     ${pageContext.request.serverPort}${pageContext.request.contextPath}
     ${wrk.menuMap[window.menuKey].actionInWithType}"
     title="${wrk.captionMap[window.menuKey]}"
     srcbig="${wrk.menuItemMap[window.menuKey].iconExpanded}">
    </div>

   </c:if>

  <div class="minimizedWindow" style="position: absolute;
   left: ${window.minimizedPositionX}px;
   top: ${window.minimizedPositionY}px;">
   <div class="removeIconActive">X</div>
   <img class="activeWindowImage" src="img/menu/default_icon.png" border="0">
   <div class="iconTextActiveWindow">${wrk.captionMap[window.menuKey]}</div>
  </div>

  </logic:equal>
 </logic:notEmpty>
</logic:iterate>
```

- iteruje přes všechna aktivní okna, která jsou minimalizovaná,
- nastaví do atributů *id* okna z databáze, pozici a velikost okna,
- nastaví velikost okna defaultní pro minimalizované okna,
- přidá elementy pro minimalizaci a zavření okna,
- kontroluje, zda je pro dané okno uložena informace o ikoně,
- nastaví, jaká akce se má při načtení okna provést,
- přes okna přidá element minimalizovaného okna a nastaví mu odpovídající pozici,

Takto nastavená okna, která jsou načtená z databáze, je ještě nutné nastavit pro změnu velikost a tažení oken po ploše. Také je nutné načíst do oken formuláře voláním *Ajaxu*. To se děje v *JavaScriptové* funkci *onDomReady.*

```
$(document).ready(function () {

// ----- set opened windows as draggable and resizable
$($('.activeWindowDiv')).each(
function() {
  var windowId = $(this).attr('_dci_windowid');

   // ----- load form to div and change from structure for ajax
  $('#activeWindowContent' + windowId)
  .load($(this).children('.activeWindowContent').attr('src'),

  function () {
    // ----- set window draggable and resizable
    ...
```

- iteruje přes všechna okna minimalizovaná i otevřená,
- nastaví třídu dle *id* okna,
- načte formulář pomocí *Ajaxu* do připraveného divu,
- nastaví okna pro změnu velikost a změnu pozice pomocí tažení.

5.9.4 Databáze

Pro uložení oken do databáze byla vytvořena nová tabulka *Active_Window_For_Users* s následující strukturou.

ID	User_ID	Menu_Key	Position_X	Position_Y	Width	Height	Is_Minimized	Minimized_Position_X	Minimized_Position_Y
1	4	printers	214	47	1618	719	1	195	830
2	4	workOrders	201	49	1618	435	1	285	830

Obrázek 5.11: Tabulka aktivní oken

- *ID* – jednoznačná identifikace okna,
- *User_ID* – *id* uživatele, kterému se otevře dané okno,
- *Menu_Key* – klíč položky z menu,
- *Position_X* – pozice okna zleva braná od levého horního rohu okna prohlížeče,
- *Position_Y* – pozice okna shora braná od levého horního rohu okna prohlížeče,
- *Width* – šířka okna v pixelech,
- *Height* – výška okna v pixelech,
- *Is_Minimized* – 0 – okno je minimalizované, 1 – okno není minimalizované,
- *Minimized_Position_X* – pozice minimalizovaného okna zleva v pixelech od levého horního rohu okna prohlížeče,
- *Minimized_Position_Y* – pozice minimalizovaného okna shora v pixelech od levého horního rohu okna prohlížeče.

Tabulka má nastavená dvě omezení, která se týkají cizích primárních klíčů *User_ID* a *Menu_Key*.

```
begin
      alter table [dciowner].[Active_Windows_For_User]
            add constraint
[FK_Menu_Active_Windows_For_User__Menu_Key__Menu_Items]
            foreign key([Menu_Key])
            references [dciowner].[Menu_Items]([Menu_Key])
end
```

Obdobně je nastaveno i omezení pro cizí *User_ID*.

Byly vytvořeny následující třídy pro práci s okny v databázi.

- *ActiveWindowsItem* – obsahuje informace z jednoho řádku tabulky *Active_Window_For_Users*, privátní atributy a k nim odpovídající *gettry* a *settry*,

```
      private Integer windowId;
      private Integer userId;
      private String menuKey;
      private Integer positionX;
      private Integer positionY;
      private Integer width;
      private Integer height;
      private Integer isMinimized;
      private Integer minimizedPositionX;
      private Integer minimizedPositionY;
```

- *ActiveWindowMapping* – obsahuje namapování sloupců na konkrétní názvy, název tabulky a primární klíč,

```
public String getPrimaryColumn() {
            return ActiveWindowsMapping.ID.getName();
      }
```

- *ActiveWindowsTable* – stará se o zápis dat z výsledků dotazu do *ActiveWindowsItem* a plní data pro dotaz z tohoto objektu.

```
public void fillColumnValues(AlfaObject<?> alfa, ColumnValueList columns,
 boolean isInsert) {
  ActiveWindowsItem activeWindowsItem = (ActiveWindowsItem) alfa;
  columns.add(ActiveWindowsMapping.USER_ID, activeWindowsItem.getUserId());
  columns.add(ActiveWindowsMapping.MENU_KEY, activeWindowsItem.getMenuKey());
  columns.add(ActiveWindowsMapping.POSITION_X,
              activeWindowsItem.getPositionX());
  columns.add(ActiveWindowsMapping.POSITION_Y,
              activeWindowsItem.getPositionY());
 ...
}
```

```
public void fillColumnValues(AlfaObject<?> alfa, ColumnValueList columns,
 boolean isInsert) {
  ActiveWindowsItem activeWindowsItem = (ActiveWindowsItem) alfa;
  columns.add(ActiveWindowsMapping.USER_ID, activeWindowsItem.getUserId());
  columns.add(ActiveWindowsMapping.MENU_KEY, activeWindowsItem.getMenuKey());
  columns.add(ActiveWindowsMapping.POSITION_X,
              activeWindowsItem.getPositionX());
  columns.add(ActiveWindowsMapping.POSITION_Y,
              activeWindowsItem.getPositionY());
  ...
}
```

5.9.5 Struts

Vytvořena nová akce *ActiveWindowsChangeAction* pro práci s okny

- Vytvořena třída *ActiveWindowsForm*, jejíž atributy odpovídají datům posílaným v *HTTP* požadavku.

```
public class ActiveWindowsForm extends AlfaActionForm<Object> {
      private String windowId;
      private String menuId;
      private String positionX;
...
```

- Cesta ke třídě s formulářem nastavena ve *struts-config.xml*

```
<form-bean
 name="windowsForm"
 type="cz.aimtec.dci.windows.ActiveWindowsForm" />
```

- Nová akce – bylo nutné upravit *stuts.config.xml*

```
<action
      path="/changeActiveWindows"
      type="cz.aimtec.dci.windows.action.ActiveWindowsChangeAction"
      name="windowsForm"
      scope="request">
 <set-property property="authClass"
  value="cz.aimtec.dci.action.DciActionAuthentication"/>
 <forward
      name="success"
      path="/template/page/default_leftMenuFavorites.jsp" />
 <forward
      name="getActiveWindowId"
      path="/template/getActiveWindowId.jsp" />
</action>
```

- Zjistí atributy, které byly poslány v *HTTP* požadavku, na základě hodnoty *action* pracuje dále
  - Create – vytvoří nové okno, dle údajů zaslaných v *HTTP* požadavku a vloží ho do databáze a přidá do kolekce. Z databáze získá unikátní *id* právě vytvořeného okna, přesměruje na *JSP* stránku, která obsahuje *id* právě přidaného okna

- Remove – dle zadaného *id* vymaže okna z databáze i z kolekce všech oken

```
else if (DciActionName.REMOVE_WINDOW.equals(xForm.getAction())) {

            ActiveWindowsItem activeWindowItem = new ActiveWindowsItem(wrk);

            // Find item by id
            activeWindowItem.setId(windowId);
            activeWindowItem.read();

            // DB delete
            activeWindowItem.delete();

            // remove from collection
            collection.remove(activeWindowItem);

            }
```

- Move – dle zadaného *id* upraví pozici a velikost daného okna, pokud bylo okno minimalizováno, nastaví tuto informaci do databáze

5.10 Použití Ajaxu

Bylo nutné vymyslet, jak realizovat načítání dat do jednotlivých oken. Nakonec bylo implementováno řešení, kdy je akce zavolána pomocí *Ajaxu* a obdržená data nejsou vrácena do nové stránky, ale do elementu, který odpovídá otevřenému oknu. Díky tomu je možné mít otevřeno více oken najednou a pracovat s nimi. Framework *Struts* ale nepodporuje použití *Ajaxu*, bylo třeba provést změny pro formuláře. Většina tlačítek pro hledání a vymazání formuláře je realizována v šabloně *DisplayTagTemplate.jsp*, stačilo tak změnit tuto šablonu a změna se promítla na velký počet obrazovek, které obsahují tlačítko pro vyhledávání. Následně bylo nutné vyřešit odesílání hodnot z formulářových prvků.

```
<html:form action="<%=formAction%>" styleId="${formId}"
 onsubmit="return false;">
...
<skin:action type="button" key="button.reset"
 styleClass="formButton" htmlName="resetButton" styleId="resetButton"
 onclick=";cz.aimtec.dci.form.sendData($(this), 'Reset')"/>

<skin:action type="submit" key="button.search" htmlName="searchButton"
 styleId="searchButton" styleClass="formButton"
 onclick=";cz.aimtec.dci.form.sendData($(this),'Save')"/>
```

- Je nutné zakázat odesílání formuláře standardní cestou, to se udělá nastavením události `onsubmit` na `false`. Následně bylo nutné vyřešit odesílání hodnot z formulářových prvků (viz níže).
- Na tlačítku pro hledání je změněna událost `onclick`, která volá funkci, která je popsána níže.

```
formSpace.sendData = function(pThis, pAction) {

var attributes = '';

var window = $(pThis).closest('.activeWindowDiv').attr('id');

var action = $('#'+ window + ' form').attr('action');


// ----- all inputs

$($('#'+ window).find('input[type=text]')).each(function () {

 if ($(this).attr('value') !== 'undefined' && $(this).attr('value') !== false

     && $(this).attr('value') !== '') {

     attributes += '&'+$(this).attr('name') + '=' + $(this).attr('value');
 }
  else {
      attributes += '&'+$(this).attr('name') + '=';
  }
});

...
 // ----- load data to div
 $('#'+ window + ' .activeWindowContent').load(action + '?action=' + pAction
   + attributes, function () {

      // ----- set window height and change min height for resizing, because
                loaded data changed height
      $('#'+ window).css('height', height + 100);
      $('#'+ window).resizable({
                  minHeight: height + 100
      });
  });

};
```

- Zjistí název aktuálně používaného okna.
- Zjistí název akce z atributu formuláře, na zjištěnou adresou se bude posílat *Ajaxový* dotaz o data.
- Projde všechna textová pole ve formuláři a pokud mají zadanou hodnotu, uloží tuto hodnotu do atributu, který se bude posílat na server. Pokud mají nastavenou prázdnou hodnotu, uloží se do atributu prázdný řetězec. Na konci přidá tyto atributy do *URL*, které je odesíláno serveru. To umožní použití filtrování, které formuláře nabízí. Tak se tomu děje i se všemi ostatními formulářovými prvky, což zde není ukázáno.
- Najde nejbližší `div`, který odpovídá elementu v oknu, do kterého se načítají data. Formulář se odešle *Ajaxem* na adresu, která se najde ve skrytém formulářovém poli na stránce a do `divu` se *JQuery* funkcí `load` vrátí načtený výsledek. Počká až přijdou data od serveru a změní výšku okna a minimální výšku, na které lze okno zmenšit dle velikosti přijatých dat.

V dalším kroku byly ručně zkontrolovány obrazovky, které lze spustit z menu a funkčnost odesílání formulářů a dat z nich. Bylo nutno zkontrolovat a vyřešit použití ostatních tlačítek. Byly zkoumány ty, které se neotevírají v novém okně *JavaScriptem*. Těchto obrazovek je více než 100 a v souboru *obrazovky.txt*, ve složce *Navody* jsou popsány problémy, které byly nalezeny, případně co bylo uděláno pro jejich opravu. Na základě tohoto průzkumu byly provedeny další změny. Pro obrazovky, které nepoužívají změněnou šablonu *DisplayTagTemplate.jsp*, bylo nutné rozhodnout o tom, jak je jednoduše změnit, aby bylo možno odesílat data *Ajaxem*. Byly zvažovány dvě možnosti.

- Ruční změna odpovídajících šablon – je nutné projít část šablon ručně a provést v nich jednoduché změny, která spočívají v zakázání odesílání formuláře standardní cestou a ve změně události *onclick* na používaných tlačítkách. Nalezení šablony, které se má upravit se zjistí jednoduše ze zdrojového kódu stránky, protože stačí najít *JSP* šablonu s odpovídajícím akcí. Samotná změna ve zdrojovém kódu už je pak jednoduchá.
- Automatické nastavení *JavaScriptem* ve funkci *OnDomReady* – řešení, které vše nastaví najednou automaticky dostatečně obecným skriptem, který přepíše atributy elementů. Problémem je však to, že je stejně nutné šablony projít ručně, aby se zohlednily rozdíly mezi nimi. Další nevýhodou je to, že funkce *OnDomReady* je volána v momentě načítání stránky, pokud jsou však data do okna načítána *Ajaxem*, nedojde ke spuštění této funkce. Takže po odeslání formuláře *Ajaxem*, by bylo nutné řešit opětovnou změnu tlačítek. Pro tyto dva důvody byla vybrána první možnost.

Dále jsou popsány dva problémy, které způsobovaly nekorektní chování některých obrazovek. A díky kterým není zcela triviální vše obecně přepsat.

Resetování formuláře

Většina tlačítek pro resetování obsahu, je řešena standardní cestou a na *Ajaxové* volání je změněna postupem uvedeným výše. Některá tlačítka však obsahuji přímo *URL* odkaz na stránku. Ukázka z *mainUserColl.jsp*.

```
<skin:action type="button" key="button.reset" page="/filterUsers.do
 action=Prepare" styleClass="formButton" />
```

V tomto případě bylo nutné upravit funkci pro odeslání dat. Jelikož dojde jen k vymazání dat, není nutné odesílat hodnoty formulářových prvků.

```
function(pThis, pAction) {
      $('#' + $(pThis).closest('div').attr('id')).
        load(GetServerPath() + pAction);

};
```

Upravené tlačítko v šabloně pak vypadá následovně. V události `onclick` je volána výše popsaná funkce.

```
<skin:action type="button" key="button.reset"
 onclick=";cz.aimtec.dci.form.resetForm($(this), '/filterUsers.do
 action=Prepare')" styleClass="formButton" />
```

Nejednotný název akce formuláře

Některé jednoduché formuláře, které obsahují pouze jedno tlačítko, nemají ve formuláři pouze název akce, ale rovnou i konkrétní atributy. Tlačítko je pak velmi jednoduché a pouze odesílá formulář. Tak je tomu například v šabloně *mainFormColl.jsp*.

```
<html:form action="/filterForms.do?action=Search">
 <skin:action type="submit" key="button.search" styleClass="formButton"/>
```

Proto je nutné šablonu změnit následovně.

```
<html:form action="/filterForms.do" onsubmit="return false;">
…
 <skin:actiontype="submit" key="button.search"
  onclick=";cz.aimtec.dci.form.sendData((this), 'Search')"
   styleClass="formButton"/>
```

- Zakáže se odesílání formuláře standardní cestou a smaže se konkrétní atribut.
- V události `onclick` tlačítka je nastavená atribut z akce formuláře.

6 Testování

Pro testování se používá framework *Jameleon*, aplikace obsahuje přes 600 již hotových automatických testů. Testovací scénáře se zapisují ve formátu *XML*. Bylo nutné zajistit, aby nová verze aplikace byla také testovatelná. Z tohoto důvodu došlo k opuštění používání *iframe* (viz 6.1.1).

6.1.1 Použití iframe

Ve staré verzi byla zobrazena vždy jen jedna obrazovka s daty. Aby bylo možné zobrazit více dat, bylo nutné vymyslet, jak data do nových oken načíst. Jednou z možností byl *iframe*, kdy každé okno je *iframe*, který se chová jako samostatná stránka. Toto řešení je jednoduché v tom, že se o posílání dat a jejich načtení postará *iframe* sám.

S použitím *iframe* se objevily mnohé problémy. Zvětšování a zmenšování okna komponentou *JQuery resizable* má pro *iframe* problémové chování, pokud se okno zmenšuje. To vyřešilo obalení elementu *divem*, ovšem stále při větším počtu otevřených oken nebylo chování správné. Jako největší problém se ukázalo to, že automatické testy, nejsou schopny s *iframem* správně pracovat. 90% testů fungovalo správně, ale pro zbylé testy se testovací nástroj zacyklí ve smyčce a nedovede se dostat z *iframe* ven. Dalším problémem bylo, že pokud se *iframe* posune kamkoliv ve struktuře *DOM*, dojde ke ztrátě dat v něm. To značně stěžuje manipulaci s okny. Proto bylo nutné vymyslet jiné řešení. Byla použita technologie *Ajax*. (viz 5.10)

6.1.2 Testování přihlášení

V tomto případě bylo testováno přihlášení uživatele. Názvy tlačítek zůstaly stejné, takže jediné co bylo třeba změnit byl název použité *JSP* šablony (viz 5).

```
<jm:pageValidation
 functionId="Validate successful login"
 htmlPageValidationText="startEntryNew.jsp"/>
```

Testování se provádí úpravou souboru *include.txt*, do kterého se zadá jméno testu, případně jména testů, které se mají provést.

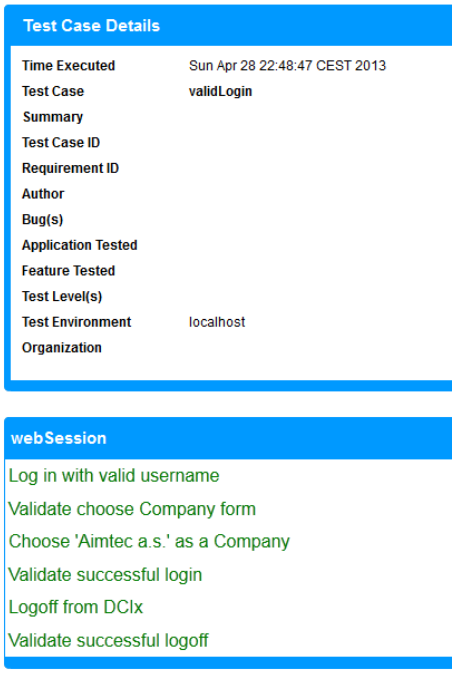
```
Logins/validLogin.xml
```

Po spuštění se do konzole vypíší výsledky testování.

```
[jmln-test] Beginning Session: "webSession"
[jmln-test] Begin: starting application
[jmln-test] End: starting application
[jmln-test] BEGIN: Log in with valid username
[jmln-test] SUCCEEDED: Log in with valid username
[jmln-test] BEGIN: Validate choose Company form
[jmln-test] SUCCEEDED: Validate choose Company form
[jmln-test] BEGIN: Choose 'Aimtec a.s.' as a Company
[jmln-test] SUCCEEDED: Choose 'Aimtec a.s.' as a Company
[jmln-test] BEGIN: Validate successful login
[jmln-test] SUCCEEDED: Validate successful login
[jmln-test] BEGIN: Logoff from DCIx
[jmln-test] SUCCEEDED: Logoff from DCIx
```

```
[jmln-test] BEGIN: Validate successful logoff
[jmln-test] SUCCEEDED: Validate successful logoff
[jmln-test] Ending Session: "webSession"
[jmln-test] validLogin.xml : PASSED
[jmln-test] ------------------------------------------------------------
[jmln-test]
[jmln-test] Test Run: 20130428221907140
[jmln-test] Tests run: 1 Failed: 0 Time: 7.605s
[jmln-test] ------------------------------------------------------------
all.my:
BUILD SUCCESSFUL
Total time: 12 seconds
```

Výsledky testování jsou k dispozici také ve formátu *HTML*.



Obrázek 6.1: Výsledky automatických testů ve formátu *HTML*

6.1.3 Testování dat ve formuláři

Pro testování zobrazení a práce s daty formuláře, bylo nutné opustit technologii *iframe* a přejít na *Ajax* (viz 6.1.1). Byl napsán jednoduchý test *searchDeliveryNoteSimple.xml*, který ukáže otevření nového okna, ve kterém dojde k použití filtru a zobrazení dat a následnému vymazání formuláře.

```
      <jm:webSession
            beginSession="true"
            application="dci">
            &createDN;
            <jm:login functionId="#1 Přihlaš se k DCIx"/>
```

```
            <jm:clickLink functionId="#2 Delivery"
                  htmlPageClickLink="Delivery"/>
            <jm:clickLink functionId="#3 Click on Delivery Notes"
                  htmlPageClickLink="Delivery Notes"/>
            <jm:sleep functionId="Počkej" time="2000"/>
            <jm:submitForm functionId="#4 Submit Search on Delivery Notes"
                  formName="orderDelNoteFilterForm"
                  submitButtonName="searchButton">
                  <jm:simpleParam name="displaytagFilter.order_number"
                   value="${delNoteNum}"/>
                  <jm:simpleParam name="displaytagFilter.order_numberTo"
                   value="${delNoteNum}"/>
            </jm:submitForm>
            <jm:sleep functionId="Počkej" time="2000"/>
            <jm:pageValidation functionId="#5 Zvaliduj, že je vyhledana
                                             Delivery Note"
                  htmlPageValidationLink="${delNoteNum}"/>
            <jm:submitForm functionId="#6 Submit Reset on Delivery Notes"
                  formName="orderDelNoteFilterForm"
                  submitButtonName="resetButton">
            </jm:submitForm>
            <jm:sleep functionId="Počkej" time="2000"/>
            <jm:pageValidation functionId="#7 Zvaliduj, že neni vyhledana
                                             Delivery Note"
                  htmlPageValidationLinkNotPresent="${delNoteNum}"/>
            <jm:clickLink functionId="#8 Logoff from DCIx"
                  htmlPageClickLink="Logoff" />
      </jm:webSession>
</jm:testcase>
```

`<jm:sleep functionId="Počkej" time="2000"/>` je použit proto, aby se počkalo, než se data *Ajaxem* načtou do okna a bude je tak možné otestovat, zda jsou správná. V současné verzi aplikace *DCIx*, však testeři již vyřešili použití *Ajaxu* tak, že není nutné použít funkcí `sleep`.

7 Závěr

V diplomové práci byly analyzovány technologie a nástroje, používané pro tvorbu uživatelských rozhraní současných webových aplikací. Z nich byly vybrány vhodné technologie s ohledem na známé požadavky funkcionality *DCIx* a na používané technologie ve společnosti Aimtec.

Součástí práce je též analýza nevyhovujících částí a vlastností původního *GUI* s odůvodněním, proč a jak je nutné tyto věci změnit. Dále byly připraveny celkem čtyři návrhy vzhledu nového *GUI*. Včetně animací, které lépe pomohly popsat zamýšlené ovládání. Tyto návrhy byly oponovány vedením firmy i samotnými uživateli a následně byl zvolen jeden z nich. Ten byl dále postupně rozšiřován.

Bylo připraveno 20 zcela nových souborů, což představuje přibližně 4000 řádek kódu. Do toho nebyla započítána řada existujících souborů, které bylo kvůli nové funkcionalitě nutné upravit. Velký čas také zabralo seznámení se s již fungující aplikací a postupy, které se v ní používají. Aplikace je rozsáhlá a používá několik frameworků, bylo nutné se v nich postupně zorientovat.

Během poslední fáze vývoje, při testování, se ukázalo, že zvolená technologie nebyla z pohledu testovatelnosti vhodná. Existující automatické testy nebyly schopny aplikaci ve všech případech správně otestovat, protože docházelo k zacyklení nástroje *Jameleon* v použitém *iframe*. Proto bylo nutno použít jinou technologii, kterou byl *Ajax*. Naštěstí, díky koncepčnímu návrhu a implementaci byl v naprosté většině případů přechod na novou technologii přímočarý.

Zadání práce je splněno, aplikace má nový interaktivní design, který splňuje požadavky uživatelů, včetně umožnění práce s více otevřenými okny najednou. Aplikace je také testovatelná technologií, kterou využívají již hotové automatické testy. Byl vytvořen ukázkový test nového rozhraní, který dokazuje, že při použití technologie *Ajax* je aplikace testovatelná, i když je v testech nutno provést drobné změny.

Seznam použité literatury

[JQBook]
Steven Holzner. jQuery: Visual QuickStart Guide. Peachpit Press, 2009. 240 s.
ISBN 0-321-64749-1.

[StrutsBook]
Srikanth Shenoy Struts Survival Guide: basics to best practices. ObjectSource Publications, 2004. 224 s. ISBN 0974848808.

[JSPBook]
Robert J. Brunner. JSP Practical Guide for Java Programmers. Elsevier, 2003. 175 s.
ISBN 978-1-55860-836-8.

[AjaxBooks]
Thomas Powell. Ajax: The Complete Reference. Mc Graw Hill, 2008. 654 s.
ISBN 978-0-07-149216-4.

Elektronické zdroje

[Aimtec]
AIMTEC.cz [online]. 2013, [cit. 2013-04-01].
URL: <http://www.aimtec.cz/>

[MVC]
Zdroják.cz [online]. 2013, [cit. 2013-04-01].
URL: <http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc>

[JQuery]
W3School - Jqeury [online]. 2013, [cit. 2013-04-10].
URL: <http://www.w3schools.com/jquery/jquery_intro.asp>

[Struts MVC]
Struts MVC Architecture Tutorial [online]. 2013, [cit. 2013-04-10].
URL: <http://www.dzone.com/tutorials/java/struts/struts-tutorial/struts-mvc-architecture-tutorial.html>

[OracleDoc]
Oracle9iAS Containers for J2EE User's Guide [online].
URL: <http://docs.oracle.com/cd/A97688_16/generic.903/a97681/jspprim.htm>

[Ajax]
W3School - Ajax [online]. 2013, [cit. 2013-04-12].
URL: <http://www.w3schools.com/ajax/ajax_intro.asp>

[DOM]
W3School - JavaScript HTML DOM [online]. 2013, [cit. 2013-04-12].
URL: <http://www.w3schools.com/js/js_htmldom.asp>

[JSON]
W3School JSON Syntax [online]. 2013, [cit. 2013-04-20].
URL: <http://www.w3schools.com/json/json_syntax.asp>

[PIA-HTML]
Rohlík Ondřej. HTML. 2013, [cit. 2013-04-22]. [dokument ve formátu PDF]
URL: <http://portal.zcu.cz/>

[PIA-CSS]
Rohlík Ondřej. Kaskádové styly - CSS. 2013, [cit. 2013-04-22].[dokument ve formátu PDF]
URL: <portal.zcu.cz>

[BOXMODEL]
Jak psát web [online]. 2013, [cit. 2013-04-25].
URL: <http://www.jakpsatweb.cz/okraje.html>

[GIT]
Git - Základy systému Git [online]. 2013, [cit. 2013-04-25].
URL: <http://git-scm.com/book/cs/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git>

[GitImage]
openSUSE - Git [online]. 2013, [cit. 2013-04-25].
URL: <http://pt.opensuse.org/Git>

[JqCon]
JQueryContextMenu [online]. 2013, [cit. 2013-04-25].
URL: <http://medialize.github.io/jQuery-contextMenu/>

Seznam zkratek

GUI – Graphical User Inrerface

ERP – Enterprise Resource Planning

HTML – HypertText Markup Languague

CSS – Cascading Style Sheets

MVC – Model View Controller

UI – User Interface

JSP – Java Server Pages

HTTP – Hypertext Transfer Protocol

DOM – Document Object model

PHP – Hypertext Preprocessor

ASP – Active Server Pages

XML – Extensible Markup Language

API – Application Programming Interface

HTTPS – Hypertext Transfer Protocol Secure

SGML – Standard Generalized Markup Language

CVS – Concurrent Version System

SVN – Subversion

XCF – eXperimental Computing Facility

GIF – Graphics Interchange Format

ODT – Open Document Type

PDF – Portable Document Format

A Uživatelské dokumentace

Nainstalování aplikace

Samotné instalaci musí předcházet nainstalování serveru *Apache Tomcat* a *MS SQL Server 2005* a vyšší. Nejsnadněji se instalace provede naklonováním z *git* repositáře. To se udělá snadno pomocí rozšíření *EGit* pro *Eclipse.* Další postup je uvedený ve složce *Navody,* která se nachází ve struktuře aplikace.

Používání aplikace

Aplikace je k dispozici na adrese, která záleží na konkrétním nastavení, nejčastěji se však nachází na:

```
http://localhost:8080/DCI/
```

Kde dojde k zobrazení úvodní obrazovky, kdy je nutné se přihlásit. Po úspěšném přihlášení se otevře aplikace.

Celá aplikace je vidět Obrázek 5.1.

Horní menu

Možnosti jsou popisovány zleva doprava (viz Obrázek 5.4).

- Logo aplikace – po kliknutí na něj se zobrazí domovská stránka společnosti Aimtec,
- Informace o uživateli – informace o uživatelském jméně, pod kterým je uživatel přihlášen, po kliknutí na odkaz, je možné editovat údaje,
- Informace o prostředí – informace o tom, jaká databáze a jaké prostředí jsou použity,
- Odkaz na nápovědu,
- Odkaz na zákaznickou podporu,
- Odhlášení se,
- Zobrazení informací o aplikaci *DCIx,*
- Vyhledávání – stačí zadat hledanou informaci a automaticky se objeví seznam odpovídajících položek,
- Schránka přijatých zpráv,
- Schránka přijatý emailů,
- Zadané úkoly,
- Vytvoření nové zprávy,
- Skrytí levého menu – dojde ke skrytí levého menu, bude více místa pro plochu a otevřená okna,
- Zobrazení levého menu – v případě, že je menu skryté – dojde k jeho opětovnému zobrazení,

- Zobrazení plochy – dojde k minimalizaci všechny otevřených oken a zobrazení plochy.

Levé menu

Levé menu obsahuje položky, která zpřístupní stránky z daty a formuláři. Menu je rozklikávací. Ikona plus značí, že položka obsahuje další menu. Ikona mínus naopak značí, že menu je již plně rozbalené. Jakákoliv jiná ikona značí, že se jedná o položku, která je odkazem přímo na formulář s daty (viz Obrázek 5.5).

- Kliknutím na položku v menu, dojde k rozbalení, nebo skrytí části menu, případně k otevření nového okna.
- Po kliknutí pravým tlačítkem myši na položku, která se otevírá v novém okně, se objeví kontextovém menu (viz Obrázek 5.6), pomocí kterého dojde k přidání nové ikony na plochu. Ikona bude přidána do levého horního rohu plochy.
- Přetažením položky z menu dojde rovněž k přidání ikony na plochu a to na místo, na kterém bylo tažení ukončeno.
- Každá ikona může být na ploše pouze jednou, pokud se ikona na ploše již nachází, dojde ke zobrazení dialogu o duplicitě ikon (viz Obrázek 5.8).

Plocha

Ikony na ploše je možné libovolně posouvat po ploše (viz Obrázek 5.7).

- Po kliknutí na křížek v pravém horním rohu ikony, dojde k zobrazení dotazu, zda si přejete ikony odstranit z plochy. Pokud ano, bude ikona z plochy vymazána, pokud ne zůstane na stejném místě.
- Po kliknutí na ikony na ploše dojde k otevření odpovídajícího okna.

Práce s okny

S otevřeným oknem se dá libovolně posouvat, či se dá měnit jeho velikost (viz Obrázek 5.9). V případě, že je oken zobrazeno více, je to s kterým je manipulováno, zobrazeno vždy nahoře. Okno je možné:

- Zavřít – kliknutím na křížek v pravém horním rohu, napřed dojde k zobrazení výzvy pro potvrzení této volby,
- Minimalizovat – dojde k minimalizaci okna, které bude přesunuto do spodní části obrazovky, kde se nachází lišta minimalizovaných oken.

Minimalizovaná okna

Okna, která jsou minimalizované ve spodní liště lze (viz Obrázek 5.10):

- Maximalizovat – kliknutím na ikonu, oknu bude vrácena původní velikost i pozice, jako před minimalizací,
- Zavřít – kliknutím na křížek v pravém horním rohu, napřed dojde k zobrazení výzvy pro potvrzení této volby.

B Obsah přiloženého média

- `DCIx` – adresář s aplikací *DCIx*, podrobnější popis viz 5.1
- `Navrhy` – adresář s návrhy nového uživatelského rozhraní
  - `1_zadani_graficky_navrh` – první čtyři návrh ve formátu *PNG*, spolu se soubory *GIMPu* ve formátu *XCF*
  - `2_zadani_konkretni_navrh` – úprava dvou návrhů, animace ve formátu *GIF*
  - `3_uprava_navrhu_pro_management` – další úprava dvou návrhů, animace ve formátu *GIF*
  - `4_uprava_vybrane_verze` – konečný návrh
- `Text` – adresář obsahující text diplomové práce formátech *PDF* a *ODT*