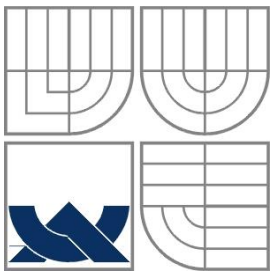


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
DEPARTMENT OF INFORMATICS

TVORBA KOMPONENT PRO ADOBE CQ5

CREATION OF COMPONENTS FOR ADOBE CQ5

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN PEŠAVA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR DYDOWICZ, Ph.D.

BRNO 2014

Abstrakt

Cílem práce je analyzovat a popsat tvorbu komponent pro systém Adobe CQ5. Výsledné komponenty budou následně implementovány a integrovány do systému. Dílčím cílem je vytvoření práce, která pomůže dalším programátorům s vývojem komponent pro tento systém.

Abstract

The goal of the thesis is to analyze and describe a creation of components for system of Adobe CQ5. Resulting components will be subsequently implemented and integrated into the system. The partial goal is to create a thesis that helps other programmers create components for this system.

Klíčová slova

Adobe CQ5, komponenta, informační systém, Hartmann - RICO

Keywords

Adobe CQ5, component, information system, Hartmann - RICO

Bibliografická citace

PEŠAVA, J. *Tvorba komponent pro Adobe CQ5*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 104 s. Vedoucí diplomové práce Ing. Petr Dydowicz, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 25. května 2014

……
podpis studenta

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce, Ing. Petru Dydowiczovi, Ph.D., za pomoc při řešení dané problematiky a přátelské vedení diplomové práce. Dále bych chtěl poděkovat Ondřejovi Semotánovi a Jakubovi Koláčkovi za rady a technické podklady z firemního prostředí.

Obsah

1 Úvod 7
2 Cíle práce, metody a postupy zpracování 8
2.1 Cíl práce 8
2.2 Postup zpracování 8
2.3 Použité metody 8
2.3.1 Obecné metody a analýzy 8
2.3.2 Metody požívané k vývoji softwaru 9
2.3.3 Metody pro sběr informací 9
3 Teoretická východiska práce 10
3.1 Základní pojmy, formátování 10
3.2 Metody použité k analýze současného stavu 10
3.2.1 SWOT analýza 10
3.2.2 Demingův cyklus 11
3.2.3 RACI matice 12
3.2.4 GE matice 13
3.3 Analytické metody použité k vývoji softwaru 14
3.3.1 Drátěný model 14
3.3.2 EPC diagram 15
3.3.3 UML diagramy 15
3.3.4 ER diagramy 17
3.4 Ekonomické metody použité k vývoji softwaru 18
3.4.1 COCOMO 18
3.4.2 Funkční body 20
3.4.3 PERT 20
3.5 Systém Adobe CQ5 21
3.5.1 Úvod do Adobe CQ5 21
3.5.2 Uživatelské rozhraní Adobe CQ5 22
3.5.3 Struktura Adobe CQ5 28
3.5.4 Vývojové prostředí 30
3.5.5 Vývoj komponent pro Adobe CQ5 31
4 Analýza současného stavu 34
4.1 Analýza společnosti Hartmann – RICO a.s. 34
4.1.1 Popis společnosti 34
4.1.2 Analýza současného vývoje komponent pro systém Adobe CQ5 36

4.2 Analýza systému Adobe CQ5 44
4.2.1 Porovnání Adobe CQ5 s konkurencí 44
4.2.2 Postavení systémů v konkurenčním prostředí 47
4.2.3 Analýza kvality prostředí pro vývoj 49
4.3 Shrnutí analýzy 49
5 Vlastní návrh řešení 51
5.1 Tvorba komponent 51
5.1.1 Aplikace „Stížnosti“ 51
5.1.2 Aplikace „Mimořádné odměny“ 63
5.2 Návrh interních pravidel 69
5.2.1 Návrh struktury vrstev 69
5.2.2 Návrh „Code Rules“ 71
5.2.3 Návrh wiki stránek 72
5.2.4 Návrh refaktoringu 73
5.3 Ekonomické zhodnocení navrhovaného řešení 74
5.3.1 Ekonomické zhodnocení vývoje komponent 74
5.3.2 Ekonomické zhodnocení zavedení interních pravidel 75
6 Závěr 76
Literatura 77
Seznam příloh 80
Seznam obrázků 81
Seznam zdrojových kódů 83
Seznam tabulek 84

1 Úvod

Tato práce se zabývá tvorbou komponent pro systém Adobe CQ5 nasazený na serverech ve firmě Hartmann – RICO a.s. Postupně se budeme zaměřovat na cíle diplomové práce a metody, které používáme k jejich dosažení. V dílčích částech popíšeme teoretická východiska a analýzu současného stavu. V hlavní části budeme popisovat návrh vlastního řešení, ke kterému jsme po předchozích analýzách dospěli. V samotném závěru výsledky ekonomicky a technicky zhodnotíme.

Lidé se snažili odjakživa sjednocovat ve větší spolky a využívat tak jejich výhod. Postupem času se lidstvo vyvinulo od rodového ke státnímu uspořádání společnosti, které je nejčastější v moderní době. K tomu je však potřebná velká míra komunikace a informovanosti. Stejné je to s obchodem. Obchod je stejně stará činnost, jako lidstvo samo. Lidé od svého počátku měli sklony směňovat a obchodovat. Ovšem jako jednotlivci. Dnes jsme svědci velkým nadnárodních korporací, kde je nutné mít vnitřní systémy umožňující vnitřní řízení.[1,2]

S tímto přechodem přišla i změna prostředků, které používáme k řízení firemních zaměstnanců a procesů s tím spojených. Z papírové formy komunikace a osobního předávání informací se přešlo na komunikaci elektronickou umožňující flexibilnější a efektivnější řízení firmy. Dnes pro tyto účely používáme univerzální informační systémy zajišťující všechny potřebné funkcionality v rámci jednoho nástroje. Tato změna přinesla do firem mnoho změn. Vznikly nové firemní oddělení mající na starost udržování a vývoj těchto systémů.

Vývoj softwaru není tak jednoduchou záležitostí, jak by se mohlo na první pohled zdát. Vyvinout program a ověřit jeho správnou funkčnost je základem tvorby softwaru. V dnešní době k tomuto hlavnímu požadavku je přidáváno několik pomocných. Jedná se především o bezpečnost a ochranu proti zneužití, správnou strukturovanost a efektivitu, případně konektivitu s okolními aplikacemi. Z tohoto důvodu je dílčím cílem naznačení parametrů, které správná aplikace obsahuje a metody nutné k jejich dosažení.

2 Cíle práce, metody a postupy zpracování

V této kapitole budou definovány vytyčené cíle práce, postupy a metody použité v textech jednotlivých kapitol. Zaměříme se na metody potřebné pro obecnou analýzu a správný vývoj softwaru. V poslední části popíšeme metody pro sběr informací.

2.1 Cíl práce

Cílem práce je analyzovat a popsat tvorbu komponent pro systém Adobe CQ5. Výsledné komponenty budou následně implementovány a integrovány do systému. Dílčím cílem je vytvoření práce, která pomůže dalším programátorům s vývojem komponent pro tento systém.

2.2 Postup zpracování

V prvním kroku je potřené prostudovat dostupné materiály o Adobe CQ5. Ty společnost Adobe udržuje na stránkách „http://dev.day.com/content/ddc/en.html“. Zde je možné naleznout informace pro uživatele a vývojáře. V dalším kroku je nutné prostudovat zavedení Adobe CQ5 uvnitř firmy Hartmann – RICO a.s. V tomto kroku se seznámíme s prostředky, které jsou využívány. Jedná se především o databáze a ostatní systémy, jako např.: SAP, atd. Ve třetím kroku budeme prostudovávat dostupné komponenty a snažit se pochopit logiku kódu a principy vývoje uvnitř IT oddělení. V následující části se pokusíme vyvinout a nasadit do praxe několik komponent dle standardních kritérií pro vývoj webových aplikací. Závěrečná část se bude zabývat zhodnocením dosažené práce.

2.3 Použité metody

Kapitola bude popisovat použité metody v této diplomové práci. Metody pro tento účel rozdělíme do několika oblastí. Jedná se o oblasti obecných metod a analýz, metod používaných k vývoji softwaru a metod pro sběr informací.

2.3.1 Obecné metody a analýzy

Tyto metody se většinou nevztahují k určitému oboru a budou použity pro obecné účely, jakými mohou být aktuální postavení firmy v konkurenčním prostředí, či silné a slabé stránky vývoje komponent pro systém Adobe CQ5. Mezi tyto metody budeme řadit SWOT analýzu, RACI matici SMART matici a Porterovu analýzu.

2.3.2 Metody požívané k vývoji softwaru

Metody uvedené v této části se stahují čistě k jednomu oboru a tím je vývoj softwaru. Tyto metody budeme používat pro různé analýzy potřebné před započetím samotného vývoje nebo pro závěrečné ekonomické zhodnocení.

Mezi tyto metody řadíme:

- EPC diagramy
- UML diagramy
- ER diagramy
- COCOMO
- Funkční body
- PERT

2.3.3 Metody pro sběr informací

Metody pro sběr informací byly použity pro získávání dat o systému Adobe CQ5 z dostupných materiálů na internetu i dat poskytnutých firmou Hartmann – RICO a.s.

Jedná se především o metody:

- Analýza dokumentů
- Nestrukturované rozhovory
- Polostrukturované rozhovory
- Analýza zdrojových souborů
- Analýza vnitřních databází

3 Teoretická východiska práce

Tato kapitola se budeme zabývat teoretickými východisky práce, které jsou v ní použity. V úvodu budou popsány základní pojmy a použité formátování textu. V dalších kapitolách popíšeme použité nástroje a metody. V poslední části se zaměříme na samotný systém Adobe CQ5 a rozebereme jeho strukturu i vlastnosti. Tato část bude tvořit teoretický základ pro analýzu současného stavu a návrh vlastního řešení.

3.1 Základní pojmy, formátování

V textu kapitol je pro přehlednost použito následujíc formátování. *Kurzívou* jsou psány názvy komponenty Adobe CQ5, anglických slov a termínů, logických celků komponent a názvů technologií. `Strojovým písmem` jsou značeny názvy objektů, tříd a jejich metod.

3.2 Metody použité k analýze současného stavu

V této práci budeme potřebovat několik metod, které nám pomohou analyzovat současný stav firmy Hartmann – RICO a.s., nasazeného systému Adobe CQ5 a techniku vývoje komponent. K vyjádření této analýzy budeme používat GE matici a SWOT analýzu. Dále budeme popisovat Demingův cyklus tvořící základní metodiku pro vývoj aplikací.

3.2.1 SWOT analýza

Jedná se o metodu nahlížející na danou problematiku ze čtyř stran a umožňující definovat následující atributy:

- Silné stánky
- Slabé stránky
- Příležitosti
- Hrozby

Průnikem jednotlivých atributů je pak možné definovat strategie pro zlepšení stavu dané problematiky. Výběr dané strategie vždy záleží na hlavní strategii a měla by s ní být v souladu. Jedná se o tyto strategie:

- **S-O strategie** – kombinace silné stránky a příležitosti
- **W-O strategie** – kombinace slabé stránky a příležitosti
- **S-T strategie** – kombinace silné stránky a hrozby
- **W-T strategie** – kombinace slabé stránky a hrozby

Tato metoda byla vyvinuta na Stamfordově univerzitě v Kalifornii Albertem Humphreymem v 60. a 70. letech 20 století. Tehdy byla pro analýzu použita data od 500 nejvýznamnějších amerických společností. [3]

Vnitřní faktory / Vnější faktory	Slabé stránky (W)	Silné stránky (S)
Příležitosti (O)	**WO strategie** **„hledám“** *Překonání slabé stránky využitím příležitostí*	**SO strategie** **„využití“** *Využití silné stránky ve prospěch příležitosti*
Hrozby (T)	**WT strategie** **„vyhýbám“** *Minimalizace slabé stránky a vyhnutí se ohrožení*	**ST strategie** **„konfrontace“** *Využití silné stránky k odvrácení ohrožení*

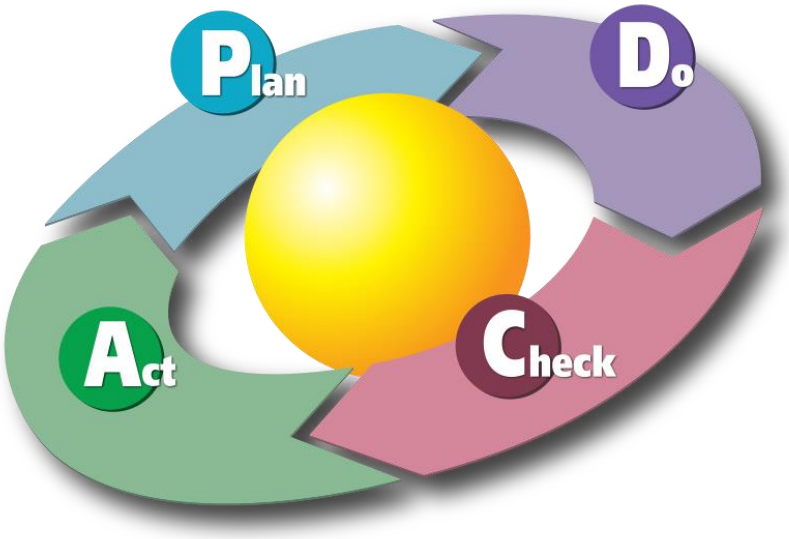
Obrázek 1: SWOT analýza. [3]

3.2.2 Demingův cyklus

Tato metodologie je základním nástrojem probíhajícím v jednotlivých cyklech pro zkvalitňování procesů pro obecné použití. Dle poznatků amerického fyzika, inženýra a statistika Waltera A. Shewharta byl definován, a do širší vědomosti prosazen, americkým profesorem Williamem Edwardem Demingem. Metoda rozděluje aktivity procesu na čtyři základní části, jedná se o tyto body:

1. **Plan** (Plánuj)
2. **Do** (Dělej)
3. **Check** (Kontroluj)
4. **Act** (Jednej)

Principem této metody je snaha o zlepšování současného stavu dané problematiky. Na začátku je nutné definovat plán, kterého chceme dosáhnout a následně jej realizovat. V posledních dvou krocích se snažíme kontrolovat, zda se stanovený plán povedl. V případě nezdaru následuje oprava (korekce) mající za úkol cíle dosáhnout a případně před dalších cyklem definovat kvantitativní zlepšení, či opatření. Důležitou podmínkou je neustále zvyšování kvality po každém proběhnutém cyklu. Zmíněný cyklus je někdy označován také jako PDCA cyklus nebo Shewhartův cyklus. Velmi často je využíván v oboru jakosti. [5, 7]



Obrázek 2: Demingův cyklu (PDCA cyklus). [4]

3.2.3 RACI matice

V praxi se RACI matice používá pro definování odpovědností jednotlivým osobám. Ve firmách se definuje na různých úrovních – mezi divizemi, uvnitř divizí, uvnitř jednotlivých týmu, atd. RACI je zkratka počátečních písmen čtyř slov:

- **R – Responsible:** Tento atribut přiřazuje danému subjektu odpovědnost za provedení určitého úkolu.
- **A – Accountable:** Tento atribut definuje odpovědnost za daný úkol, přičemž tato osoba jej nemusí vykonávat, ten může vykonávat osoba jiná.
- **C – Consulted:** Tímto atributem jsou označeny subjekty poskytující radu či konzultaci k dané problematice.
- **I – Informed**: Tímto atributem jsou označeny subjekty, které o dané problematice mají být informovány.

RACI matice je velice silný nástroj pro řízení firemní odpovědnosti při provádění vnitřních i vnějších aktivit. Je však nutné dodržovat základní pravidla a doporučení. Tato doporučení bychom mohli rozdělit na vertikální a horizontální analýzu.

Při vertikální analýze zkoumáme jednotlivé prvky sloupce. Sloupce představují odpovědnosti daného subjektu. V případě, že jsou všechny položky ve sloupci vyplněny, může být daný subjekt přetížen a bylo by vhodné převést kompetence k jednotlivým aktivitám na jinou osobu. Tento subjekt by taktéž měl zastávat rozumné role, není vhodné zastávat ve všech řádcích jen role s označením A a R.

Horizontální analýzou se snažíme zjistit vyváženost jednotlivých rolí na daném projektu. Základním pravidlem je, že v daném řádku by se mělo vyskytovat právě jedno A – což znamená, že

projekt vede pouze jeden subjekt a právě jedno R – vykonává jeden subjekt. V případě více R je vhodné daný úkol rozčlenit na více částí a ty pak hodnotit samostatně. [8, 9]

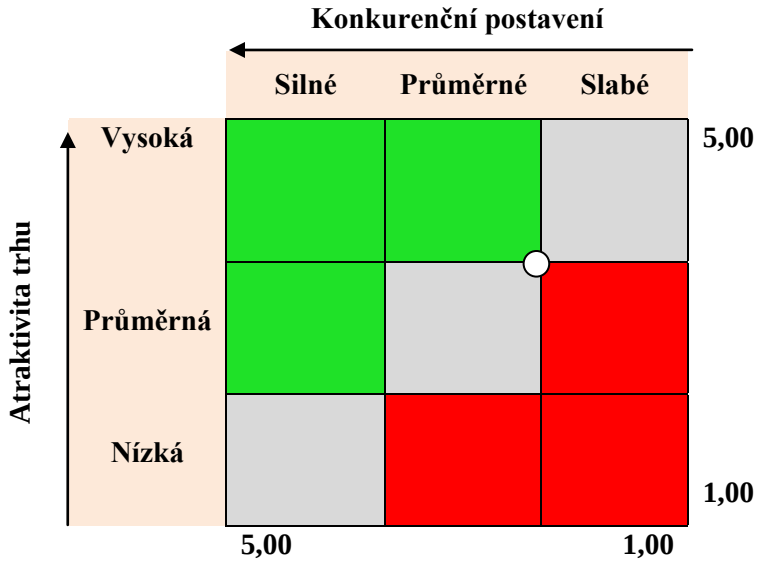
Proces / osoba	Alexandra Malá	Daniela Votavová	Tomáš Klapal	Karel Němec
Návrh architektury SW	CA	I	I	R
Zajištění kancelářských potřeb	R	CA	I	I
Týmové porady	I	R	CA	I
Firemní audit	I	I	R	CA

Tabulka 1: GE matice pro jednoduchý případ (Zpracováno dle [8, 9]).

3.2.4 GE matice

Matice General Electric/McKinsey je výsledkem projektu firmy *McKinsey & Company* z počátku 70. let 20. století. Touto technikou se snažili stanovit pro korporaci General Electric, co má v budoucnu se svými společnostmi podniknout. Zda si je ponechat a dále rozvíjet, či prodat.

Metoda dělí na danou problematiku na dva pohledy. Na atraktivitu odvětví a konkurenční sílu. Každý pohled se pokusíme diverzifikovat na další body a ty pak ohodnotit v rozmezí od 1 do 5. Tyto vlivy pak sečteme a vyjde nám hodnota daného pohledu, které vyneseme do matice.



Obrázek 3: Příklad GE matice (Zpracováno dle [56]).

Výsledky GE matice mohou pomoci s uvědoměním současné situace. Pokud se analyzovaný subjekt nenachází v požadované oblasti, pak je nutné definovat nové plány, které umožní posun k vytyčeným cílům. [56]

3.3 Analytické metody použité k vývoji softwaru

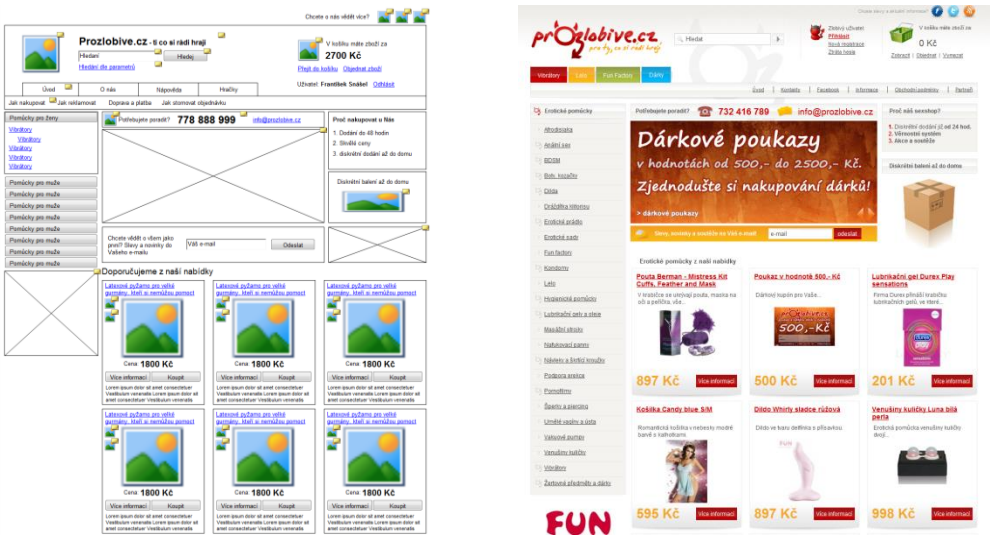
V této kapitole se zaměříme na metody používané v oblasti informatiky pro správnou analýzu a vývoj softwaru. Mezi tyto nástroje patří Drátěné modely, UML diagramy a ER diagramy. Budeme je především využívat v části vlastního řešení pro vývoj komponent pro systém Adobe CQ5.

3.3.1 Drátěný model

Drátěný model je v největší míře používán ve webových technologiích. Pomocí těchto nástrojů je vytvořena specifikace celého webu a přesněji řešeno rozmístění jednotlivých prvků na stránkách a podstránkách, včetně vzájemné komunikace. Jednotlivými prvky je myšleno menu, submemu, patička, hlavička, různé obrázky, rozmístění textu, atd. Klient připravený model odsouhlasí a podepíše, případně navrhne změny k zapracování. V dalších krocích drátěný model poskytuje programátorům zhmotněnou zákazníkovu vizi, kterou ve výsledku očekává. Drátěné modely můžeme dělit do těchto kategorií:

- Textové drátěné modely
- Blokové drátěné modely
- Podrobné drátěné modely
- Interaktivní drátěné modely

Vývoj s pomocí drátěných modelů sice trvá déle a samotná fáze vývoje se opozdí, nicméně zákazník má jistotu, že dostane skutečně to, co očekává. Další výhodou je, že tento model se stává součástí smlouvy a je možné dle něho řešit případné spory. [10]



Obrázek 4: Příklad drátěného modelu a reálného webu. [10]

3.3.2 EPC diagram

Diagram procesního řetězce řízeného událostmi (EPC) je diagram definující průběh provádění procesu na jednotlivé části s jasně definovanými odpovědnostmi. Paradigma definuje následující prvky:

- **Událost** – popisuje aktuální stav procesu
- **Procesní aktivita** – činnost měnící stav procesu
- **Nástroj pro podporu procesní aktivity** – Podpora funkcionalit informačního systému.
- **Procesní role** – definují vztah a oprávnění k procesní aktivitě
  - Responsible – zodpovědnost za provedení procesní aktivity
  - Accountable – zodpovědnost za aktivitu
  - Consulted – možnost konzultace s danou aktivitou
  - Informed – povinnost informovat danou osobu o aktivitě
- **Logický operátor XOR** – vykonává pouze jednu větev při dělení.
- **Logický operátor AND** – vykonává všechny větvě při dělení
- **Logický operátor OR** – vykonává jednu a více větví při dělení

Diagram je pro svou všeobecnost používán pro definice a popisy procesů uvnitř firem. Poskytuje všechny potřebné objekty pro tento popis. [39]

3.3.3 UML diagramy

UML (Unified Modeling Language) je v oboru softwarového inženýrství využívaný grafický jazyk pro specifikaci, návrh nebo dokumentaci vyvíjené aplikace. Jedná se o komplexní množinu všech potřebných nástrojů umožňující kvalitní vývoj softwaru. Mezi tyto nástroje řadíme například diagram tříd, diagram užití, diagram aktivit, diagram komponent, diagram objektů, diagram balíčků, sekvenční diagram nebo diagram komunikace. V této diplomové práci budeme využívat pouze první tři uvedené diagramy. [11]

Jazyk UML byl prvně použit v 80. letech 20. století těmito třemi autory – Grady Boochem, Jamesem Rumbaughem a Ivara Jacobsonem, kteří současně tvořili vlastní grafické nástroje. Na počátku devadesátých let se tito pánové domluvili a vytvořili systém jediný, pojmenovaný UML. [11]

3.3.3.1 **Diagram případu užití**

Tato technika je používána většinou v počátečních fázích vývoje softwaru. Umožňuje specifikovat jednotlivým autorům jejich aktivity a tím kvalitněji popsat funkcionalitu výsledného systému. V diagramu se vyskytují tři základní prvky:

- **aktér/účastník** – prvek je vyjádřen jednoduchou postavou člověka, označuje jednotlivé uživatele užívající systém.
- **případ užití** – prvek definuje všechny aktivity a funkcionality systému, které musí být v realizaci obsaženy.
- **vztah** – prvek vyjádřený čárou vedoucí od případu užití a aktérovi nebo od případu užití a jinému případu užití. Definuje přesné závislosti mezi jednotlivými prvky v systému. Vztah je vhodné pojmenovat dle zamýšlené aktivity. Vymezeny jsou dva základní vztahy, které mají přesnou sémantiku užití.
    - Include – případ užití obsahuje jiný případ užití.
    - Exclude – případ užití rozšiřuje jiný případ užití.

Někdy jsou tyto diagramy označeny jako „Use case diagramy“. [11]

3.3.3.2 Diagram tříd

Diagram zobrazuje v daném systému použité třídy a vztahy mezi nimi, včetně hierarchické struktury. Jedná se o statické vztahy, které neříkají nic o vzájemných interakcích. Diagram tříd může také obsahovat mimo tříd i jiné prvky – rozhraní, seskupení, instance nebo vztahy. Každý prvek v tomto diagramu má popsanou vnitřní strukturu. Jedná se především o atributy, včetně jejich viditelnosti, konstanty nebo příslušné metody. Ve všech standardních programovacích jazycích máme k dispozici tři základní viditelnosti. Jedná se o viditelnost:

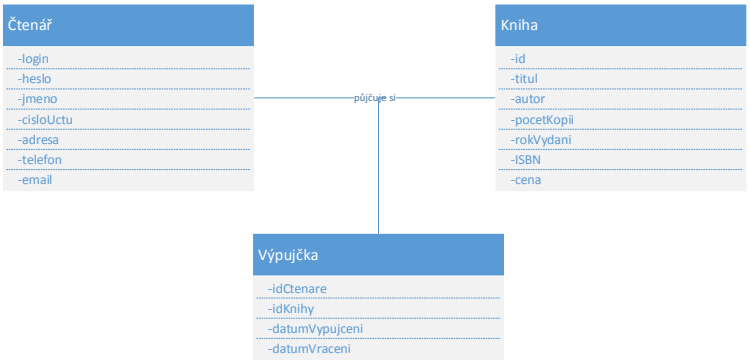
- **Veřejnou** – běžně se označuje znaménkem „+“. Tato metoda, či atribut je viditelná všemi ostatními subjekty, které s touto třídou spolupracují. U atributů se tato viditelnost silně nedoporučuje. U metod je doporučeno používat ji obezřetně a co nejméně. Nevhodné použití zvyšuje závislosti mezi třídami, což může později způsobovat problémy při restrukturalizaci celého kódu.
- **Soukromou** – bývá označována znaménkem „-“. Atributy a metody jsou viditelné pouze uvnitř třídy. Nejsou tedy přístupné ostatním subjektům. U atributů jsou pak definovány tzv. „gettery“ a „settery“ pro komunikaci s dalšími objekty.
- **Chráněnou** – v diagramech je označována znakem „#“. Viditelnost jednotlivých metod a atributů je pouze uvnitř třídy a všech potomků dané třídy. Tzn., že viditelnost je možné získat dědičností z předka na potomka. Jedná se o velmi silnou, statickou a za běhu neměnitelnou vazbu. Je doporučeno ji nahrazovat kompozicí.

Pomocí diagramu tříd je možné definovat všechny běžné druhy používaných struktur. Mezi tyto struktury zahrnujeme:

- **Dědičnost**: definuje vztah mezi rodičem a potomkem. Způsob vyjádření generalizace – specializace. Potomek má všechny vlastnosti rodiče, které nemají viditelnost nastavenou na „private“.

- **Agregace:** Představuje vztah mezi částí a celkem, přičemž obě části na sobě nejsou existenčně závislé, tudíž mohou existovat samostatně. Může se například jednat o vztah objektů „Počítač“ – „Tiskárna“.
- **Kompozice:** Jedná se o silnější formu vztahu. Princip je podobný agregaci s tím rozdílem, že jedna z komponent nemůže existovat samostatně. Samostatná existence většinou postrádá sémantický smysl. Například se může jednat o vztah „Faktura“ – „Položka“.
- **Asociace:** Vyjadřuje vztah mezi jednotlivými třídami. V diagramu je zobrazena jako čára s popisem mezi dvěma prvky.
- **Rozhraní:** Graficky se znázorňuje stejně jako třída, nicméně obsahuje klíčové slovo „interface“. Deklaruje pouze metody s viditelnosti „public“. V praxi se používá pro zpřesnění komunikace mezi jednotlivými typy tříd.

Diagram tříd nejvěrněji definuje vnitřní strukturu aplikace. V podnikové praxi navazuje na specifikaci požadavků a počáteční analýzu. Pro vývojáře a analytiky se jedná o základní dokument zobrazující realitu pomocí programovacích technik. [11, 12, 13]



Obrázek 5: Příklad diagramu tříd (Zpracováno dle [12]).

3.3.4 ER diagramy

V softwarovém inženýrství jsou tyto diagramy využívány pro abstraktní a konceptuální znázornění dat. ER diagramy obsahují tyto prvky:

- **Entitní množiny**: představují reálné tabulky vytvořené ve výsledné relační databázi.
- **Vztahové typy**
  - Kardinalita
  - Stupeň
  - Členství
- **Integritní omezení**

Na první pohled mohou být ER diagramy podobné jako diagramy tříd. V některých jednoduchých případech tomu tak opravdu být může. Rozdíly zde jsou patrné až při netriviálních aplikačních strukturách. Základním smyslem těchto diagramů je přizpůsobit architekturu tříd do technologického

prostředí relačních databází, které neumožňují některá data a závislosti uložit tak, jak jsou strukturovány pomocí tříd. [14, 15, 16]

3.4 Ekonomické metody použité k vývoji softwaru

V této kapitole budou popsány ekonomické metody použité pro stanovení očekávaných nákladů pro vývoj. Tyto metody neslouží jen pro určení ceny, která se z nich samozřejmě následné odvozuje, ale také na dobu trvání vývoje.

Každé vedení společnosti, respektive projektoví manažeři, potřebují pro stanovení nákladů vývoje určit, dle specifikace a počáteční analýzy, nákladnost provedených prací. Cílem analýzy je zhodnocení reálnosti projektu a jeho následné rozčlenění na jednodušší části. Dle těchto odhadů je možné stanovit pomocí níže popsaných nástrojů očekávanou dobu vývoje a následně i cenu. Mezi tyto nástroje řadíme metodiky COCOMO, Funkční body a PERT.

3.4.1 COCOMO

COCOMO (Constructive Cost Model) je program pro stanovení ceny vývoje zamýšleného softwaru vyvinutý Barry W. Boehmem. Program je postavený na velkém množství vstupních údajů, které musí uživatel zadat. Výsledek je vyhodnocen pomocí vnitřních konstant a nastavení vycházejících z empirických dat. Součástí empirických dat jsou zkušenosti z předchozích komplexních projektů a rozhovory s manažery. COCOMO poskytuje tři základní úrovně detailu:

- **Základní model**
- **Střední model**
- **Pokročilý model**

Tento nástroj je vhodný pro všechny rozsahy projektů a je možné jej použít na menší i rozsáhlejší projekty. V tomto hledisku jsou definovány tři základní vývojové módy:

- **Organický mód:** vhodný pro jednodušší a dobře řešitelné projekty bez složitých závislostí.
- **Bezprostřední mód:** tento mód je vhodné použít pro středně složité projekty.
- **Vázaný mód:** koncipován pro rozsáhlé projekty vyznačující se vysokou náročností na řízení vývoje

V současné době existuje modernější nadstavba programu COCOMO s označení COCOMO II. Tato změna byla zapříčiněna novými jevy a procesy v oboru softwarového návrhu. [18, 37]

Parametry ovlivňující úsilí nutné k jeho vytvoření jsou závislé na atributech uvedených v tabulce číslo 2. Následně je možné vypočíst skutečnou dobu vývoje a optimální počet pracovníků na měsíc.

Parametr	Very Low	Low	Nominal	High	Very High	Extra High
PREC	0,05	0,04	0,03	0,02	0,01	0,0
FLEX	0,05	0,04	0,03	0,02	0,01	0,0
RESL	0,05	0,04	0,03	0,02	0,01	0,0
TEAM	0,05	0,04	0,03	0,02	0,01	0,0
PMAT	0,05	0,04	0,03	0,02	0,01	0,0
RELY	0,75	0,88	1,00	1,15	1,40	
DATA		0,94	1,00	1,08	1,16	
CPLX	0,75	0,88	1,00	1,15	1,30	1,65
RUSE		0,89	1,00	1,16	1,34	1,56
DOCU	0,85	0,93	1,00	1,08	1,17	
TIME			1,00	1,11	1,30	1,66
STORE			1,00	1,06	1,21	1,56
PVOL		0,87	1,00	1,15	1,30	
ACAP	1,5	1,22	1,00	0,83	0,67	
PCAP	1,37	1,16	1,00	0,87	0,74	
PCON	1,23	1,10	1,00	0,88	0,80	
AEXP	1,26	1,12	1,00	0,88	0,80	
PEXP	1,24	1,11	1,00	0,9	0,82	
PCON	1,26	1,11	1,00	0,91	0,83	
LTEX	1,20	1,10	1,00	0,88	0,75	
SETE	1,24	1,10	1,00	0,92	0,85	0,79
SCED	1,23	1,08	1,00	1,04	1,10	

Tabulka 2: Parametry ovlivňující výpočet celkového úsilí pro vývoj aplikace. [54]

Rovnice pro výpočet hodnot jsou definovány takto:

$$E = a * KSLOC^b,$$

$$T = c * E^d,$$

$$P = E * T,$$

kde E je potřebná práce na vývoj uvedená v „člověkoměsících“, $KSLOC$ odhadovaný počet řádků v tisících, T vyjadřuje dobu vývoje a P optimální počet alokovaných lidí na projekt na měsíc. Konstanty a, b, c, d jsou hodnoty závislé na modelu a módu. [54, 49]

3.4.2 Funkční body

Tvorba softwaru je v podstatě proces podobný výrobnímu postupu vyžadující lidskou práci. Na základě této úvahy stačí určit jednotku výroby a cenu práce na jednotku. Touto jednotkou je myšlen právě funkční bod. V pravém slova smyslu funkční bod definujeme jako normalizovanou jednotku softwarového projektu.

$$odhad = velikost\ projektu * složitost * rizikové\ faktory$$

Tato metodika je založená na zkoumání aplikačních funkcionalit a jejich následnému ohodnocení. Nezkoumá technickou oblast, ani velikost kódu. Měří vstupní a výstupní funkce, vnitřní paměti, vnější paměti, atd. Funkční body vztahujeme k:

- **Transakčním funkcím**
  - Externí vstupy (EI)
  - Externí výstupy (EO)
  - Externí dotazy (EQ)
- **Datovým funkcím**
  - Vnitřní logické soubory (ILF)
  - Soubory vnějšího rozhraní (EIF)

Cílem této metody je získání komplexní jednotky postupnou analýzou softwaru, kterou je možné následné přepočíst na dobu realizace. Ta se největší měrou promítne do výsledných nákladů potřebných pro zajištění vývoje. [17, 49]

3.4.3 PERT

Existuje několik způsobů, jak stanovit informace o době trvání analyzovaného projektu. Jedním z nich je i metodika PERT – Program Evaluation and Review Technique. Ta zavádí tři odhady trvání průběhu:

- Optimistický
- Pesimistický
- Očekávaný

Výsledná doba trvání je vypočtena na základě níže uvedeného vzorečku, který dle zadaných parametrů očekávaný odhad.

$$t_e = \frac{a + 4m + b}{6},$$

kde t_e - očekávaná doba, a – optimistický odhad trvání, b – pesimistický odhad trvání, m – realistický odhad průběhu.

Metoda PERT definuje interval důvěry, který je založen na standardní odchylce doby. Díky tomu je možné plánovat rezervy pokrývající případné neúspěchy.

$$s = \frac{b - a}{6},$$

kde s – odchylka očekávané doby.

Velkou výhodou této metody je zahrnutí nejistoty do výpočtu finální doby trvání. Nevýhodami je velká pracnost a předpoklad neomezených zdrojů. [36]

3.5 Systém Adobe CQ5

V této kapitole popíšeme systém Adobe CQ5, který tvoří základ diplomové práce. Budou zde uvedeny podstatné funkcionality, včetně názorných příkladů. Systém bude popsán ze dvou hlavních oblastí – uživatelské a technologické. Uživatelská oblast bude sloužit k lepšímu pochopení podstaty funkcionality, technologická pak k názornějšímu pochopení postupů pro vývoj jednotlivých součástí systému. V samém závěru se více zaměříme na strategii vývoje jednotlivých komponent v Adobe CQ5.

3.5.1 Úvod do Adobe CQ5

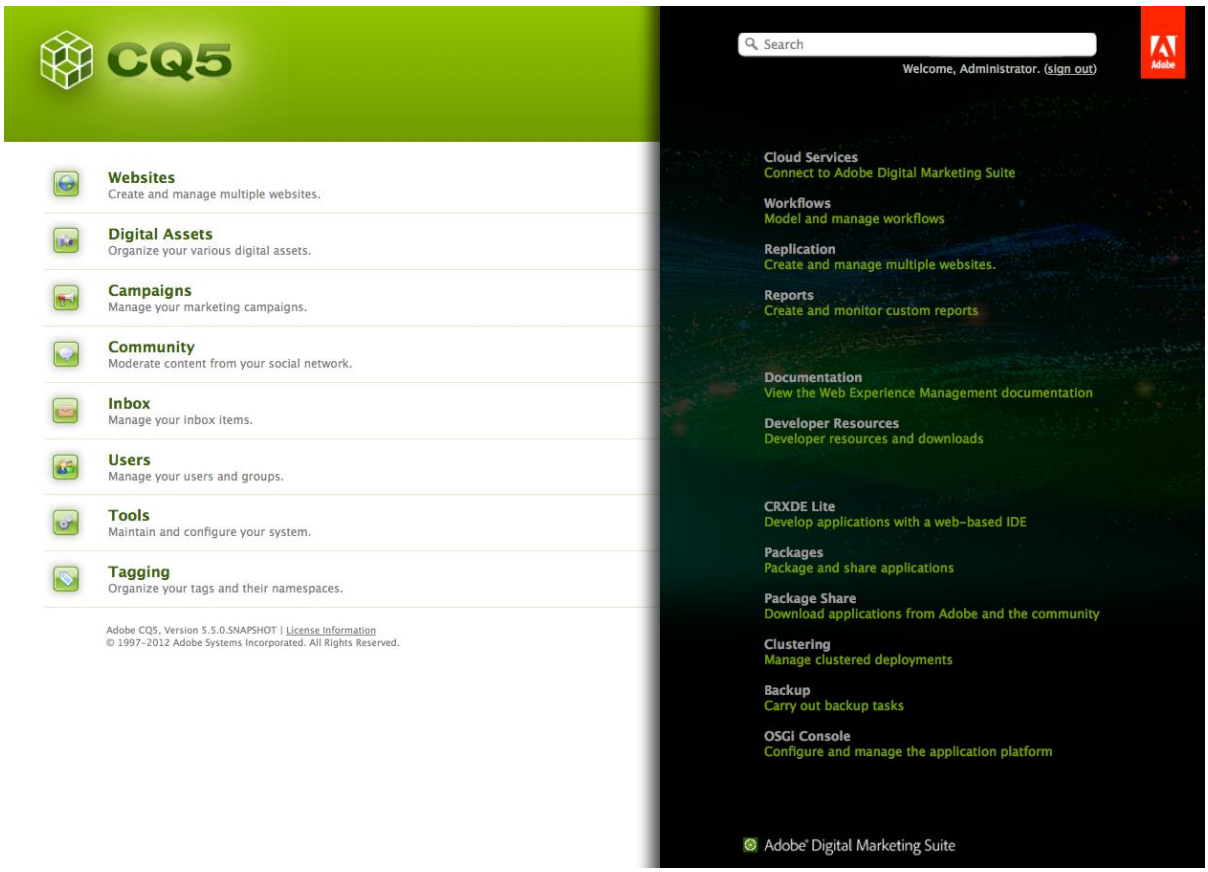
Systém Adobe CQ5 je komplexní nástroj pro správu obsahu a informací určený především pro velké společnosti. Počátky vývoje jsou datovány do roku 1993, kdy v Basileji vzniká společnost Day Software, která tento systém začala vyvíjet. V začátcích je označován jako WCM – Web Content Management a využívá dostupné technologie jako Java SE, Apache Sling a Apache Jackrabbit. Postupem času si systém získává velmi důležité zákazníky, jakými jsou McDonald's, Avanza, Daimler, Audi nebo Volkswagen. V roce 2010 byla technologie koupena společností Adobe Systems za 240 miliónů dolarů. [19, 20]

Adobe CQ5 je multiplatformním internetovým systémem umožňující intuitivní správu webových stránek v jednom administrativním rozhraní. Je možné jej spustit na všech obvyklých operačních systémech a internetových prohlížecích. Nejnovější verze byla vydána v roce 2013 a je označena jako Adobe CQ 5.6. Funkčnost je zajišťována prostřednictvím jednotlivých webových konzolí:

- **Websites** pro tvorbu a administraci internetových stránek.
- **Digital Assets** pro ukládání internetového obsahu.
- **Campaigns** pro organizování marketingu a jednotlivých kampaní.
- **Community** pro moderování sociálních sítí.
- **Inbox** pro řízení vstupních položek.
- **Users & Groups** pro řízení uživatelů a skupin v systému.
- **Tools** pro řízení a konfiguraci systému CQ5.
- **Tagging** pro organizování tagů a jmenných prostorů.
- **Workflows** pro modelování a řízení průchodů programem.

- **Reports** pro tvorbu a sledování použití systému.
- **Packages** pro instalaci, řízení a sdílení aplikací.
- **Replication** pro konfiguraci agenta určeného k replikaci.
- **CRXDE Lite** pro vývoj aplikaci přímo v CQ5 IDE.

Dále budou detailněji popsány jen funkcionality WebSites, Digital Assets, Users & Groups, Tools, Workflows, Reports, Packages a CRXDE Lite. [21, 22, 23]



Obrázek 6: Úvodní stránka Adobe CQ5. [23]

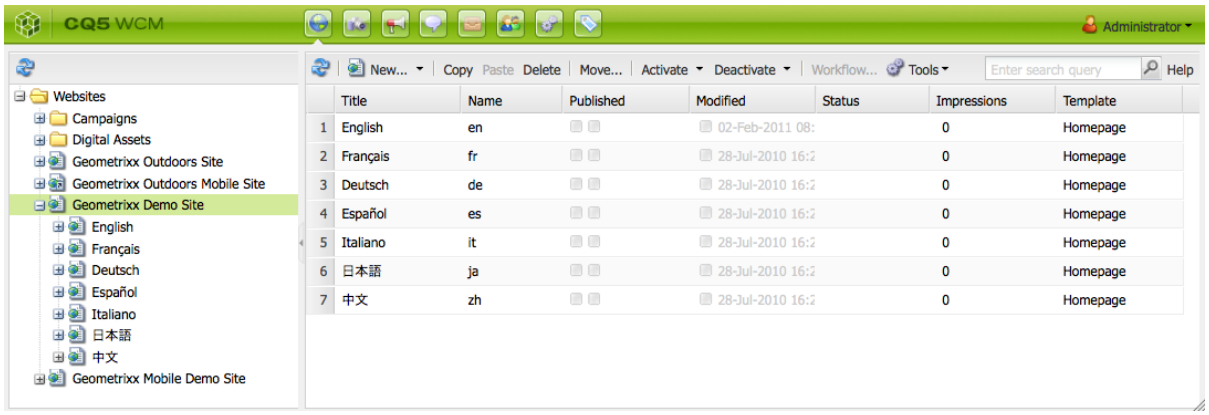
3.5.2 Uživatelské rozhraní Adobe CQ5

V této kapitole budou popsány důležité funkcionality, které systém poskytuje pro správu stránek a obsahu, či pro kvalitnější přehled o systému a vývoji. Tyto funkcionality jsou přímo integrovány do systému Adobe CQ5 a tvoří tak jeden celistvý nástroj.

3.5.2.1 Funkcionalita Websites

Funkcionalita Websites umožňuje vytvářet, spravovat a prohlížet stránky vytvořené v instanci Adobe CQ5. V rámci jednotlivých stránek je možné tvořit podstránky a tím vytvářet vnitřní hierarchii mezi stránkami. Dále jsou poskytovány funkce pro kopírování, přesouvání, publikování (aktivace) a odstranění jednotlivých položek. Dvojitým stiskem je stránka otevřena k náhledu s možnosti přidání jednotlivých komponent. [23]

Při vytváření stránek musí uživatel vyplnit titulek a název stránky. V posledním kroku zvolí jednu z předdefinovaných šablon. Komponenty je možné přidávat pomocí postranního plovoucího panelu technikou „drag and drop“ a jsou děleny na dvě základní množiny. První množina jsou komponenty dodávané standardě s Adobe CQ5 označované „General“, druhá množina zahrnuje komponenty vyvinuté a integrované do systému externě. [23]



Obrázek 7: Funkcionalita pro administraci internetových stránek - Websites. [23]

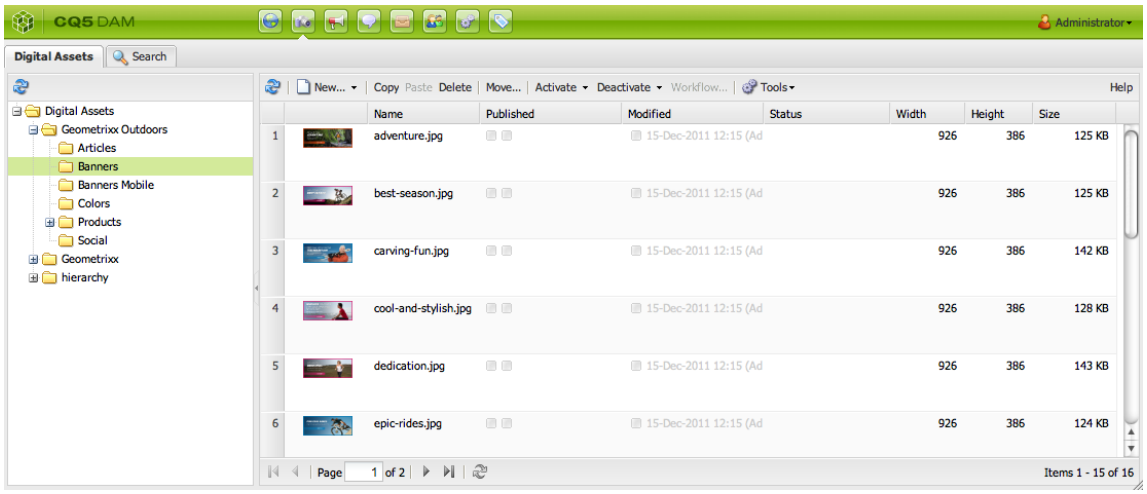
3.5.2.2 Funkcionalita Digital Assets

Tato funkcionalita umožňuje správu uživatelského obsahu jakými jsou dokumenty, audio, video, obrázky, atd. Obsah je možné dále zobrazovat kdekoliv v systému. Jedná se o centralizovaný přístup v uchovávání a poskytování dat v jednotlivých stránkách. Díky tomu jsme schopni mít přehled o obsahu poskytovaném jednotlivým prvkům.

Podporované formáty:

- **Obrázky:** PNG, GIF, TIFF, JPEG, BMP, PNM, PGM, PBM, PPM, PSD, EPS, DNG
- **Dokumenty:** DOC, DOCX, ODT, PDF, HTML, RTF, HTML, RTF, TXT, XLS, XLSX, ODS, PPT, PPTX, ODP, INDD, PS, QXP, EPUB
- **Multimédia:** ACC, MIDI, 3GP, MP3, M4A, MPG, OGA, OGG, RA, WAV, WMA, DVI FLV, M4V, MPEG, OGV, MOV, WMV, SWF
- **Archivační formáty:** TZG, JAR, RAR, TAR, ZIP
- **Jiné:** SVG

Všechny výše uvedené formáty je možné v Adobe CQ5 uložit a dále použít. Mimo této základní funkcionality systém poskytuje u většiny formátů náhledy a informace o metadatech, včetně jejich editace. [23,24]



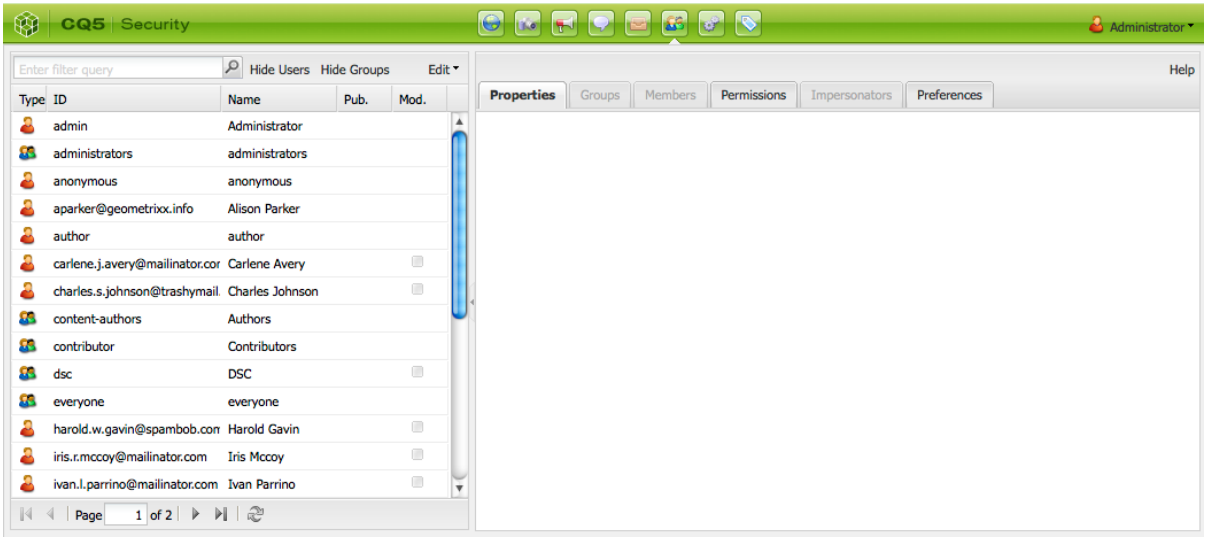
Obrázek 8: Funkcionalita Digital assets. [23]

3.5.2.3 Funkcionalita Users & Groups

V této sekci můžeme provádět administraci uživatelů a skupin. Ty je možné přidávat, odebírat, aktivovat a deaktivovat. Uživatelské rozhraní se dělí vertikálně na dvě úrovně. V levé části jednotlivé skupiny, či uživatele vybíráme a v pravém nahlížíme, případně měníme aktuální nastavení. Nastavení je dále rozděleno na šest horizontálních záložek:

- **Properties** – administrace základního nastavení, jakými jsou login (u skupiny označeno jako ID), jméno, příjmení (pouze pro uživatele), email, popis a cesta.
- **Groups** – záložka zobrazuje skupiny, ve kterých je položka obsažena.
- **Members** – povolené pouze pro skupiny, zobrazuje všechny prvky přímo příslušící dané skupině.
- **Permissions** – administrace práv jednotlivým skupinám a uživatelům do složek uvnitř systému.
- **Impersonators** – záložka je povolena pouze pro uživatele. Funkcionalita umožňuje delegovat pravomoci na jiného uživatele.
- **Preferences** – možnost nastavení předvoleb pro položky, například jazykové preference, atd.

Při vytváření položek je umožněno nastavit jejich příslušnost v rámci hierarchie. Zatímco skupinu i uživatele je možné přiřadit do jakékoliv jiné skupiny, uživatelovi není povoleno dále přiřazovat jiné uživatele, či skupiny. [23, 25]

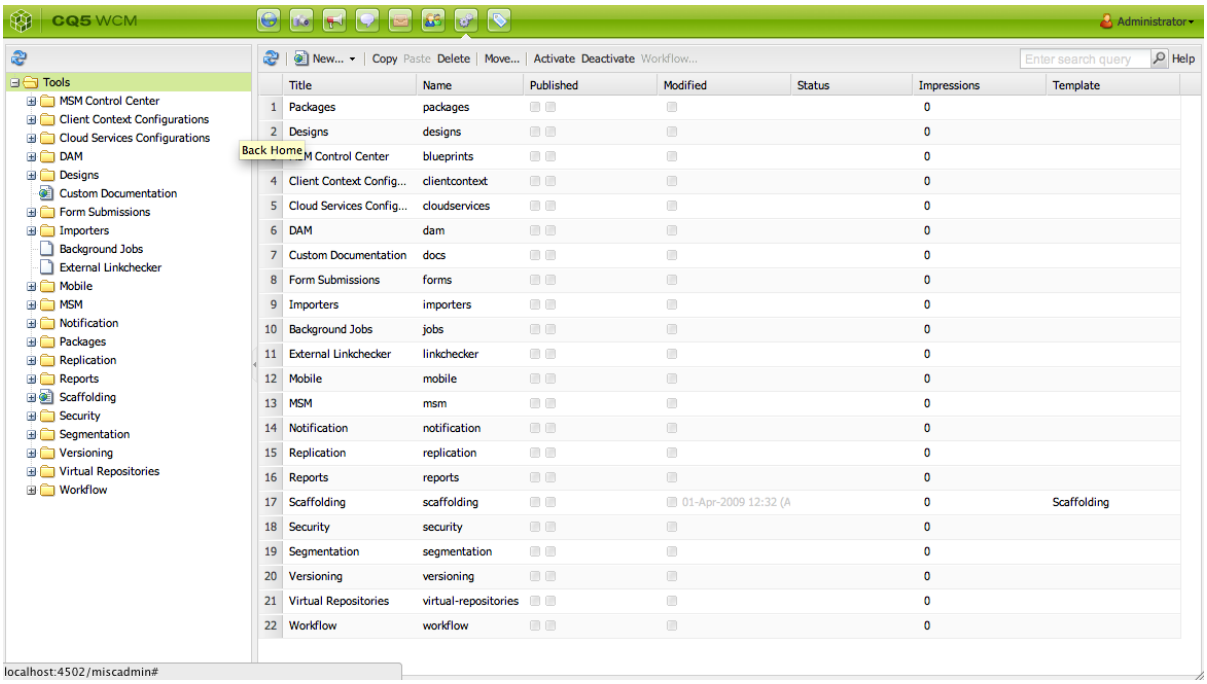


Obrázek 9: Funkcionalita Users & Groups. [25]

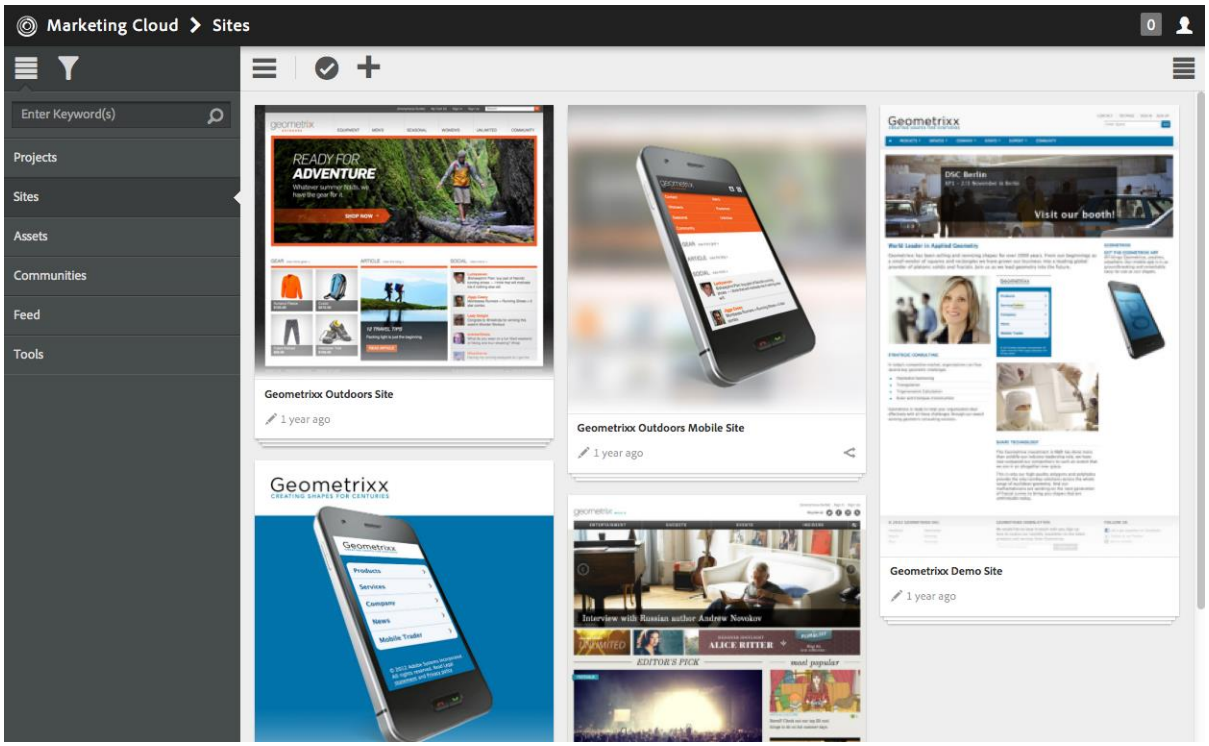
3.5.2.4 Funkcionalita Tools

Funkcionalita Tools integruje dostupné nástroje umožňující komplexnější administraci webových stránek a obsahu dalších prvků v systému. Nástroje jsou členěny na dvě základní podkategorie závislé na použitém uživatelském rozhraní:

- **Tools – UI classic** – nástroje určené pro klasické desktopové uživatelské rozhraní, ovládání je vhodné pomocí myši a klávesnice. Tlačítka jsou malá, nevhodná na dotyk. Můžeme zde zařadit tyto nástroje:
  - **DAM** – funkcionalita byla popsána v kapitole 3.5.2.2
  - **Dashboards** – umožňuje vytvářet panely přizpůsobené na míru
  - **Designs** – spravuje seznam všech grafických šablon, včetně css
  - **Packages** – tento nástroj poskytuje import a export obsahu a nových funkcionalit.
  - **Reports** – nástroj umožňuje široké sledování stavu systémů, je možná uživatelská konfigurace, včetně tvorby vlastních utilit.
  - **Segmentation** – základ pro tvorbu kampaní, pomáhá dělit uživatele dle zájmů, cílů, atd.
  - **Workflow** – postupy definující akce na stránkách nebo ve vloženém obsahu.
- **Tools – Touch-Optimized UI** – vybrané nástroje v této množině jsou optimalizovány na dotykové uživatelské rozhraní. Tlačítka jsou velká a přehledná, je možné je bez větších problémů ovládat dotykem.
  - **Campaigns** – administrace marketingových kampaní
  - **Launches** – správa marketingových aktivit
  - **Task Management** – organizování značek a jmenných prostorů
  - **CRXDE Lite** - nástroj určený pro vývoj, viz. Kapitola 3.5.2.8
  - **Backup** – provádí úlohy zálohování [23]



Obrázek 10: Tools – UI classic. [23]



Obrázek 11: Tools - Touch-Optimized UI. [55]

3.5.2.5 Funkcionalita Workflows

Workflow představuje množinu kroků vedoucích k úspěšnému provedení celkového požadavku, včetně uživatelů a podmínek s nimi souvisejícími. Kroky na sebe postupně navazují, tudíž není možné přejít na následující krok bez dokončení kroku předchozího. V jednoduchosti se jedná o pracovní postup určující logické body ke splnění určitého cíle. V praxi tento nástroj umožňuje

„backendové“ sledování správného fungování aplikace. V případě nesprávného fungování je možné dle nich dohledat chybu. Adobe CQ5 definuje v souvislosti s Workflows následující pojmy: [23]

- **Model** – je tvořen z WorkflowNodes a WorkflowTransitions, přechody jsou spojeny uzly a tím definují průchod. Model má vždy počáteční a koncový uzel.
- **Step** – existuje pět druhů (Participant, Process, Container, OR, AND)
- **Transition** – definuje vztah mezi dvěma po sobě jdoucími kroky, umožňuje užití pravidel
- **WorkItem** – odkazuje na instanci pracovního postupu
- **Payload** – odkazuje na cíl, který bude ve workflow použit
- **Lifecycle** – je rozdělen na Terminate, Suspend, Resume, Restart
- **Inbox** – každý přihlášený uživatel má svůj vlastní workflow inbox

3.5.2.6 Funkcionalita Reports

Tato funkcionalita umožňuje monitorovat a analyzovat systém Adobe CQ5. Administrace nabízí konfiguraci výchozích reportů a umí se tak přizpůsobit aktuálním potřebám firmy. Je zde možné sledovat komponenty, vytížení disku, aktivitu jednotlivých stránek, informace o uživatelích, případně data o workflow. Všechny reporty jsou k dispozici přes systémovou konzoli. [23, 27]

3.5.2.7 Funkcionalita Packages

Funkcionalita „Packages“ povoluje import a export obsahu úložiště. V důsledku to znamená větší možnosti pro kompatibilitu s ostatními systému. Při importu jsou funkce do systému Adobe CQ5 vkládány, v případě exportu jsou poskytovány systémům okolním. „Packages“ je ve skutečnosti ZIP soubor obsahující metainformace, filtry, informace o konfiguraci importu, atd. [23, 26]

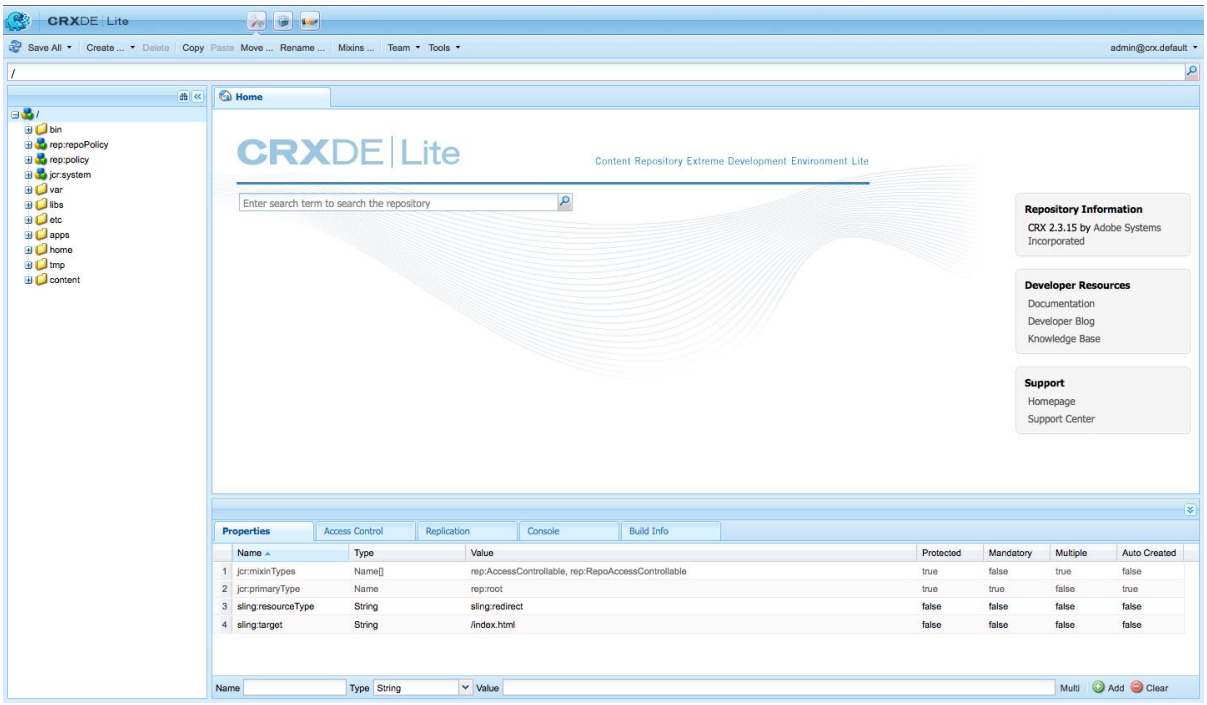
3.5.2.8 Funkcionalita CRXDELite

Pomocí CRXDELite je možné vyvíjet aplikace a komponenty do Adobe CQ5. Nástroj je integrován do systému a poskytuje všechny základní prvky nutné po kvalitní tvorbu programů – vytváření a editace projektů, práce se soubory (.jsp, .java), složkami, šablonami, dialogovými okny, uzly a dalšími. Pro podporu týmové práce je uvnitř zabudován nástroj SVN.

Vizuálně rozdělujeme CRXDELite na čtyři funkční části – lišta menu, textový prostor pro vývoj, zobrazovací panel souborů a složek a dolní panel pro nastavení a výstupy z konzole a kompilátoru. Velikosti jednotlivých oken je možné uživatelsky měnit tahem myši. V případě potřeby je umožněno některá okna úplně skrýt.

Použití tohoto přístupu k vývoji aplikací má své výhody a nevýhody. Mezi výhody patří hardwarová nenáročnost na připojeného klienta. Je možné vyvíjet i na méně výkonných strojích a odkudkoliv, kde máme možnost se připojit na internet. Mezi nevýhody můžeme zahrnout málo prostoru pro vyvíjený zdrojový kód. Velkou část zobrazovací plochy zabírají prvky internetového

prohlížeče a menu CRXDELite. Díky tomu je práce s tímto nástrojem při některých operacích mnohem méně přehledná než u klasických desktopových IDE. [23, 29]



Obrázek 12: Integrované vývojové prostředí CRXDELite. [29]

3.5.3 Struktura Adobe CQ5

Tato kapitola bude popisovat strukturu systému Adobe CQ5 včetně architektury a technologií, které ke své činnosti používá.

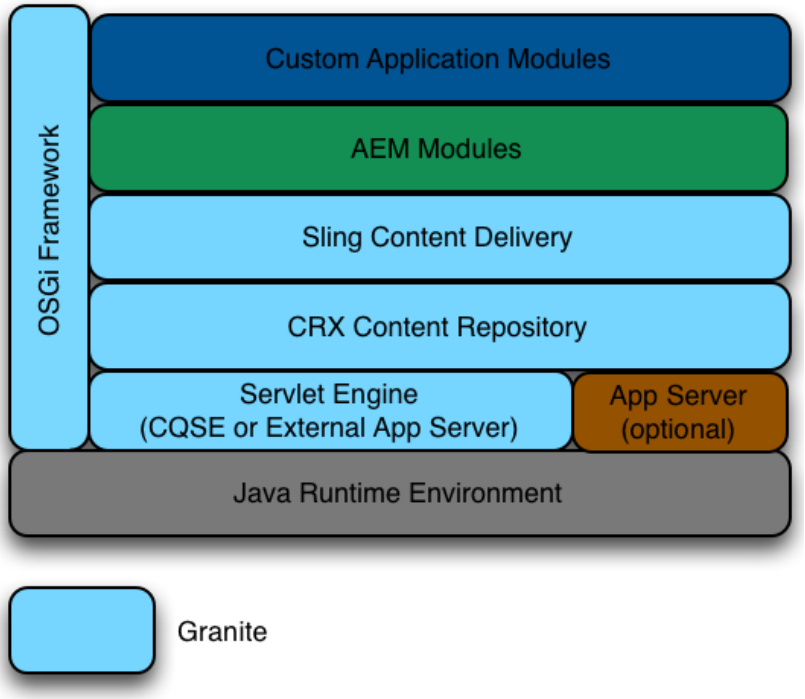
3.5.3.1 Architektura Adobe CQ5

Systém je složen z několika vrstev mezi sebou komunikujících a zajišťujících celkovou robustnost. Každá vrstva má svůj smysl a systému zajišťuje určitou funkcionalitu. Jedná se tyto čtyři základní vrstvy: [31, 30]

- **Java Platform –** doporučená verze je Java Runtime Environment (JRE) 1.7, minimální JRE 1.6.
- **Granite Platform –** vrstva funguje s podporou nižší vrstvy – JRE. Poskytuje služby vrstvám vyšším. Součástí této platformy jsou prvky:
    - **OSGi Service Platform** – založen na Apache Felix, jedná se o specifikaci modulárního systému pro programovací jazyk Java. Aplikace je založena na komponentách (modulech), se kterými je možné za běhu pracovat – instalovat, spouštět, zastavovat, atd. Moduly jsou v OSGi terminologii označovány jako „bundles“.
    - **CQSE Servlet Engine –** zjednodušuje používání servletů uvnitř systému.

- **CRX Content Repository** – všechna data v Adobe CQ5 jsou uložena v tomto uložišti, které je postaveno na JCR (Java Content Repository). Nepoužívá žádnou databázi, či XML soubory.
- **Sling Content Delivery** – jedná se o multiplatformní framework zaměřený pro webové technologie. Je postavený na REST principech umožňujících jednoduchý vývoj obsahově orientovaných aplikací. Obsahově orientovaný (content-centric) způsob znamená transformaci http požadavku, která je v tomto případě mapovaný na obsah. Velkou výhodou aplikací využívající Sling Content Delivey je flexibilnost a dobrá škálovatelnost.
- **Granite UI** – poskytuje UI (User Interface) Framework. Hlavními cíli jsou:
  - Poskytovat UI widgety
  - Realizace uživatelského rozhraní pomocí osvědčených postupů
  - Poskytovat rozšiřitelnost

- **Adobe Experience Manager (CQ5) –** pod tímto bodem je možné si představit veškerou vestavěnou funkcionalitu v systému.
  - Jednotlivé Adobe CQ5 modely (WCM, DAM, Workflow)
- **Customer Applications** – do této kategorie můžeme zařadit aplikace vytvořené uživatelem systému.



Obrázek 13: Architektura Adobe CQ5. [31]

3.5.3.2 Struktura uložiště

V předchozí kapitole jsme uvedli informaci, že Adobe CQ5 pro své potřeby nevyužívá žádnou databázi, ale CRX – Content Repository. Toto uspořádání sebou nese i definovanou strukturu, kterou

je nutné dodržovat. Je možné ji měnit, nicméně to není se strany tvůrců doporučeno. Ve struktuře jsou definovány tyto složky: [32]

- **/apps** – obsahuje definice a zdrojové soubory jednotlivých aplikací.
- **/content** – složka shromažďuje informace o vytvořených stránkách, jejich obsahu a použitích komponentách
- **/etc** – obsahuje inicializační a konfigurační soubory
- **/home** – informace o skupinách a uživatelích
- **/libs** – ve složce jsou uloženy soubory a knihovny pro správnou práci jádra systému, soubory uvnitř složky nesmějí být nijak upravovány
- **/tmp** – složka dočasných souborů
- **/var** – systém zde ukládá změněné nebo upravené soubory, například se může jednat o logy, statistiky. Obsahuje podsložku „classes“ obsahující servlety, kompilované formuláře, atd.

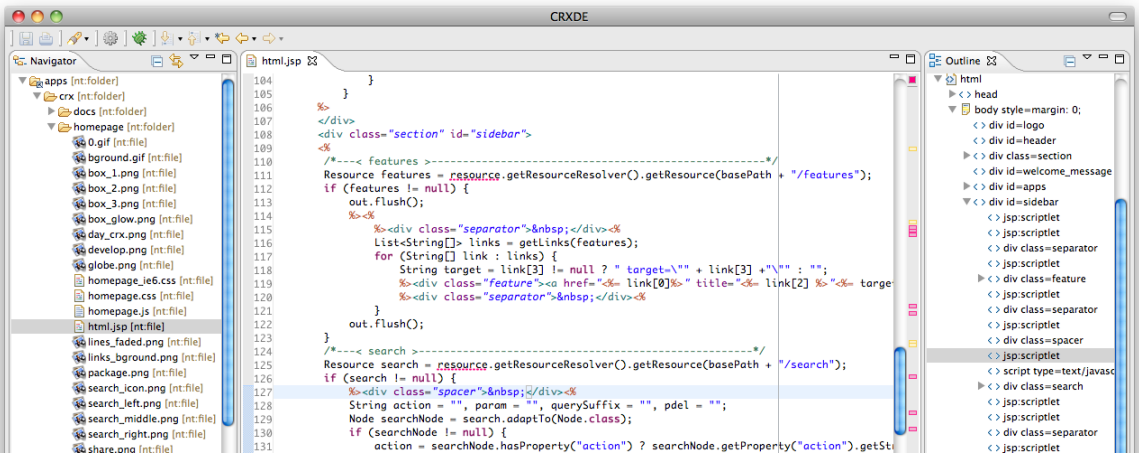
3.5.4 Vývojové prostředí

V této kapitole budou rozebrána externí integrovaná vývojová prostředí. O jednom vývojovém prostředí již byla zmínka v kapitole 3.5.2.8. Jedná se však o vývojové prostředí vložené přímo do systému, proto se o něm v této kapitole nebudeme zmiňovat.

Kvalitní vývoj je možný pouze s kvalitními prostředky, a ty k dispozici pro Adobe CQ5 jsou. Dokonce ve dvou různých variantách – prostředí CRXDE a Eclipse. [38, 33]

3.5.4.1 Prostředí CRXDE

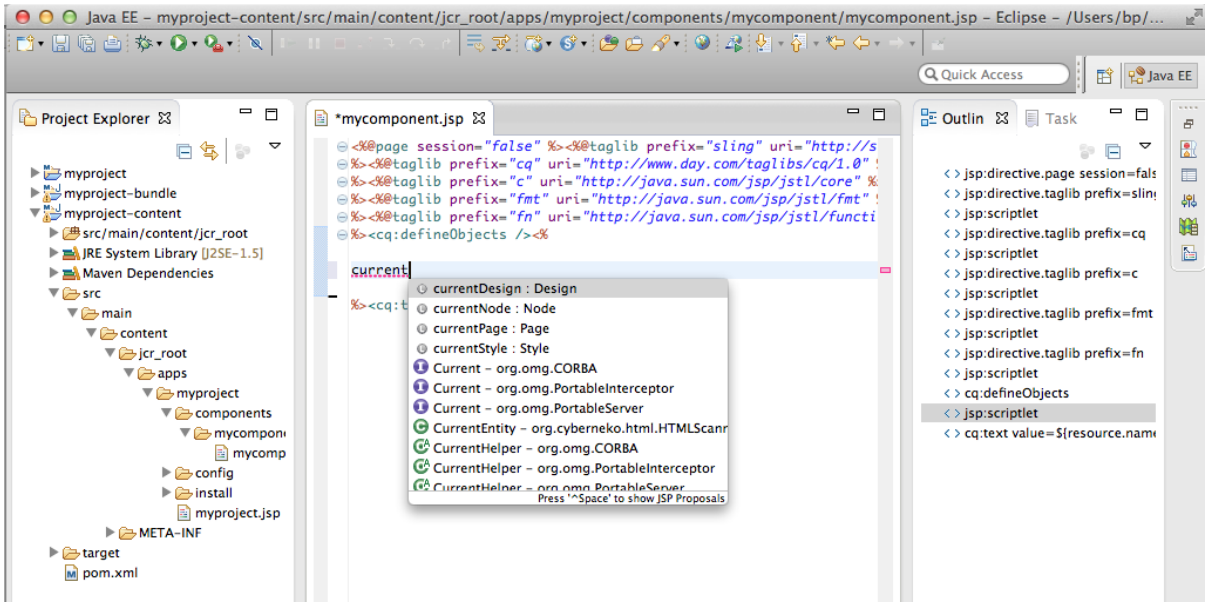
Vývojový nástroj CRXDE je postavený přímo pro vývoj aplikací a komponent pro Adobe CQ5 a poskytuje profesionální množinu funkcionalit ke kvalitnímu vývoji. Umožňuje aplikace vytvářet, upravovat, kompilovat a v případě potřeby i ladit. Veškeré úpravy se ihned projevují na serveru. Tento nástroj je pro vývoj doporučen. [33]



Obrázek 14: Vývoj aplikace pomocí prostředí CRXDE. [33]

3.5.4.2 Prostředí Eclipse

Jedná se o celosvětově známé vývojové prostředí pod „open source“ licenci určené k programování v jazyce Java. Mimo těchto jazyků je možné programovat i v jiných jazycích – PHP, C/C++ a dalších. Mezi konkurenty tohoto prostředí můžeme zařadit Netbeans, IntelliJ Idea a Visual Studio, přičemž poslední dvě jsou komerční a je možné je používat pouze za poplatek. Výhodou užívání Eclipse je jeho velká uživatelská základna a podpora ze strany vydavatele. V porovnání s nástrojem CRXDE nepřináší žádná dramatická vylepšení, či obsáhlejší funkcionalitu. [35, 38]



Obrázek 15: Vývoj aplikace pomocí prostředí Eclipse. [6]

3.5.5 Vývoj komponent pro Adobe CQ5

Nyní se dostáváme k podstatné části práce týkající se struktury a vývoje komponent. Komponenta je velmi obecný pojem znamenající v každém systému něco jiného. V obecné rovině se vždy jedná o jakousi část, kterou je možné do systému importovat a využívat. Ta se pro vnější okolí tváří jako pevná součást systému.

3.5.5.1 Definice komponenty

Komponentou v systému Adobe CQ5 je chápána určitá část aplikace integrována do systému poskytující komplexní funkce okolním subjektům. Je možné ji umístit kamkoliv do stránky, případně použít jako součást jiné komponenty. Komponenta je v dokumentaci definována těmito pravidly: [34]

- Jedná se o modulární jednotku k realizaci určité funkcionality a zobrazení na webových stránkách
- Je znovu použitelné
- Je definována jako samostatná jednotka v rámci jedné složky v uložišti a nemá skryté konfigurační soubory

- Může obsahovat i jiné komponenty
- Komponenta může být spuštěna kdekoliv v systému
- Má standardizované uživatelské rozhraní
- Používá widgety
- Může upravovat chování pomocí konfigurace

3.5.5.2 Vlastnosti komponenty

Komponenta je v systému definována jako typ cq:Component a obsahuje vlastnosti uvedené v tabulce č. 2. [34]

Název artibutu	Typ	Popis
.	cq:Component	Aktuální komponenta. Komponenta je uzel typu cq:Component .
allowedChildren	String[]	Adresa komponent, které mohou být potomky komponenty.
allowedParents	String[]	Adresa komponent, které mohou být předky komponenty.
componentGroup	String	Název skupiny, do které je komponenta zařazena.
cq:cellName	String	Pokud je vyplněno, definuje id buňky
cq:isContainer	Boolean	Určuje, zda se jedná o kontejner.
cq:noDecoration	Boolean	Rozhoduje o designu komponenty, pokud je true, pak není generován s div a css třídami
cq:templatePath	String	Definuje absolutní cestu k šabloně.
dialogPath	String	Definuje cestu k alternativnímu dialogu.
jcr:created	Date	Datum vytvoření komponenty.
jcr:description	String	Popis komponenty.
jcr:title	String	Titulek komponenty.
sling:resourceSuperType	String	Cesta komponenty, od které stávající komponenta dědí.
<breadcrumb.jsp>	nt:file	Soubor skriptu.
design_dialog	cq:Dialog	Definice vzhledu dialogu.
cq:childEditConfig	cq:EditConfig	V případě kontejneru edituje nastavení potomků.
cq:editConfig	cq:EditConfig	Upravuje nastavení komponenty.
cq:htmlTag	nt:unstructured	Vrací přídavné značky přidané do základu HTML značek.
cq:template	nt:unstructured	Pokud je atribut nalezen, pak je použit jako šablona.
dialog	nt:unstructured	Definice úpravy dialogu.
icon.png	nt:file	Ikona komponenty zobrazená v přehledech systému.
thumbnail.png	nt:file	Volitelná ikona zobrazená v přehledech systému
virtual	sling:Folder	Povoluje tvorbu virtuálních komponent.

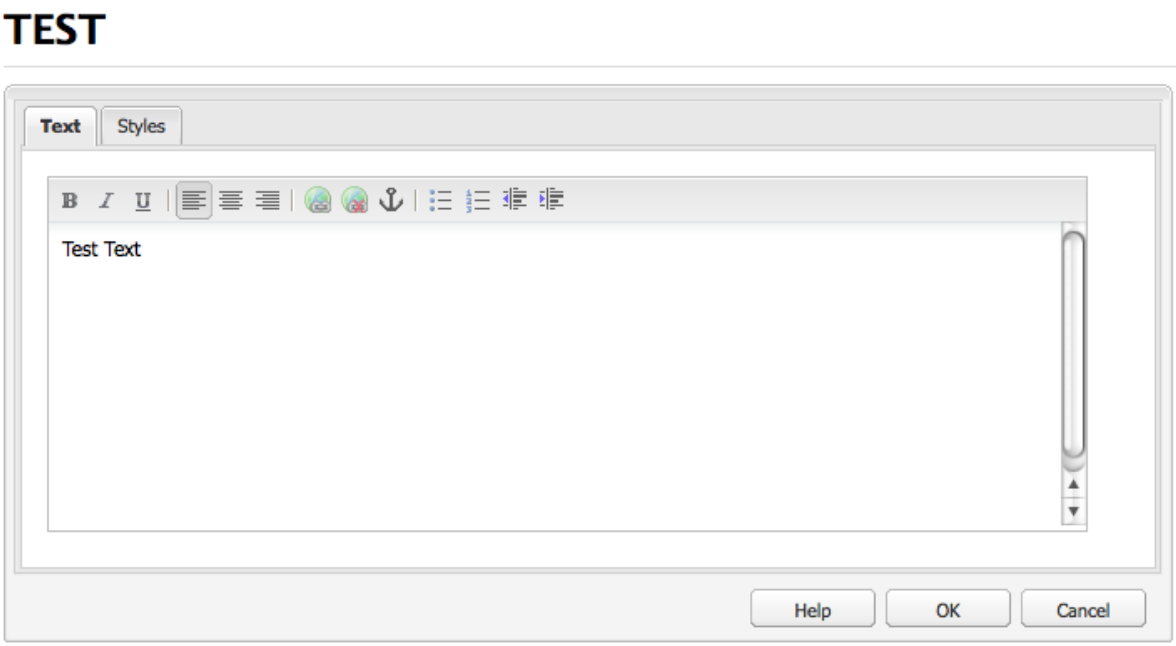
Tabulka 3: Vlastnosti komponenty. [34]

Všechny systémové komponenty jsou založeny na Sling Content Delivery s umístěním ve složce `/libs/foundation/components/`. Ostatní komponenty jsou uloženy ve složce `/apps/<nazevstranky>/components/`. Každá komponenta musí mít definovány alespoň povinné atributy poskytující základní informace. Mezi tyto atributy patří: [34]

- **Seznam jcr properties:** určuje základní informace o komponentě a jejími spolupracující subjekty.
- **Dialog komponenty:** umožňuje konfiguraci komponenty.
- **Cq:editConfig:** definuje editaci vlastností komponenty.
- **Zdroje:** určuje všechny prvky, které jsou komponentou použity.
- **Vnitřní logika:** obsahuje skripty a zdrojové soubory definující vnitřní logiku a funkcionalitu komponenty.
- **Thumbnail:** ikona zobrazující se uvnitř systému.

3.5.5.3 Dialog komponenty

Dialog je povinným prvkem všech komponent, bez kterých nemohou pracovat. Díky tomu je možné uživatelsky přizpůsobit chování k požadovaným potřebám. Dialog poskytuje tvorbu uživatelských vstupů pro vkládání textů, závislostí jiných komponent, atd. V případě potřeby definování velkého počtu vstupních informací a nastavení dialog umožňuje tvorbu jednotlivých záložek. Dialog je v systému definován jako typ `cq:Dialog`. [34]



Obrázek 16: Dialog komponenty. [34]

4 Analýza současného stavu

Tato kapitola se bude zabývat analýzou současného stavu firmy Hartmann – RICO a.s. a systému Adobe CQ5. V první polovině se zaměříme na popis a analýzu společnosti z pohledu tržního postavení a rozebereme současný vývoj komponent ve firmě. V té se budeme zaměřovat na jednotlivé etapy vývoje, využitou architekturu a porovnávání systému Adobe CQ5 s konkurenčními nástroji. V úplném závěru zacílíme na sumarizaci provedených analýz a vyvodíme z nich příslušné závěry.

4.1 Analýza společnosti Hartmann – RICO a.s.

Firmu Hartmann – RICO a.s. budeme analyzovat ze dvou směrů. První oblast se bude zabývat popisem firmy a jejím konkurenčním postavením. Ve druhé oblasti provedeme analýzu IT oddělení a stávajících procesů pro vývoj komponent.

4.1.1 Popis společnosti

Společnost Hartmann – RICO a.s. se specializuje na výrobu a distribuci zdravotnických a hygienických prostředků v České republice. Jedná se o stabilní firmu s dlouholetou historií. První závod byl založen více jak před sto lety. Výroba začala v roce 1891 v Chomutově pod názvem Richter&Compagnon. Odtud je odvozen název RICO (Richter, Kohn), který si společnost udržela až do roku 1991. V roce 1991 přichází do firmy mezinárodní korporace Paul Hartmann, přední světový výrobce zdravotnických prostředků a hygienických výrobků, a firmu přejmenovala na Hartmann – RICO a.s. [40, 41]

Společnosti se daří, každým rokem roste její celkový obrat. Od roku 1993 jej zvýšila více jak pětinásobně. Export na cizí trhy pak vzrostl téměř dvacetinásobně a v posledních letech dosáhl skoro dvě a půl miliardy korun. [41]

Výrobní závody Hartmann – RICO a.s.: [42]

- Veverská Bítýška
- Havlíčkův Brod
- Chvalkovice

Společnost je držitelem certifikátu ISO 9001, ISO 13 485, EN 550. Každoročně se umísťuje v soutěžích Zlatý středník a Exportér roku. [43, 44]

4.1.1.1 Tržní postavení společnosti

Při sestavování GE matice bylo přihlíženo ke vlivům přímo se vztahujícím ke společnosti. Přesně se jedná o tyto vlivy:

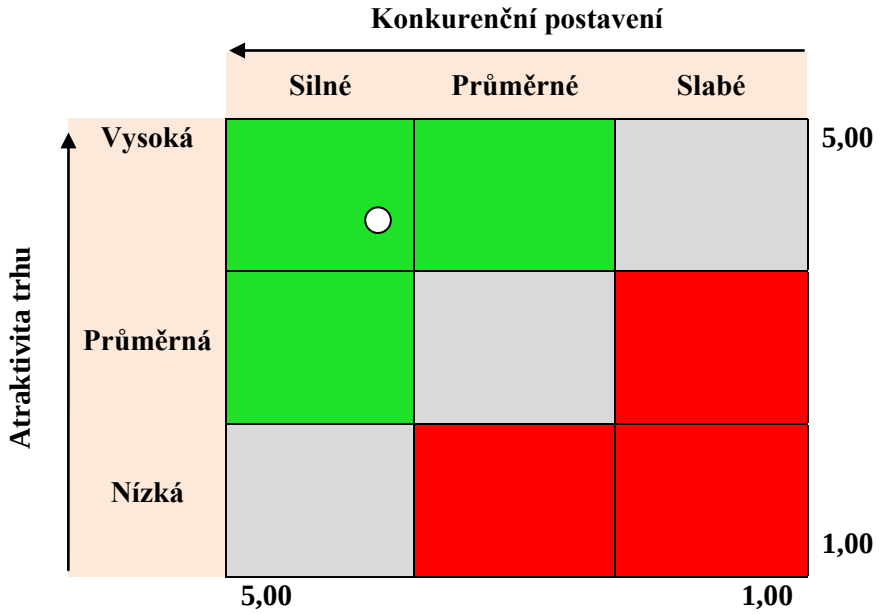
- **Atraktivita trhu:**
  - Velikost trhu
  - Roční tempo růstu
  - Charakter konkurence
  - Stabilita na ekonomickém trhu
  - Technologický vývoj, náročnost produkce
  - Rentabilita podnikání
- **Konkurenční postavení:**
  - Relativní tržní podíl na trhu
  - Relativní výrobní kapacita
  - Relativní inovační potenciál
  - Distribuční schopnosti
  - Marketingový potenciál
  - Ziskovost
  - Zkušenost a schopnost vedení

Vlivy na atraktivitu trhu	Váha	Hodnocení	Součin
Velikost trhu	0,3	5	1,5
Roční tempo růstu	0,1	4	0,4
Charakter konkurence	0,1	2	0,2
Stabilita na ekonomickém trhu	0,2	4	0,8
Technologický vývoj, náročnost produkce	0,1	3	0,3
Rentabilita podnikání	0,2	4	0,8
Celkový vliv			**4**

Tabulka 4: Vyhodnocení vlivů na atraktivitu trhu

Vlivy na konkurenční postavení	Váha	Hodnocení	Součin
Relativní tržní podíl na trhu	0,3	5	1,5
Relativní výrobní kapacita	0,1	3	0,3
Relativní inovační potenciál	0,1	2	0,2
Distribuční schopnosti	0,2	4	0,8
Marketingový potenciál	0,1	4	0,4
Ziskovost	0,1	3	0,3
Zkušenost a schopnost vedení	0,1	2	0,2
Celkový vliv			**3,7**

Tabulka 5: Vyhodnocení vlivů na konkurenční prostředí.



Tabulka 6: Tržní postavení společnosti v konkurenčním prostředí.

Díky této analýze jsme zjistili, že firma má na trhu velmi silné postavení. Díky vysoké atraktivitě trhu je pro ni důležité chránit si tuto pozici před novými i stávajícími konkurenty.

4.1.2 Analýza současného vývoje komponent pro systém Adobe CQ5

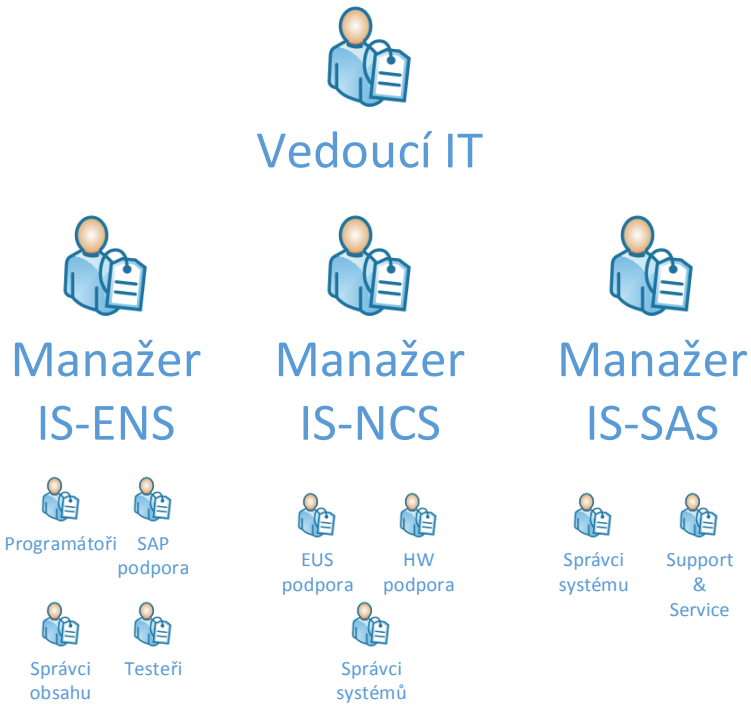
Cílem této kapitoly je zjištění aktuálních procesů ve společnosti Hartmann – RICO a.s., respektive uvnitř IT oddělení. V první části se detailněji zaměříme na samotné IT oddělení, jeho strukturu a postavení. V druhé polovině bude analyzován současný technologický postup vývoje komponent, odpovědnosti za vývoj, využitou architekturu a tvorbu dokumentací.

4.1.2.1 Analýza IT oddělení ve společnosti

Společnost Hartmann – RICO a.s. má jednotné IT oddělení pro všechny jednotky uvnitř firmy. Oddělení vyvíjí aplikace pro závody ve Veverské Bítýšce, Havlíčkově Brodě a Chvalkovicích. Dále tvoří support pro Slovenskou republiku a Bulharsko.

Struktura IT oddělení

IT oddělení je tvořeno zhruba 20 pracovníky, z tohoto počtu zhruba 6 osob vyvíjí software. Struktura oddělení je hierarchická. V jejím čele stojí vedoucí IT pracovník mající pod sebou tři manažery, kteří tvoří prostřední vrstvu hierarchie mezi pracovníky a vedoucím IT oddělení. Každý vedoucí má svůj vlastní tým tvořený šesti až osmi členy složený z kvalifikačně odpovídajících pracovníků. Sktruktura je názorněji zobrazena na obrázku č. 17. Sídlem IT oddělení je závod ve Veverské Bítýšce.



Obrázek 17: Organizační struktura IT oddělení.

Postavení IT oddělení uvnitř společnosti

Oddělení IT tvoří ve společnosti samostatnou jednotku s vlastním ročním rozpočtem, jehož základním cílem je zajištění technických prvků v rámci firmy. Na základě požadavků ostatních oddělení jsou zpravovány projekty usnadňující fungování interních procesů. Tato spolupráce funguje na základě obchodních pravidel. Objednávající strana většinou projekty, které IT oddělení zadá, následné financuje. Další nezanedbatelnou aktivitou oddělení je podpora pro velké firemní klienty, kterými jsou nemocnice a lékárny. Tato skupina projektů tvoří zhruba 40% – 50% všech vytvořených aplikací. V průběhu několika let chce tato čísla společnost Hartmann – RICO a.s. dále navyšovat.

Projektové řízení vývoje

V rámci vývoje aplikací je použit pro projektové řízení nástroj The Domino Project od společnosti IBM, který je součástí softwarového programu Lotus Notes. Tento balík obsahuje základní nástroje pro řízení zdrojů, vedení týmu, tvorby projektové dokumentace, tvorby workflow nebo modul pro řízení času. Vedoucí pracovníci jej používají pro velké projekty. Při tvorbě malých komponent je tento nástroj nevyužit. [46]

Interní pravidla pro vývoj – „code-rules“

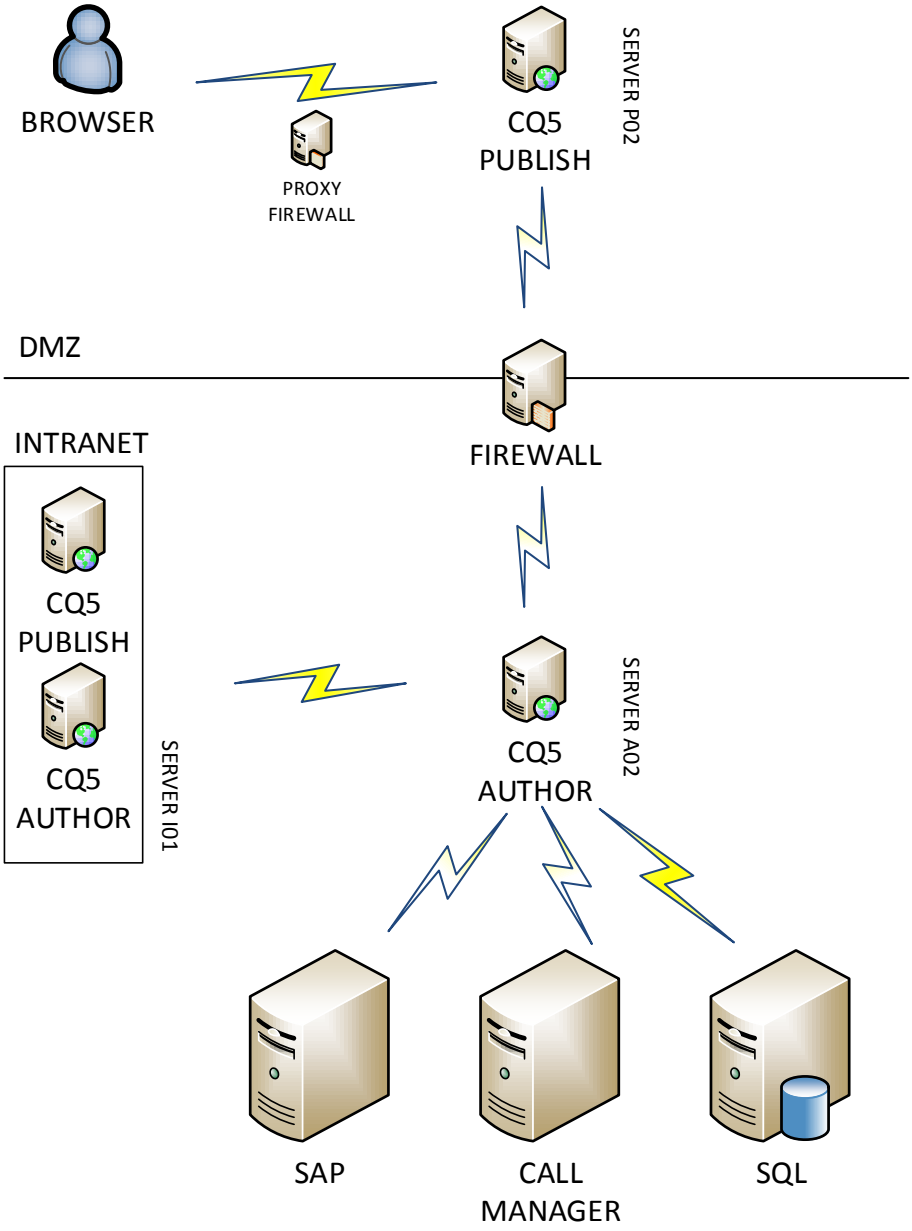
IT oddělení nemá vytvořená žádná závazná pravidla pro vývoj komponent a samotný vývoj obecně. Řídí se pouze obecnými pravidly dostupnými na internetu.[1] Nicméně i přesto jsou zdrojové texty a pojmenování databází v některých částech nekonzistentní. U databází je to projevuje libovolným pojmenováním atributů. V některých případech jsou atributy pojmenovány anglicky, jindy česky.

[1] Oracle na svých webových stránkách uveřejňuje konvence pro tvorbu zdrojových kódů v programovacím jazyce java. Zdroj je dostupný na adrese http://www.oracle.com/technetwork/java/codeconv-138413.html.

Mezi slovy je někdy použito podtržítko - „_“, jindy jsou slova napsaná dohromady. U tříd se například jedná o definování velkým a malých písmen u atributů, názvů tříd nebo metod, případně definování konstant, atd. Interní pravidla pomáhají definovat týmovou kulturu při tvorbě informačních projektů. Proto je důležité je definovat a stanovit jim patřičnou závaznost.

4.1.2.2 Architektura systému

Výsledná architektura systému je tvořena několika servery. Většinou se jedná o servery databázové, webové, případně o proxy servery. Cílem této struktury je rozložení zátěže a zkvalitnění bezpečnosti pro jednotlivé instance daného systému Adobe CQ5. Některé instance jsou použity pro vývoj, jiné pro produkční nasazení. V obrázku jsou uvedeny pouze servery pro produkční využití, servery určené pro vývoj a testování zde uvedeny nejsou.



Obrázek 18: Zjednodušené schéma produkční architektury.

Instance každého systému Adobe CQ5 běží souběžně ve dvou modech:

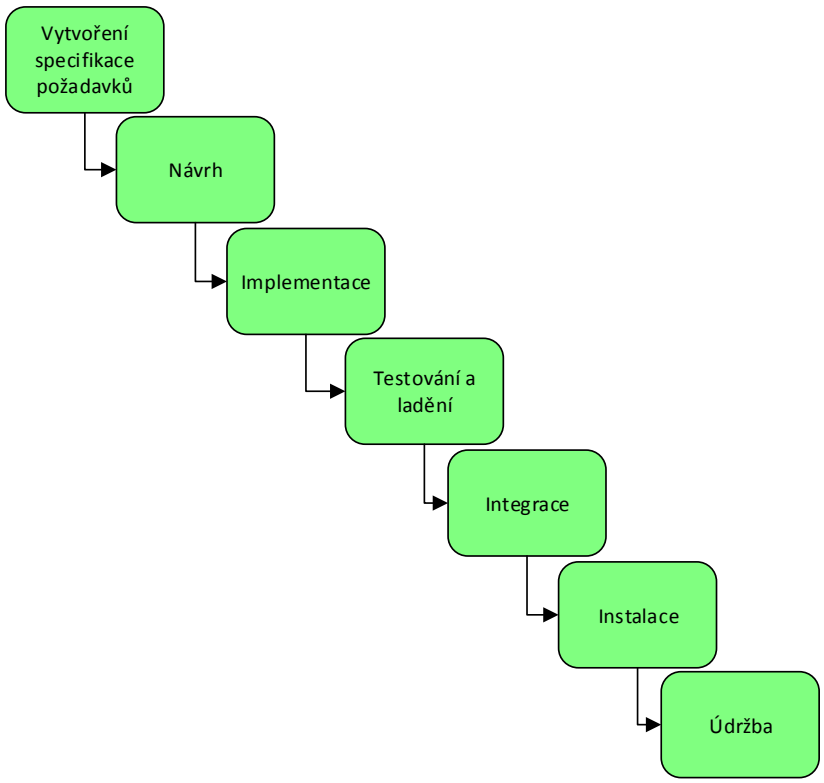
- **Author** – spuštěn na portu 4502, přístupný pouze z interní sítě, určený pro vývoj a správu dat.
- **Publish** – spuštěn na portu 4503, určen pro produkční zobrazení dat, data jsou na něj replikována z author modu.

4.1.2.3 Vývoj komponent pro systém Adobe CQ5

Základní náplní IT oddělení je vývoj komponent sloužících jako podpůrný prostředek pro firemní procesy. V běžné praxi jsou dva základní zadavatelé – interní a externí. Rozdíl v nasazení aplikace pro oba zadavatele není téměř žádný. Obě vytvořené aplikace výsledně běží na serverech ve Veverské Bítýšce a jsou k dispozici přes internetové připojení.

Etapy vývoje komponent

Vývoj komponent je tvořen ze sedmi po sobě následujících procesů. Prvním procesem je samotné vytvoření zadání a specifikace v textové formě, která je vytvořena zadávající stranou. Dalším procesem je tvorba návrhu. Na tomto bodu se podílejí obě strany – jak zadavatel, tak dodavatel. Jakmile je návrh hotov, následuje část implementace. Za tuto a následující části plně zodpovídá dodavatel aplikace. Po dokončení implementační části je otestována funkčnost a opraveny případné chyby. V posledních třech krocích je aplikace převedena na produkční server, spuštěna a udržována ve funkčním stavu. Model vývoje je shodný s vodopádovým modelem pro vývoj softwaru a je vhodný pro malé a nenáročné projekty. Názorněji je proces vývoje zobrazen na obrázku č. 19. [45]



Obrázek 19: Etapy vývoje komponent.

Odpovědnosti za vývoj komponent

Za každý proces při vývoji komponent je zodpovědný určitý pracovník. Ten však provedení zadaného procesu, případně podprocesu může delegovat někomu jinému. Za vytvoření specifikace je zodpovědný zadavatel. Ten však určité podrobnosti může konzultovat s manažerem oddělení. Jakmile jsou požadavky vytvořeny, je informován manažer, který již dále dopovídá za vedení dalších procesů. V návrhu jsou zapojeny zaměstnanci ze zadavatelského i dodavatelského oddělení. Drobné změny v návrhu je možné upravovat během dalších procesů – implementace a testování a ladění aplikace. Implementaci provádí programátoři dle předložené specifikace a návrhu. Po dokončení procesu je provedeno testování aplikace. Aplikaci testuje vývojář, interní tester a zadavatel. Jakmile je program odladěn a schválen zadavatelem, je možné ho nasadit na produkční servery. Zadavatel a vedoucí IT jsou o všech procesech, kterých se přímo neúčastní, informováni.

	Vedoucí IT	Manažer	Programátor	Tester	HW support	Zadavatel
Vytvoření požadavků	I	IC	-	-	-	AR
Návrh	I	AR	C	-	-	CI
Implementace	I	AC	R	-	-	CI
Testovaní a ladění	I	AC	IC	R	-	CRI
Integrace	I	AC	R	-	R	I
Instalace	I	AC	R	-	R	I
Údržba	I	AC	R	-	R	I

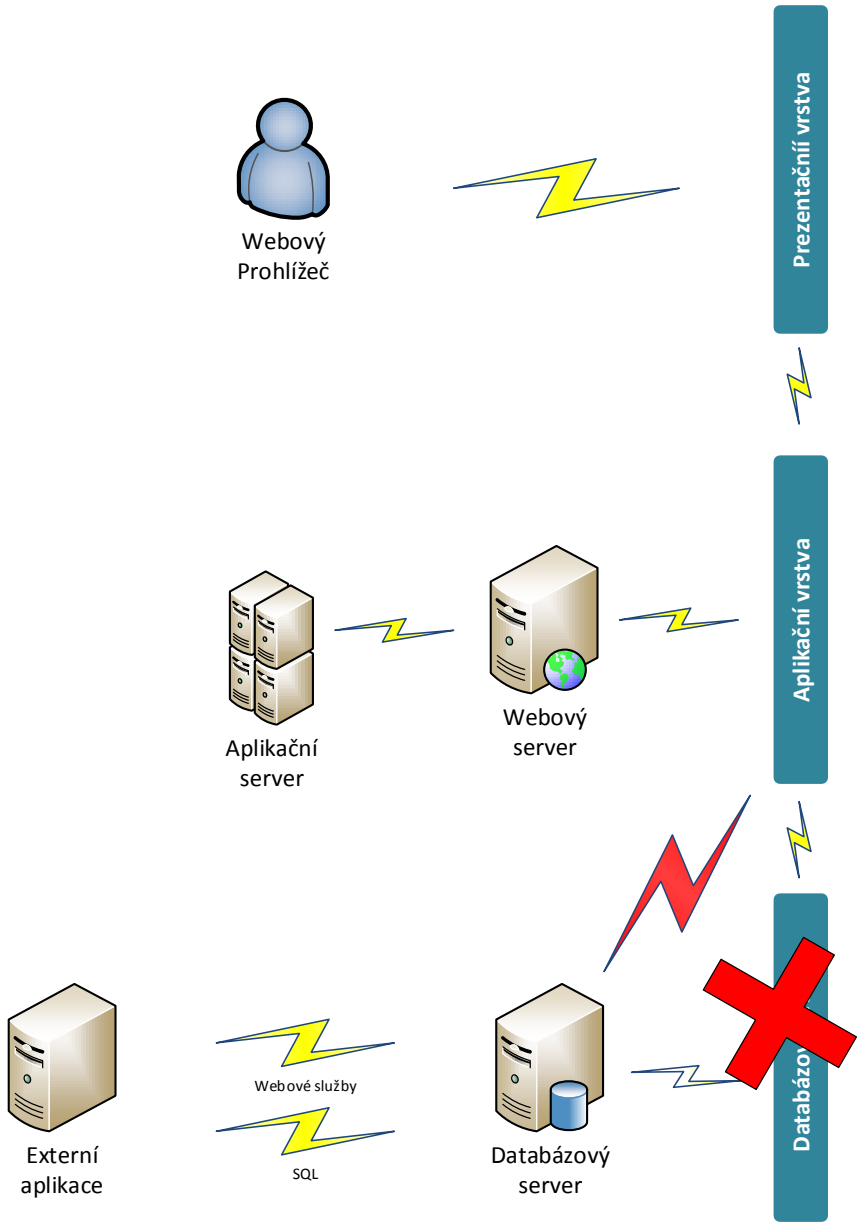
Tabulka 7: RACI matice odpovědností jednotlivých etap vývoje.

Architektura vrstev v komponentách

Jedná se o základní schéma při tvorbě webových aplikací. Je tvořena ze třech vrstev – prezentační, aplikační a databázové. Pro náročnější aplikace je možné ji o další potřebné vrstvy rozšířit. Základním smyslem je rozdělení odpovědností mezi více prvků – vrstev mezi sebou spolupracujících. Úkoly jednotlivých vrstev:

- **Prezentační:** jejím cílem je zobrazování dat uživateli, v komponentách v Adobe CQ5 je tato technologie zajištěná pomocí JavaServer Pages (JSP). V nich je možné definovat strukturu stránky pomocí HTML značek atd. Komunikuje pouze mezi uživatelem a aplikační vrstvou. Vrstva nekomunikuje přímo s datovou vrstvou.
- **Aplikační:** Aplikační vrstva leží mezi vrstvou datovou a prezentační. Řídí sémanticky průběh vykonávaného programu. V systému je tvořena třídami, případně servlety. Ty mohou rozhodovat, jaká data se odešlou prezentační vrstvě a jak se zobrazí. Dále čeká na aktivity přicházející z prezentační vrstvy a ty pak dále zpracovává.

- **Databázová:** Tato vrstva se stará o získávání dat z konkrétní databáze a poskytuje je vyšší aplikační vrstvě. V současném systému vrstva zcela chybí. Aplikační vrstva získává data přímo z databáze. Zmíněné provedení sebou přináší několik negativních jevů. Prvním jevem je, že pro opakované získávání dat v různých komponentách je duplikován zdrojových kód, který se postupných zvětšováním systému stává hůře udržovatelným. Taková struktura postupem času prodražuje celkový vývoj. Ten je prodražován podstatnou měrou, neboť je při vstupech do databáze nutné odchytávat výjimky apod. Dalším velkým problémem je velká závislost mezi aplikační vrstvou a databází, kterou bude v budoucnu obtížné upravovat, případně provádět refaktoring některých tabulek a atributů. Nežádoucí vazba mezi aplikační vrstvou a databázovým serverem je tvořena jakýmkoliv přímým databázovým příkazem.



Obrázek 20: Schéma nedokonale použité třívrstvé architektury.

Využití návrhových vzorů

Návrhové vzory jsou dnes zabudovány do všech velkých systémů. Ty ve většině případů nutí dané programátory a uživatele tyto metodiky používat, aniž by o nich měli jakékoliv znalosti. Tento případ nastává i u systému Adobe CQ5. V jádru je postaven na použití výše zmíněného vrstveného modelu, další návrhové vzory není problematické v případě potřeby použít. Použití dalších návrhových vzorů ve stávajících aplikací nebylo nalezeno.

Refaktoring

Jedná se o informatickou disciplínu provádění částečných změn ve zdrojovém kódu takovým způsobem, který nemění funkcionalitu vůči okolí. Při změnách se mění jednotlivé konstrukce programu a nahrazují se za sémanticky méně složité, případně efektivnější. Výsledný efekt má za následek lepší udržitelnost vývoje. Refaktoring se zaměřuje také na úpravu architektury jednotlivých částí a snaží se daný úsek zjednodušit. Při provedení změn v architektuře se může jednat o sémantické změny – převádění jednotlivých metod mezi třídami. Tyto změny pak obecněji zajišťují intuitivnější vývoj.

Ve společnosti se refaktoring neprovádí. Je to částečně dáno skutečností, že jsou vytvářeny pouze jednoduché komponenty obsahující pouze několik stovek až tisíců řádků zdrojového kódu. V případě tvorby více vytěžovaných vrstev je pak refaktoring nutný. V tomto případě se nejedná o žádný velký problém. [48]

Tvorba dokumentací

Tvorba dokumentace je velmi důležitá, ale zároveň také podceňovaná aktivita. Slouží jako jeden z nástrojů pro dobrou spolupráci členů v jednom týmu. Každý tímto způsobem může sdělit svoje poznatky, případně informace o právě provedené práci. Dokumentace jsou zpětně dohledatelné a umožňují uchování znalostí v případech pracovní migrace.

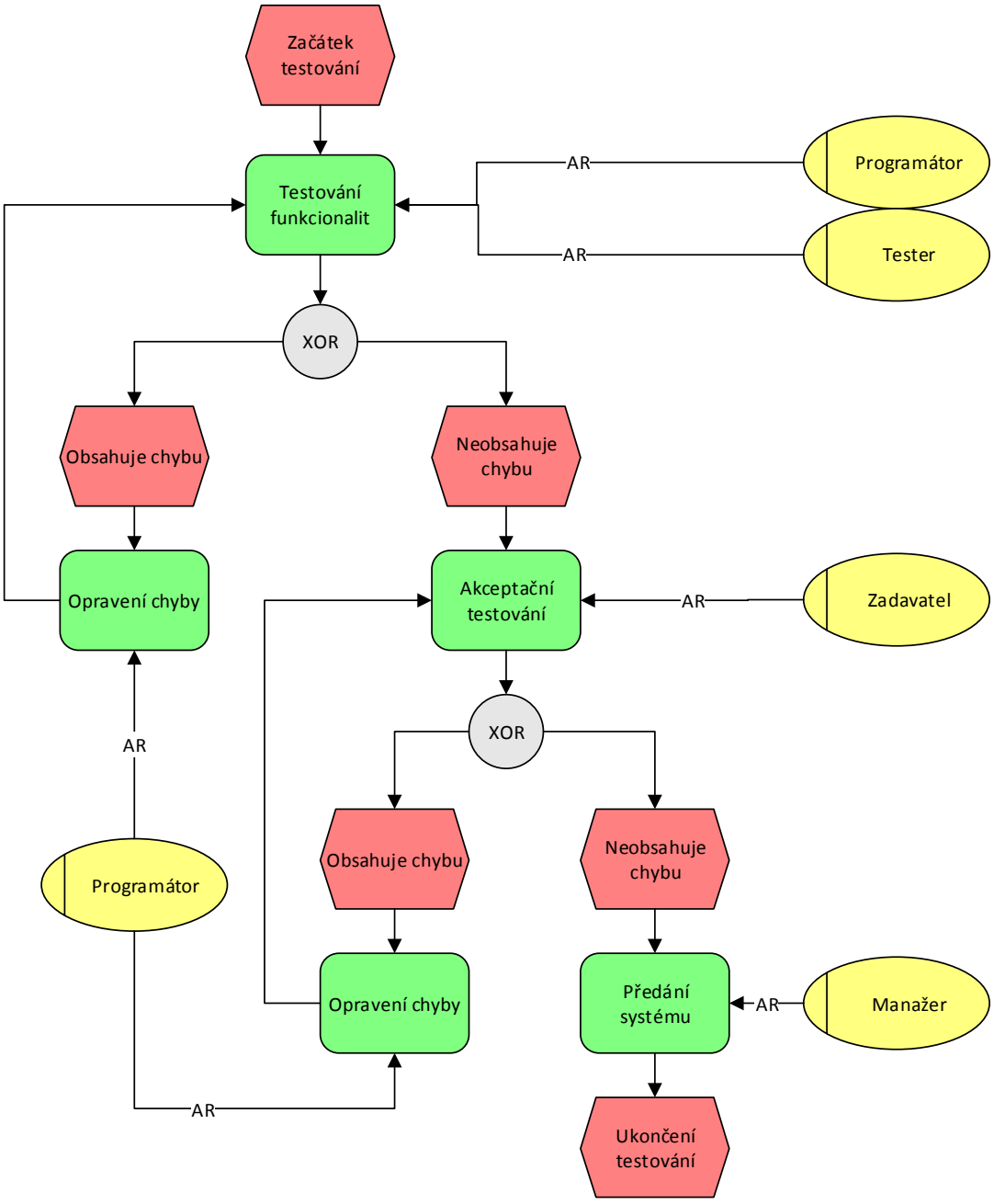
V rámci vývoje jsou tvořeny tyto dokumentace:

- **JavaDoc** – slouží k dokumentaci jednotlivých prvků v programovacím jazyce Java. Prvky jsou myšleny třídy, rozhraní, metody, případně přístupně atributy a konstanty. Pro programátory je to velmi užitečný nástroj. Komentáře jsou v aplikacích tvořeny z 10-20%, což je velmi málo.
- **Dokumenty na sdíleném disku:** Všichni zaměstnanci IT oddělení mají přístup ke sdílenému disku, na kterém jsou uloženy důležité dokumenty a informace.

Testování

Tato činnost slouží k odhalení chyb ve vyvinutých programech. První etapa testování je zahájena ihned po dokončení vývoje aplikace a provádí ji sám programátor. Jakmile program nevykazuje chyby, provede další testování tester. Pokud program vykazuje chybovost, jsou chyby reportovány

zpět programátorovy. Ve stavu bezchybnosti následují poslední akceptační testy, které provádí samotný zákazník. Dle poskytnutých materiálů je ve firmě používán tiketovací systém pro zadávání nalezených chyb při testování. Ten slouží pro podporu procesního řízení objevených chyb, které je možné po zadání do systému přiřadit libovolné osobě a případně jim nastavovat různé stavy. Umožňuje získání přehledu o všech zadaných chybách, vyřešených opravách připravených k nasazení a následných statistikách.



Obrázek 21: EPC diagram pro testovací proces.

4.2 Analýza systému Adobe CQ5

V předchozí kapitole jsme se pokusili analyzovat IT oddělení a průběh jednotlivých etap vývoje komponent. Tato kapitola bude zaměřena na porovnání Adobe CQ5 s jinými systémy a definování celkového postavení na trhu. U každého systému budou určeny silné a slabé stránky. Zkusíme postupně definovat kvalitu prostředí pro vývoj, včetně dostupných vývojových nástrojů, dokumentace a komunity.

4.2.1 Porovnání Adobe CQ5 s konkurencí

Tuto problematiku řešilo vedení společnosti Hartmann – RICO a.s. v době, kdy chtělo vybrat nový systém pro firemní použití. Vybíralo z početné množiny proprietárních systému velkých firem, jakými například jsou firmy IBM, Adobe, HP, Oracle, Microsoft a další. Některé s těchto systému se pokusíme analyzovat a srovnat a vybraným systém Adobe CQ5. [47]

4.2.1.1 OpenText

Systém byl vyvinut v Kanadě a nabízí širokou informační podporu pro podnikové procesy. Web Experience Management (WEM) je nabízený softwarový balík známý pod názvem „Customer Experience Management“ zahrnující správu komunikace se zákazníky, správu digitálních aktiv, podporu sociálních komunit, atd. [47]

Silné stránky

Systém je komponentově orientován, tzn., že vnitřní prvky je možné skládat z jednotlivých komponent. Novější verze mají vylepšené uživatelské rozhraní umožňující in-line editaci obsahu. Má pevné vztahy s firmami Microsoft, Oracle a SAP, což zajišťuje plnou integritu jejich systému. [47]

Slabé stránky

Dle společnosti Gartner OpenText ztratil v posledních letech dobré postavení na trhu a nedokáže konkurovat silnějším firmám. Ty dokáží lépe prosazovat svá řešení, navzdory tomu, že jejich systémy většinou nemají tak moderní vzhled jako jejich menší konkurenti. [47]

4.2.1.2 Sitecore

Firma, který vyvinula tento systém má sídlo v Kodani v Dánsku. Nasazení do firemního prostředí je vhodnější spíše pro menší společnosti. Vlajkový produkt je založen na technologii .NET a je označován jako Sitecore 6.6. [47]

Silné stránky

Z velké části je zaměřen na digitální marketing. V roce 2012 zaznamenala firma velký nárůst zájmu nových zákazníků. Největších úspěchů dosahuje v asijských zemích, především v Japonsku a Číně. Sitecore dokáže elegantně řešit vícejazyčné nasazení. [47]

Slabé stránky

SiteCore je sice velmi dobře odladěný a spolehlivě funkční, nicméně společnost není schopna flexibilně reagovat na aktuální potřeby zákazníků. Je vhodná pro komunikaci s technologií SharePoint, pro zákazníky používající platformu JAVA a jiné již tak vhodná být nemusí. [47]

4.2.1.3 SDL Tridion

Společnost SDL byla založena v Maidenheadu ve Velké Británii. Systém je zaměřen na marketing a e-commerce řešení. Společnost zavedla možnost přidávání systémových modulů, od kterých očekává zvýšení zájmu na obchodním trhu. [47]

Silné stránky

Jedná se o dobře udělaný systém prezentující se velkou celistvostí, který je možné přizpůsobit svému obchodnímu odvětví. V posledních letech firma provádí chytrou obchodní a marketingovou politiku podpořenou několika strategickými akvizicemi. [47]

Slabé stránky

Systém zaostává vůči konkurenčním řešením v oblasti mobilní strategie. Odstranění tohoto nedostatku může nějaký čas trvat. Podobně jako SiteCore je vytvořen na platformě .NET. Podle informací Gartner často figuruje v širším výběrovém řízení, velmi málo se však dostane do výběru užšího. [47]

4.2.1.4 Oracle WebCenter Sites

Společnost Oracle, sídlící ve státě Kalifornia ve Spojených státech amerických, získala WCM produkt převzetím společnosti FatWire Software v roce 2011. Hlavním zaměřením systému je digitální marketing. Koncepčně je rozdělen do třech částí – WebCenter Content, WebCenter Portal a WebCenter Social. [47]

Silné stránky

WCM Oracle WebCenter Sites je jedním z nejvíce komplexních systému s velkým záběrem pro podnikové použití. Vývoj se v minulosti z oboru IT nasměroval do oboru podnikání a podnikového rozhodování. Jedná se o velice výkonný produkt, schopný spolupracovat s velkým množstvím e-commerce aplikací. [47]

Slabé stránky

Společnost má tendenci nutit svoje zákazníky přecházet na novější software a měnit tak svoji firemní strategii. Pokud si však firma chce ponechat tuto verzi systému a nepřecházet, je vystavována riziku, že Oracle přestane tyto verze podporovat. Existuje zde riziko vycházející z předpokladu, že se nepodaří vybrat konzistentní aplikaci dlouhodobě podporovanou vyvíjející firmou. To pak může přinést nežádoucí náklady na přechod na jinou verzi systému. [47]

4.2.1.5 Microsoft Web Context Manager

Zájem Microsoftu o oblast WCM vzrostl se spuštěním služby SharePoint 2013 poskytující výraznou podporu pro firemní správu obsahu a interní spolupráce. SharePoint je prodáván jako součást některých verzí Office řešení. Online řešení SharePointu obsahuje podporu správy emailů, obsahu a spolupráce. [47]

Silné stránky

V poslední verzi WCM učinil Microsoft několik změn. V hojnější míře začal používat FAST Search, lepé vykresluje uživatelské rozhraní na tabletech a mobilních zařízeních. SharePoint nadále nabízí vyhledávání, podporu sociálních sítí, business intelligence, atd. [47]

Slabé stránky

Systém v některých ohledech stále zaostává za konkurencí nebo požadavky zákazníku aplikuje s velkým zpožděním. Možnosti WCM v cloud prostředí jsou velmi omezené a pro většinu potřeb jsou nedostačující. [47]

4.2.1.6 IBM Web Content Manager

Firma IBM sídlí ve městě Armonk, stát New York a nabízí většinou systém jako samostatné řešení. V některých případech je možné ho získat i jako součást širšího podnikového řešení. IBM Customer Experience kombinuje WCM s produktem WebSphere Portal. [47]

Silné stránky

IBM se snaží cílit na různé typy zákazníků. Díky tomu je WCM vhodné pro široké použití s podporou dalších e-commerce řešení. Firma je schopna integrovat další poskytované aplikace, které tvoří kompaktní systém. Nabídka IBM je flexibilní a tvořená pro potřeby zákazníka. [47]

Slabé stránky

Zákazníci si velmi často stěžují na složitost jednotlivých funkcionalit ve srovnání s některými konkurenčními produkty. Další nevýhodou je pevná svázanost s korporátními aplikacemi, například WebSphere Portal Server. Pevná vazba je tvořena tím, že spolu tyto aplikace nejlépe spolupracují, což uživatele nutí tyto nástroje využívat. [47]

4.2.1.7 HP Web Content Manager

Společnost HP se sídlem v Palo Alto v Kalifornii vstoupila na trh s WCM v roce 2011díky akvizici firmy Autonomy. [47]

Silné stránky

Jedná se o největšího hráče na trhu s WCM díky kombinaci jednotlivých nabídek. Systém je vyvíjen takovým způsobem, aby vyhovoval současným požadavkům a zároveň reagoval na nově vznikající požadavky. Díky tomu je pro potencionální zákazníky přitažlivý. [47]

Slabé stránky

Vize firmy v oblasti WCM je stále ve vývoji a v roce 2012 moc nepokročily. Proto některé firmy nakonec nezvolí tuto možnost, neboť nemají jistotu, zda bude systém splňovat za několik let jejích potřeby. [47]

4.2.2 Postavení systémů v konkurenčním prostředí

Všechny analyzované systémy spojuje stabilní korporátní zázemí a cílení na velké firemní prostředí. Diferenciace je patrná v oblasti zaměření, zatímco jedni se snaží být komplexními hráči na poli WCM, jiní zkouší uspět přesnou specializací. Analyzované skupiny jsou rozděleny do čtyř základních skupin – Challangers, Leaders, Niche Players a Visionaries. Podrobné rozřazení do jednotlivých skupin je vidět na obrázku č. 22. [47]

Obrázek 22: Postavení systému v konkurenčním prostředí. [47]

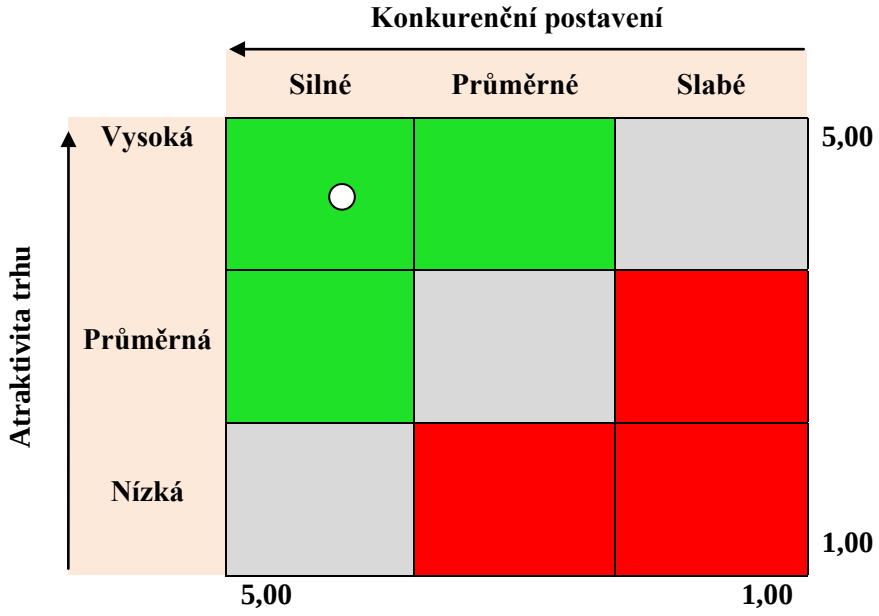
4.2.2.1 GE matice tržního postavení

Vlivy na atraktivitu trhu	Váha	Hodnocení	Součin
Velikost trhu	0,3	5	1,5
Roční tempo růstu	0,3	5	1,5
Charakter konkurence	0,15	4	0,6
Stabilita na ekonomickém trhu	0,05	2	0,1
Technologický vývoj, náročnost produkce	0,1	2	0,2
Rentabilita podnikání	0,1	3	0,3
Celkový vliv			**4,2**

Tabulka 8: Vyhodnocení vlivů na atraktivitu trhu.

Vlivy na konkurenční postavení	Váha	Hodnocení	Součin
Relativní tržní podíl na trhu	0,3	5	1,5
Relativní inovační potenciál	0,1	3	0,3
Distribuční schopnosti	0,1	2	0,2
Marketingový potenciál	0,2	5	1
Ziskovost	0,2	3	0,6
Zkušenost a schopnost vedení	0,1	5	0,5
Celkový vliv			**4,1**

Tabulka 9: Vyhodnocení vlivů na konkurenční prostředí.



Tabulka 10: Tržní postavení společnosti v konkurenčním prostředí.

4.2.3 Analýza kvality prostředí pro vývoj

Systém Adobe CQ5 je vyvinut na platformě Java Standard Edition, ke které je dostupné velké množství dokumentací, případně hotových řešení. K samotné platformě jsou vedeny stránky dokumentace na adrese http://docs.oracle.com/javase/7/docs/api/.

Problémem je dokumentace k samotnému systému. Je nedostatečná a neobsahuje potřebné informace a příklady s řešením. Pokud již dokumentace pro vývoj existuje, je většinou psána v uživatelském duchu. Jedná se o obecný popis bez potřebných detailů pro implementaci.

Ani uživatelská komunita není nikterak obsáhlá. Některé informace se nechají nalézt na serveru http://www.stackowerflow.com, http://dev.day.com, případně na několika dalších serverech. Jejich řešení však nepokrývají dostatečně potřeby vývojářů po informacích, což znatelné prodražuje vývoj komponent.

V celkovém zhodnocení kvality prostředí pro vývoj můžeme zhodnotit tento stav za neodpovídající potřebám IT oddělení. Tuto záležitost by firma Adobe měla velmi intenzivně řešit, což se doposud neděje.

4.3 Shrnutí analýzy

Touto analýzou jsme se snažili poznat současný stav a následné jej zhodnotit. V první části jsme se pokoušeli zaměřit na společnost Hartmann – RICO a.s., respektive na její IT oddělení. Ve druhé části byl předmětem rozboru samotný systém Adobe CQ5. Pomocí analytických metod jsme zjistili, že firma a systém si ve svém konkurenčním prostředí daří poměrně době. Zatím nedostatky systému není možné již nijak ovlivnit, nedostatky zjištěné při analýze procesů spojených s vývojem ovlivnit možné je. Cílem této práce je vytvoření komponent pro systém Adobe CQ5. Na těchto komponentách navrhneme změny proti stávajícímu stavu.

V tomto odstavci definujeme nedostatky, které budeme chtít v následném řešení zapracovat do vyvíjených komponent. Některá doporučení budou implementována, některá budou ponechána v rovině návrhu a některá budou pouze doporučena ke zvážení. Východisky pro tvorbu vlastního řešení jsou:

- Návrh a Implementace několika komponent
- Návrh struktury databázové vrstvy
- Vytvoření Code Rules
- Návrh na vytvoření wiki stránek pro sdílení informací
- Tvorba refaktoringu u důležitých komponent
- Tvorba JavaDoc komentářů

Vnitřní faktory / Vnější faktory	**Slabé stránky** *Neexistence "Code Rules"* *Neexistence jednotné architektury pro vývoj* *Neexistence wiki stránek*	**Silné stránky** *Vlastní servery* *Zavedený verzovací systém* *Vlastní vývojové oddělení* *Finanční stabilita*
Příležitosti *Příznivé podmínky na trhu s technologiemi a oborovém trhu, možnost nabízení doplňkových služeb obchodním partnerům*	Díky dobrým podmínkám na trhu je firma schopna nabízet možné technologické služby i ostatním obchodním partnerům	Flexibilní nastavení díky vlastním serverům Rozvoj lepšího vývojového prostředí, vlastní verzovací systém Lepší přizpůsobení aplikací pro klienty
Hrozby *Neudržitelnost vývoje komponent, bezpečnost* *Finanční ztráty díky neuchovávání informací* *Finanční ztráty při vývoji komponent*	Zkvalitnění zdrojového kódu díky „Code Rules" Zlepšení komunikace díky používání wiki stránek Zrychlení vývoje díky kvalitnější architektuře a refaktoringu	Vývoj aplikací na míru sníží korporátní náklady Zvýšení produktivity práce, snížení finančních ztrát při vývoji komponent

Tabulka 11: SWOT analýza firmy Hartmann – RICO a.s. v oblasti IT.

5 Vlastní návrh řešení

Tato kapitola se bude zabývat návrhem a implementací vlastního řešení. Budou zde popsány specifikace zadání, jejich návrh, implementace, testování a závěrečné nasazení. Postupně budou popsány tři vytvořené komponenty – komponenta „Stížnosti“, „Synchronizace skupin“ a „Mimořádné odměny“.

5.1 Tvorba komponent

Komponenty byly tvořeny pro potřebu podniku, respektive konkrétních firemních oddělení tvořících specifikaci zadání. Vývoj probíhal na testovacím serveru czvbicqt02 pomocí nástroje CRXDELite, případně na lokální verzi CQ5 nainstalovanou na virtuálním stroji CentOS. K virtualizaci byl použit komplexní nástroj od společnosti Oracle – VirtualBox. Po dokončení vývoje a testování byly aplikace převedeny na cílové prostředí. Většinou se jednalo o server czvbicqi01.

Základním cílem je tvorba komponent dle nejlepších o ověřených metodik, které jsou k dispozici v oboru analýzy, návrhu a implementace softwarových komponent. Jsou použita objektová paradigmata, zdrojový kód je členěn do objektů poskytující jednotlivé služby ostatním objektům. Díky tomu je umožněna komunikace s ostatními aplikacemi v budoucnosti a zamezeno masivní duplikaci zdrojového kódu a funkcionalit.

Struktura každé komponenty je rozčleněna do dvou částí – část samostatné viditelné komponenty je ve složce „components“ a jedná se o soubory ve formátu .jsp[2], .xml[3], případně další soubory s metadaty.

5.1.1 Aplikace „Stížnosti“

Tato aplikace byla implementována v prosinci minulého roku a rozdělena do třech základních komponent. Hlavním smyslem aplikace byla podpora zpětné vazby mezi vedením firmy a zaměstnanci s případným rozšířením do obchodního styku se zákazníky.

5.1.1.1 **Specifikace zadání**

Primárním cílem byla implementace tří komponent s odlišnou funkcionalitou – vkládání stížností, zobrazení vložených stížností a odpověď na stížnost. V aplikaci budou rozlišení uživatelé na tyto skupiny:

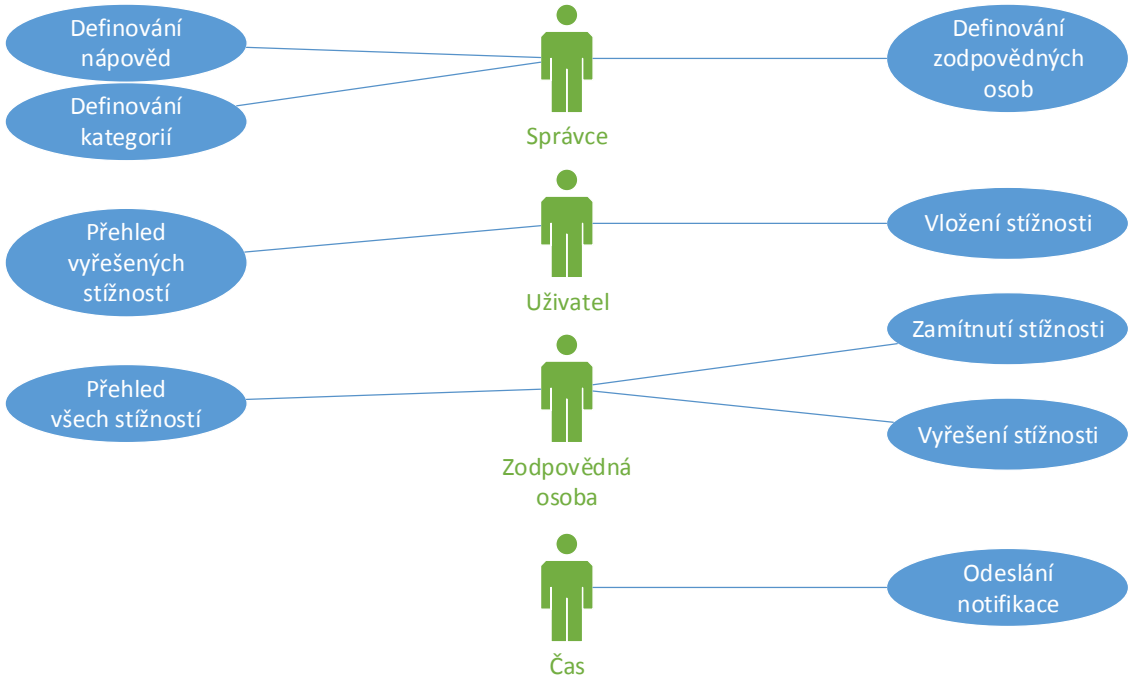
[2] Tento soubor definuje vzhled a obsah komponenty. Jedná se o prezenční vrstvu zajišťující zobrazení dat z vrstvy aplikační. Je možné jich definovat více a v příslušném servletu přepínat vzhled a chování komponenty.
[3] Dialogy jsou definovány pomocí XML dokumentů, umožňují nastavit počet a typ vstupů, kterými je možné komponentu nastavovat bez zásahu do zdrojového kódu. Určeno pro parametrizaci chování.

- **Správce** – má přístupy do DB přímo, může nastavovat aplikací pomocí vkládání hodnot přímo do databáze.
- **Uživatel** – jedná se o osobu s nejnižšími právy. Je mu umožněno prohlížení vyřešených stížností a vkládání příspěvků.
- **Zodpovědná osoba** – stará se o vyřizování jednotlivých příspěvků, je přiřazena k určité kategorii, za kterou má zodpovědnost.

Jednotlivé příspěvky budou vkládány do určitých kategorií, kterým bude možné přiřadit určitou zodpovědnou osobu řešící daný příspěvek. Příspěvek bude po celou dobu pohybu v systému v jednou z těchto stavů:

- **Čekající** – Defaultní stav příspěvku ihned po vložení uživatelem do systému, po tomto stavu je možné přejít pouze do stavu „Zamítnuto“ nebo „Vyřešeno“.
- **Zamítnutý** – Takto je označen příspěvek s neoprávněnou stížností, případně nevhodným a vulgárním obsahem. Jedná se o konečný stav stížnosti.
- **Vyřešený** – do tohoto stavu je možné přejít pouze tak, že zodpovědná osoba odpoví zadavateli dotazu. Jedná se o druhý konečný stav stížnosti.

Aplikace bude odesílat emailové notifikace zodpovědné osobě po vložení příspěvku do její kategorie a uživateli v případě zodpovězení jeho dotazu. Dále bude poskytovat možnost anonymního zadávání.

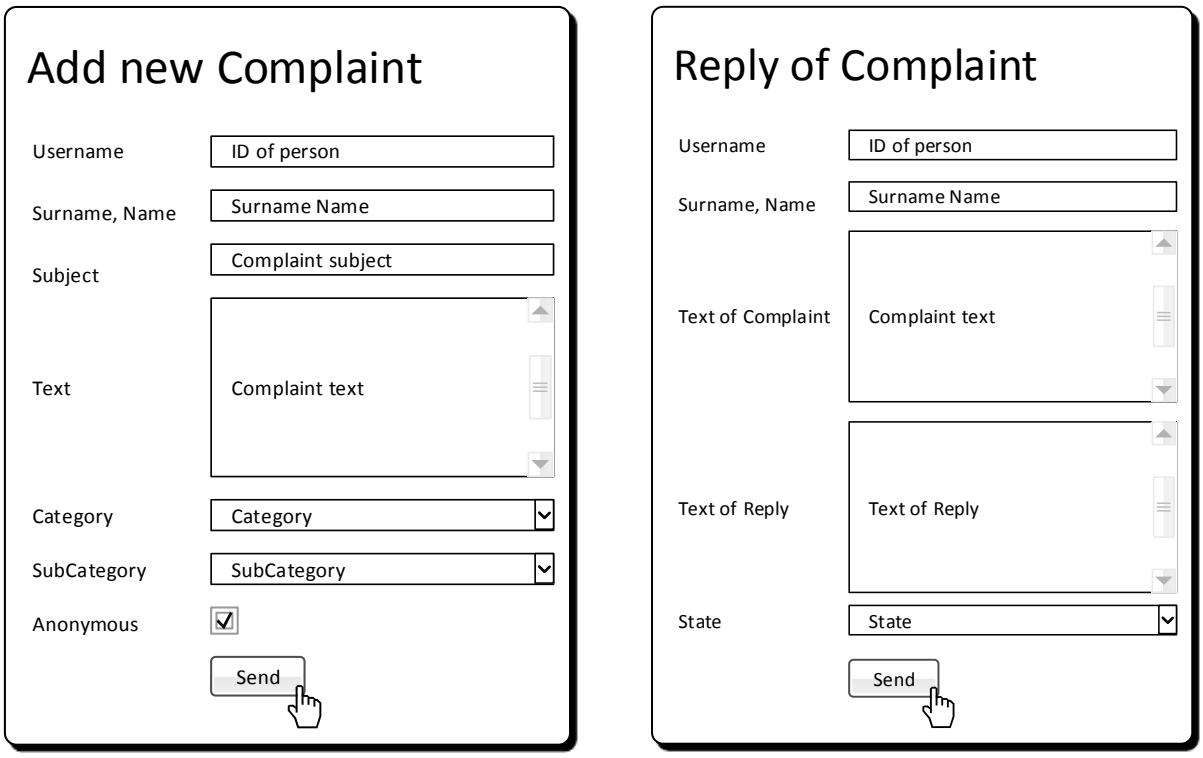


Obrázek 23: Diagram případů užití pro aplikaci „Stížnosti“.

5.1.1.2 Návrh řešení

Návrh řešení je tvořen ve třech rovinách – rovině uživatelského rozhraní, systémové architektuře aplikace a databázové architektuře. Pro první část jsou použity drátové modely sloužící pro upřesnění

potřeb a požadavků zadavatelů. Pro část systémové architektury je použit diagram tříd. Ten slouží k definování relací mezi jednotlivými objekty. Pro návrh databázové struktury jsou použity ER diagramy.



Obrázek 24: Formulář vložení stížnosti a odpověď na stížnost.

List of Complaints

Category | SubCategory | Status | Search

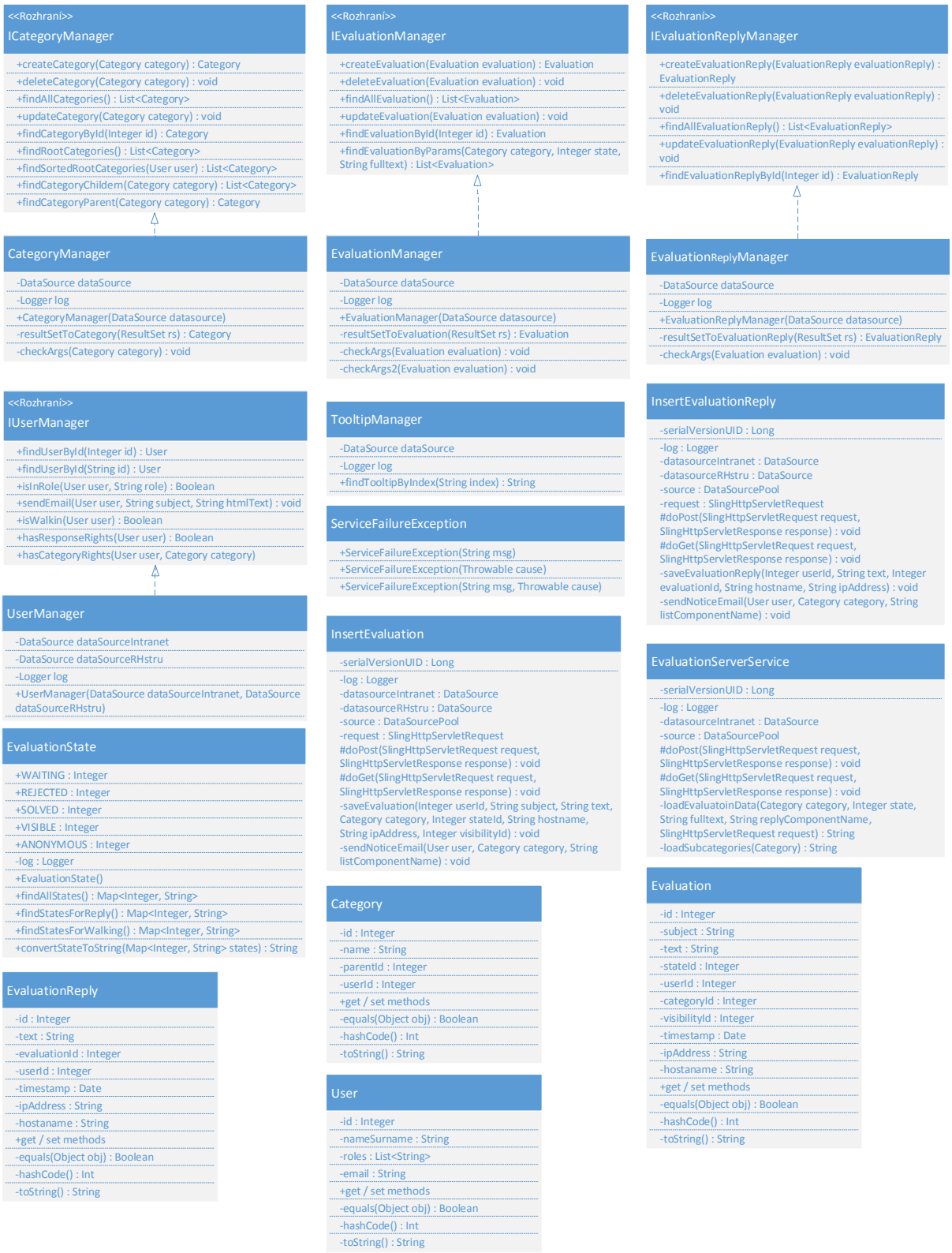
Subject	Author	Text	Reply	Options
Subject 1	Karel Novák	Text 1	View	-
Subject 2	Anonymous	Text 2	-	Reply / Reject
Subject 3	Jan Svoboda	Text 3	View	-
Subject 4	Anonymous	Text 4	-	Reply / Reject

Obrázek 25: Seznam vložených stížností do systému.

Základním smyslem použitý drátových modelů je převedení obecných textových frází do konkrétní podoby určující podrobnější podobu aplikace. Na obrázcích č. 24 a 25 jsou strukturně definovány použité prvky včetně jejich uspořádání. Některé prvky není možné editovat a vyplní je přímo systém dle přihlášeného uživatele. Jedná se o prvky „Username“, „Surname, Name“ a „Text of Complaint“. Ten je vyplněn dle textu v navazující stížnosti. Při výpisu stížnosti je dle definovaných autorizačních pravidel a stavu stížnosti poskytována funkčnost a možnosti nakládání s položkou.

Návrh architektury aplikace byl rozvrhnut do jednadvaceti částí – čtyřech rozhraní, čtrnácti tříd a třech jsp šablon. V návrhu není zahrnuta třída „Activator“, neboť se jedná o systémovou třídu, která

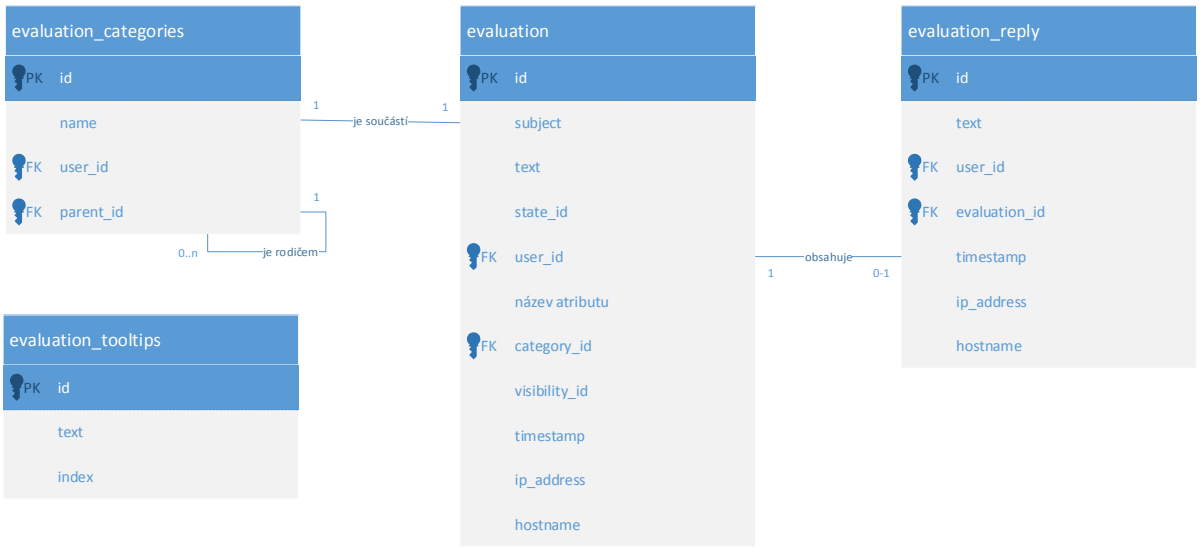
se do jednotlivých komponent vkládá a při startu jsou definované funkce aktivovány systémem Adobe CQ5. Diagram tříd je detailněji popsán na obrázku č. 26.



Obrázek 26: Diagram tříd aplikace „Stížnosti“.

V ER diagramu jsou definovány čtyři základní entity. Hlavním prvkem je entita „evaluation“ obsahující potřebné informace o stížnosti a uživateli, který ji zadával. Na ni jsou navázány entity

„evaluation_reply“ a „evaluation_categories“. V současném návrhu jsou násobné vazby definovány jedna ku jedné, což by bylo možné sloučit do jedné tabulky a nevytvářet tři. Toto není provedeno z důvodu problémové rozšiřitelnosti, neboť se jedná o statickou vazbu. Takto bude rozšiřitelnost mnohem jednodušší a intuitivnější. ER diagram je popsán v obrázku č. 27.



Obrázek 27: ER diagram výsledného řešení.

5.1.1.3 Výpočet funkčních bodů, COCOMO II

Funkční body

V aplikaci se pokusíme určit přesný počet funkčních bodů – ty budeme hledat v následujících oblastech: externí vstupy (EI), externí výstupy (EI), externí dotazy (EI), vnitřní logické soubory (EI) a soubory vnějšího rozhraní (EI).

název	typ	počet
Databáze • jedna tabulka, jeden funkční bod)	ILF	4
Soubory • funkční bod za každý soubor	ILF	34
Komunikace s uživatelem • zahrnuty vstupy uživatele	EI	17
Komunikace s uživatelem • zahrnuty výstupy pro uživatele	EO	20
Celkem (FPC)	-	75

Tabulka 12: Výpočet funkčních bodů.

V každém programovacím jazyce existuje přepočet jednoho funkčního bodu na průměrný počet příkazů dané syntaxe. V tabulce jsou zobrazeny nejčastěji použité programovací jazyky moderními firmami. Součástí jsou i jazyky použité v naší práci – Java, HTML, Javascript a SQL.

Jazyk	Průměr	Medián	Minimum	Maximum	Podíl na projektu
C	97	99	39	333	0%
C++	50	53	25	80	0%
C#	54	59	29	70	0%
Java	53	53	14	134	80%
HTML	34	40	14	48	10%
JavaScript	47	53	14	48	5%
SQL	21	21	13	37	5%

Tabulka 13: Přepočet jednoho funkčního bodu na počet příkazu daného jazyka. [50]

$$\boldsymbol{VFB = p_{Java} * Ø_{Java} + p_{HTML} * Ø_{HTML} + p_{js} * Ø_{js} + p_{SQL} * Ø_{SQL}}$$

$$\boldsymbol{VFB = 0,8 * 53 + 0.1 * 34 + 0,05 * 47 + 0,05 * 21}$$

$$\boldsymbol{VFB = 49,2}$$

$\boldsymbol{VFB}$ je výsledná velikost funkčního bodu, $\boldsymbol{p_{jazyk}}$ odhad celkového výskytu poměru jazyka v aplikaci, $\boldsymbol{Ø_{SQL}}$ průměr jednoho funkčního bodu v přepočtu na syntaktické příkazy.

Odhad počtu řádků zdrojového kódu:

$$\boldsymbol{SLOC = VFB * FPC = 49,2 * 75}$$

$$\boldsymbol{SLOC = 3690,0}$$

COCOMO

Parametry byly nastaveny podle prostředí, podmínek a znalostí tak, aby co nejvíce odpovídaly realitě. Pro tuto metodu je nutné odhadnout počet řádků zdrojového kódu ve výsledné aplikaci. Ta byla odhadem stanovena na 3000 nových řádků a 500 modifikovaných. Odhad počtů řádků je možné určit výpočtem funkčních bodů, případně odhady a následným výpočtem metodou PERT.

Parametr	Hodnota	Parametr	Hodnota
PREC	0,04	**STORE**	1
FLEX	0,03	**PVOL**	0,87
RESL	0,03	**ACAP**	1
TEAM	0,03	**PCAP**	0,74
PMAT	0,03	**AEXP**	0,88
RELY	1,15	**PEXP**	0,8
DATA	0,94	**LTEX**	1,11
CPLX	1	**PCON**	0,83
RUSE	1,08	**TOOL**	0,75
DOCU	0,93	**SITE**	1
TIME	1	**SCED**	1

Tabulka 14: Zvolené parametry pro výpočet metody COCOMO.

Nyní budou vypočteny očekáváné parametry pro připravovaný vývoj. Jedná se o výpočet celkového úsilí, času vývoje a optimálního nasazení počtu pracovníků. Díky malým rozsahům aplikace budeme počítat v organickém módu a základním modelu. Pro tyto případy jsou definovány tyto konstanty: $a = 2{,}4;\ b = 1{,}05;\ c = 2{,}5;\ d = 0{,}38$

$$E = a * KSLOC^b = 2{,}4 * 3^{1{,}05}$$

$$E = 7{,}6$$

V tuto chvíli je nutné zohlednit okolnosti vývoje, ty jsou vybrány z výše uvedené tabulky a výslednou hodnotu upravují součinem.

$$E = 7{,}6 * 0{,}3944 = \mathbf{2,6}$$

$$T = c * E^d = 2{,}5 * 2{,}6^{0{,}38} = \mathbf{3,6}$$

$$P = \frac{E}{T} = \frac{2{,}6}{3{,}6} = \mathbf{0,72}$$

Doba trvání jednotlivých fází vývoje

Výpočet závisí na mnoha proměnných, dle kterých se výsledná doba vývoje může měnit na obě strany. Výsledný výpočet byl proveden ručním výpočtem a stanovil výslednou dobu vývoje komponent na 3,6 měsíce.

5.1.1.4 Implementace

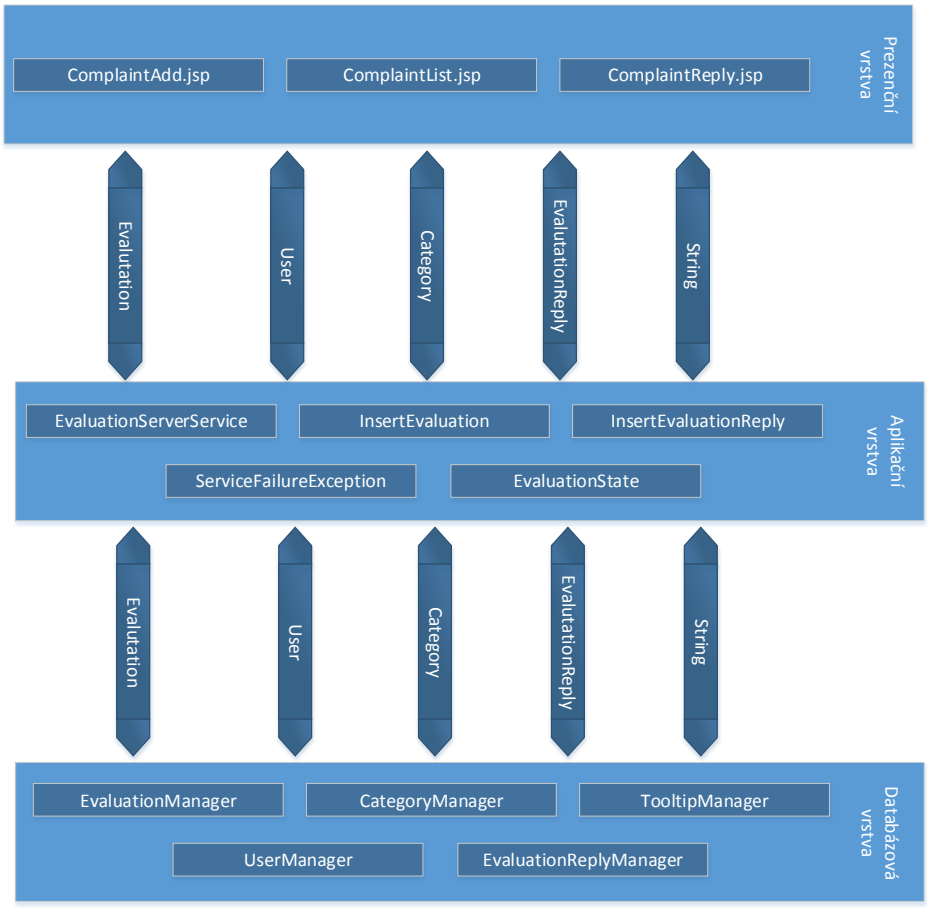
Implementace byla provedena na základě předchozích kroků – specifikace požadavků a analýzy řešení. Díky nimž jsme jasně stanovili přesnou architekturu aplikace a jednotlivé závislosti mezi vrstvami v systému. Tyto vazby jsou lépe viditelné na obrázku č. 28, kde pro komunikaci mezi vrstvami jsou definovány objekty, které jsou mezi nimi předávány.

Jednotlivé komponenty byly implementovány v jazyce Java, HTML, JavaScript a v některých případech byly použity jednoduché CSS styly. Pro uchování perzistentních dat byla využita MS SQL databáze. Databáze jsou ve společnosti Hartmann – RICO a.s. rozděleny do několika částí – v této aplikaci jsme pracovali s databázemi:

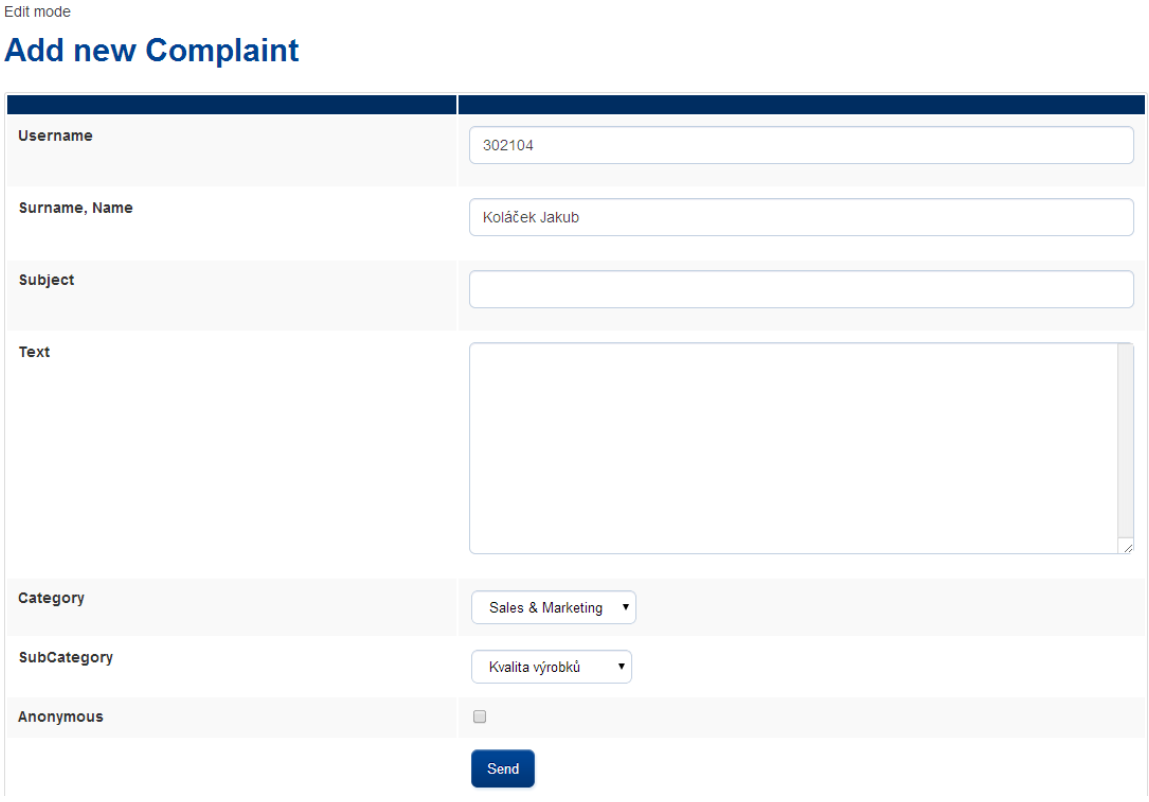
- **INTRANET** – ukládání dat o stížnostech, odpovědí na stížnost, kategorií a nápověd pro jednotlivé inputy.
- **RHSTRU** – z této databáze byla získávána data o zaměstnancích, včetně jejich emailových adres.

K vývoji jednotlivých komponent byly využity nástroje na testovacím serveru cczvbicqt02:4502.

Bezpečnostní opatření, autorizační a autentizační pravidla nebylo potřebné definovat, neboť systém Adobe CQ5 tyto funkcionality má dobře propracované. Přesně je možné definovat přístupy určitým osobám na požadovaná místa v systému. Autorizační pravidla byla tvořená pouze pro sémantické potřeby vyvíjených komponent. Jedná se o poskytováních různých funkcionalit dle postavení uživatele v kategoriích stížností.

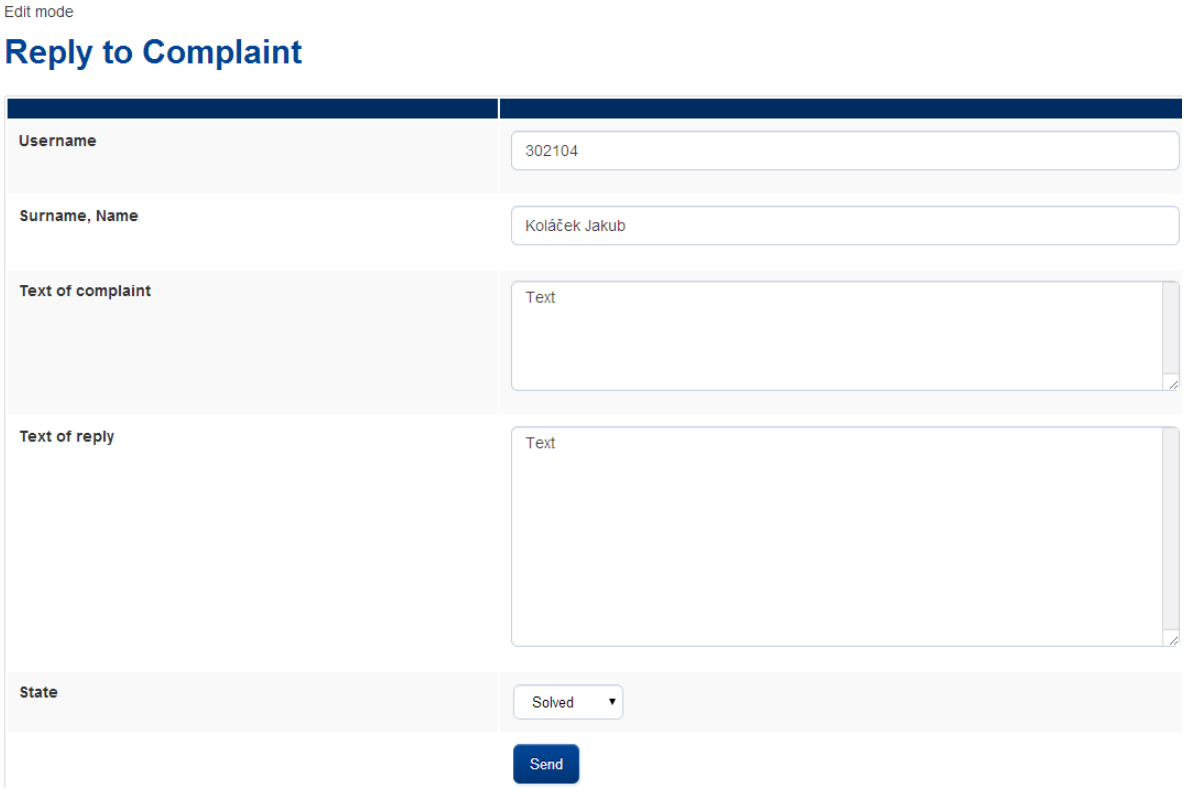


Obrázek 28: Přiřazení objektů jednotlivým vrstvám a vzájemná komunikace mezi nimi.



Obrázek 29: Výsledná implementace komponenty „Přidání stížnosti“.

Obrázek 30: Výsledná implementace komponenty „Zobrazení vložených stížnosti“.



Obrázek 31: Výsledná implementace komponenty „Odpověď na stížnost“.

Výjimky a logování chyb

Pokud se v programu vyskytne jakákoliv chyba, je automaticky zapsána do systémového logu s označením „ERROR“ a v případě potřeby přeposlána vyšší vrstvě. Za hlavními funkcemi je opět nastaveno logování aktivity. Jedná se o zaznamenávání průběhu workflow. Zápisy jsou do logu zapisovány s označením „INFO“. Logování bylo zavedeno ze dvou důvodů. Prvním důvodem je zkvalitnění ladění chyb při vývoji a testování. Druhým důvodem je možnost sledování nestandardního chování ze strany systému při komerčním nastavení v produkci. Pokud se takové chování vyskytne, je možné při kvalitním logování tuto chybu zjistit a případně odladit. V případech, že logování chybí, je pak odchytávání chyb velmi problematické.

```
private final static Logger log =
LoggerFactory.getLogger(EvaluationManager.class);

if (evaluation.getId() != null) {
     log.error("Internal Error: Evaluation exists in DB.");
     throw new IllegalArgumentException("evaluation id is
     already set");
}

log.info("WorkFlow => EvaluationManager.CreateEvaluation() :
Evaluation: "+evaluation);
```

Box 1: Příklad práce s výjimkami a logováním chyb.

Komentování zdrojového kódu

Zdrojový kód je komentován dle zavedených standardů pro tvorbu JavaDoc dokumentace. Dokumentace je rozdělena dle komentovaných bloků na tyto části:

- **Komentáře souborů:** Definuje informace o souboru, ve kterém je třída, případně rozhraní uloženo. Obsahuje informace o autorovi, datum vývoje, název souboru, balíček, ke kterému patří a krátký popis.

```
/*
 * Project to Master Thesis - EvaluationManager of Evaluation
 *
 * @description Class describes object EvaluationManager
 * @package info.hartmann.evaluation
 * @file EvaluationManager.java
 * @author Jan Pesava - xpesav00
 * @email xpesav00@fbm.std.vutbr.cz
 * @date 10. 12. 2013
 */
```

Box 2: Příklad komentáře pro soubor EvaluationManager.java.

- **Komentáře celé třídy, případně rozhraní:**

```
/**
 * Project to Master Thesis - EvaluationManager of Evaluation
 * Class provides services for work with Evaluation with
 * database. Class provides CRUD operations – create, read,
 * update and delete
 *
 * @version     %I%, %G%
 * @since       1.0
 */
```

Box 3: Příklad komentáře pro třídu EvaluationManager.

- **Komentáře jednotlivých metod:** Slouží k popisu funkcionality dané metody včetně jednotlivých parametrů a návratové hodnoty. Dále definuje výjimky, které mohou v metodě nastat a nejsou vyřešeny uvnitř bloku.

```
/**
 * Method creates evaluation in database
 * @param evaluation is object required to save in database
 * @throws ServiceFailureException
 * @return the same object with id
 */
public Evaluation createEvaluation(Evaluation evaluation)
throws ServiceFailureException;
```

Box 4: Příklad komentáře metody pro vytvoření stížnosti v rozhraní IEvaluationManager.

- **Inline komentáře**: Jsou používány uvnitř jednotlivých metod nebo při definici jednotlivých atributů. Jsou uvozeny znakem „//“ a mají platnost od místa uvedení do konce řádku. Většinou slouží k upřesnění konkrétních programových zápisů pro lepší pochopitelnost a přehlednost.

```
//add new groups from RTStru to CQ
for (Map.Entry<Groups, List<Users>> entry :
rhStruGroups.entrySet()) {
      Groups groups = (Groups)entry.getKey();

       //control group
      Group group = (Group)userManager.getAuthorizable(
                    groups.getId().toString());
     if(group == null) {
           group = createGroup(groups, userManager, session);

           //synchronize
           synchronizeUserGroups(groups, group, userManager,
                                 session);
      } else {
           this.addInfo(groups, "EXISTING");
     }
}
```

Box 5: Příklad inline komentáře pro synchronizaci skupin ve třídě GroupSynchronzationManager.

Všechny uvedené typy komentářů se vztahují k programovacímu jazyku Java, pro ostatní použité technologie nemusí být podobné. Například pro značkovací jazyk HTML je koncepce komentování naprosto odlišná. V ostatních technologiích nebyly komentáře v tak velké míře používány.

5.1.1.5 Testování a nasazení

Aplikace byla testována pomocí základních nástrojů pro testování softwaru. Nebylo použito žádných sofistikovanějších řešení – jako jsou jednotkové testy, testy uživatelského rozhraní – Selenium testy. Nebyla použita ani žádná z dostupných metodiky zaměřujících se právě na testování.[4] Testování probíhalo na testovacím serveru `czvbicqt02:4502` pomocí ručního testování pomocí informací z logu a chování aplikace v prohlížeči. Testování probíhalo v přesně definovaných postupech, tyto postupy jsou tyto:

- Testování samostatných funkcionalit
- Testování celé aplikace programátorem před nasazením na produkční server
- Testování celé aplikace zadavatelem před nasazením na produkční server

[4] Zde je především myšlena metodika TDD – Test Development Driven, která nejdříve definuje a implementuje podmínky pro testování a až následně implementuje samotnou funkcionalitu systému. Tím se zabrání neplánovaným změnám a takzvanému. „ohýbání“ funkcionality systému.

- Přenesení aplikace na produkční server, opětovné otestování funkcionalit

Všechny komponenty byly nasazeny na produkční server `czvbicqt02:4502` sloužící pro interní potřeby v informační podpoře pro zaměstnance. Nasazení proběhlo v pořádku, vše prošlo akceptačním testování, aplikace je schválena a připravena na spuštění.

5.1.1.6 Metriky

V poslední kapitole shrneme základní údaje o aplikaci pro práci se stížnostmi z několika pohledů. Většinou se bude jednat o kvantitativní jednotky vztahující se ke zdrojovým datům.

Počet komponent: 3

Počet souborů a řádků:[5]

Jazyk	soubory	komentářů	Zdrojový kód
Java	**20**	**570**	**2594**
JSP	**3**	**163**	**686**
XML	**11**	**0**	**128**
Celkem	**34**	**733**	**3408**

Tabulka 15: Základní data o metrikách aplikace „Stížnosti“.

Počet funkčních bodů: 75

Doba vývoje (odhad): 3,6 měsíce

Doba vývoje (realita): 3,2 měsíce

5.1.2 Aplikace „Mimořádné odměny“

Tato aplikace byla implementována v únoru letošního roku a je rozdělena do třech hlavních komponent. Hlavním cílem tvorby komponenty je centralizace firemních procesů a zrušení podávání současných papírových žádostí.

5.1.2.1 Specifikace zadání

Cílem tvorby aplikace je získání komponent integrovaných v systému Adobe CQ5 umožňující podávání žádostí o mimořádné odměny v elektronické podobě. Tento způsob nahradí současnou těžkopádnou papírovou formu podání. Aplikace by měla poskytovat přehled všech podaných žádostí, tedy i ty, které jsou již vyřešeny. Systém bude pracovat s třemi základními osobami – uživatel, manažer a člen představenstva.

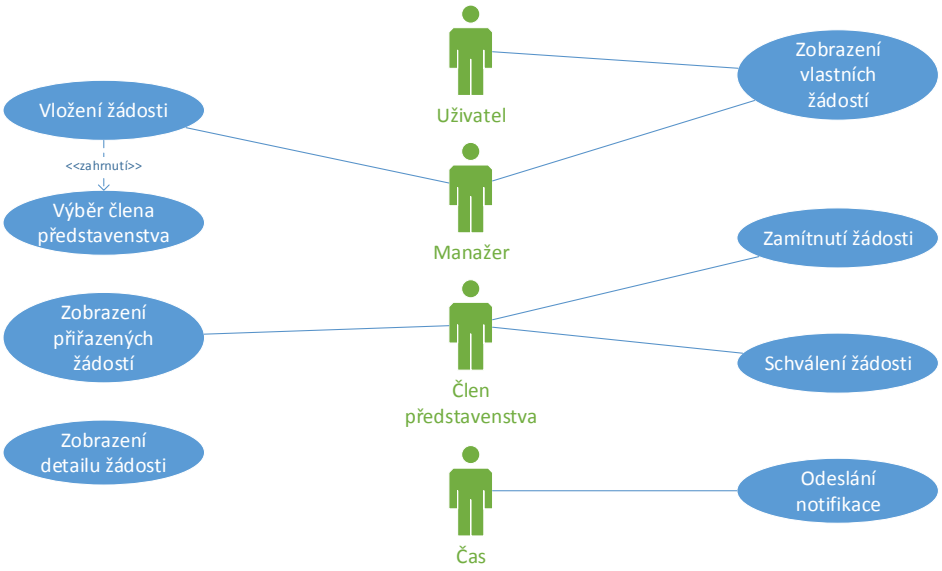
Žádost do systému vkládá vždy manažer oddělení a podává ji za svého zaměstnance, kterého tímto navrhuje vedení firmy na finanční odměnu. Do formuláře by mělo být možné vložit základní

[5] Statistika je generována programem cloc-1.60.exe. Ve výčtu jsou zahrnuty i některé soubory vytvořené systémem Adobe CQ5 nutné k vnitřnímu nastavení. Jedná se o soubory typu XML.

informace o navrhované částce, její splatnost a vhodné zdůvodnění žádosti. Na samém konci formuláře je možné vybrat člena představenstva, který bude danou žádost posuzovat a následně o ní rozhodovat.

Po vložení žádosti do systému je odeslána notifikační zpráva na email zvolenému členovi představenstva. Ten si ji zobrazí a může rozhodnout takto:

- Zamítnout: žádost je odmítnuta, odměna nebude vyplacena.
- Schválit: žádost je schválena, odměna bude vyplacena.

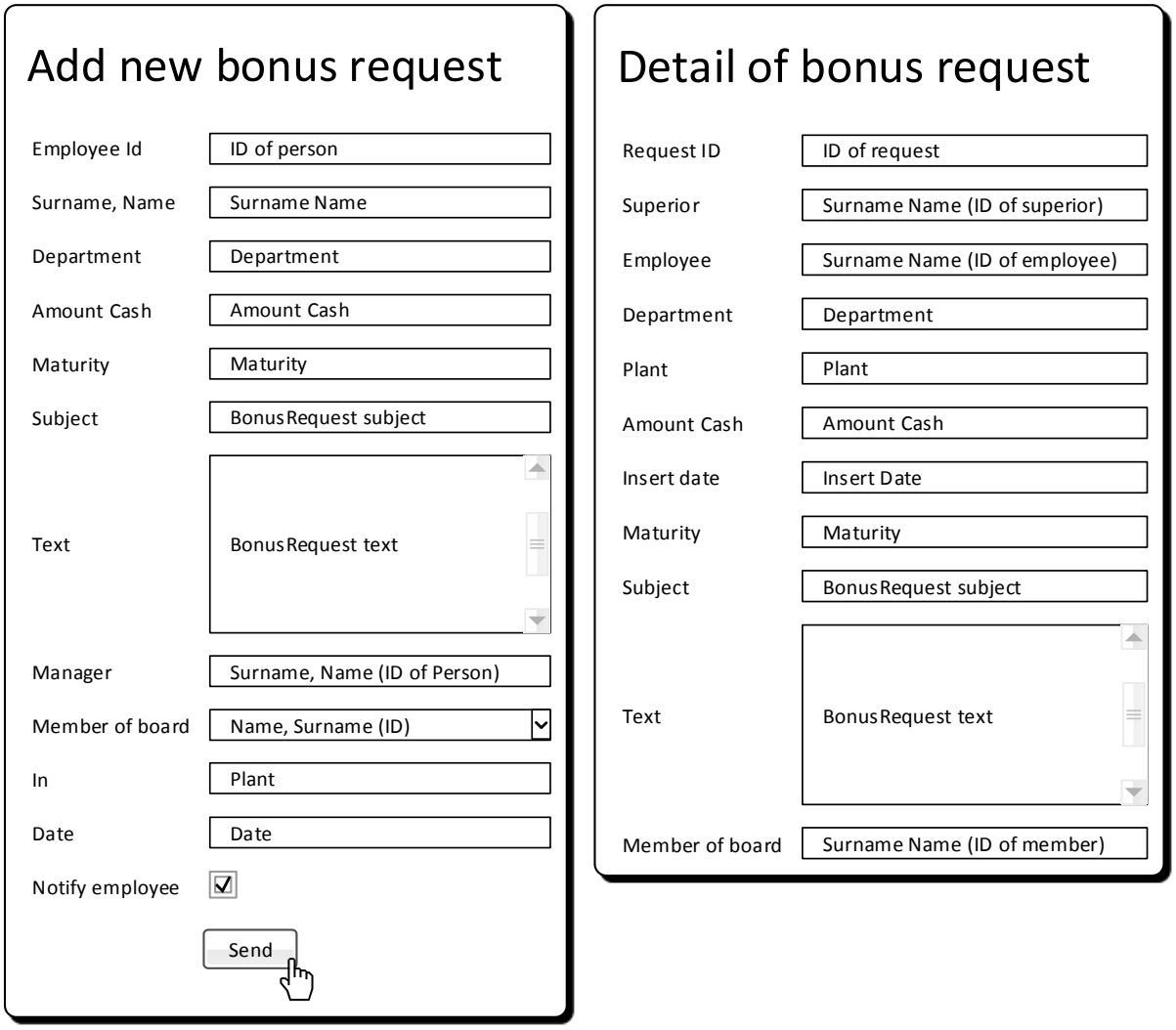


Obrázek 32: Diagram aktivit pro aplikaci „Mimořádné odměny“.

Po schválení, případně zamítnutí odměny, je poslána notifikace podavateli žádosti – manažerovi. Pokud pří vkládání žádosti byla zvolena možnost „Notify employee“, pak bude notifikace o výsledku posouzení zaslána jak manažerovi, tak zaměstnanci. V opačném případě bude notifikační email odeslán pouze manažerovi, který o výsledku zaměstnance informuje osobně.

5.1.2.2 Návrh řešení

Aplikace bude rozdělena do třech systémových komponent – přidání žádosti, zobrazení žádostí a detail žádosti. Jednotlivé komponenty budou navrženy ve dvou rovinách – rovině uživatelské rozhraní a návrhu databázových tabulek. K těmto potřebám budou využity již známé nástroje a metodologie: drátové modely a ER diagramy.



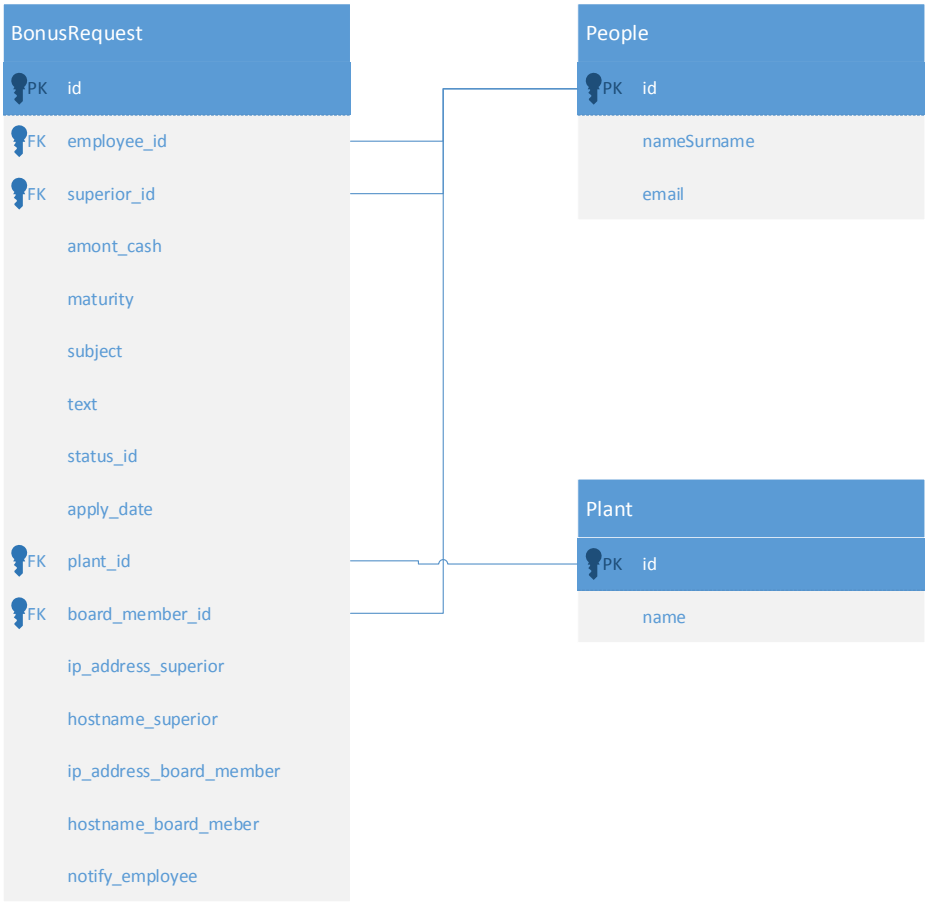
Obrázek 33: Formulář vložení žádosti o odměnu a detail žádosti o odměnu.

List of bonus requests

Status

Supervizor	Employee	Subject	Date of post	Detail	Options
Karel Novák	Karel Vostrý	Subject 1	1. 1. 2014	View	Approve / Reject
Jan Kolomý	Simona Malá	Subject 2	10. 1. 2014	View	Approve / Reject
Aleš Soukal	Jana Černá	Subject 3	12. 2. 2014	View	Approve / Reject
Jan Kolomý	Anna Mokrá	Subject 4	17. 2. 2014	View	Approve / Reject

Obrázek 34: Návrh komponenty pro zobrazení vložených žádostí.



Obrázek 35: ER-diagram aplikace „Mimořádné odměny“.

5.1.2.3 Výpočet funkčních bodů, COCOMO

Funkční body

V aplikaci se pokusíme určit, podobně jako v předchozím případě, přesný počet funkčních bodů, které budeme opět hledat v následujících oblastech: externí vstupy (EI), externí výstupy (EI), externí dotazy (EI), vnitřní logické soubory (EI) a soubory vnějšího rozhraní (EI).

název	typ	počet
Databáze • jedna tabulka, jeden funkční bod)	ILF	1
Soubory • funkční bod za každý soubor	ILF	36
Komunikace s uživatelem • zahrnuty vstupy uživatele	EI	10
Komunikace s uživatelem • zahrnuty výstupy pro uživatele	EO	12
Celkem (FPC)	-	59

Tabulka 16: Výpočet funkčních bodů.

Odhad počtu řádků zdrojového kódu:

$$\boldsymbol{SLOC = VFB * FPC = 49,2 * 59}$$

$$\boldsymbol{SLOC = 2902,8}$$

COCOMO

Parametry byly nastaveny dle prostředí, podmínek a znalostí tak, aby co nejvíce odpovídaly realitě. Pro tuto metodu je nutné odhadnout počet řádků zdrojového kódu ve výsledné aplikaci. Ta byla odhadem stanovena na 2000 nových řádků a 1000 modifikovaných.

Parametr	Hodnota	Parametr	Hodnota
PREC	0,04	**STORE**	1
FLEX	0,03	**PVOL**	0,87
RESL	0,03	**ACAP**	1
TEAM	0,03	**PCAP**	0,74
PMAT	0,03	**AEXP**	0,88
RELY	1,15	**PEXP**	0,8
DATA	0,94	**LTEX**	1,11
CPLX	1	**PCON**	0,83
RUSE	1,08	**TOOL**	0,75
DOCU	0,93	**SITE**	1
TIME	1	**SCED**	1

Tabulka 17: Zvolené parametry pro výpočet metody COCOMO.

Nyní budou vypočteny očekáváné parametry pro připravovaný vývoj. Jedná se o výpočet celkového úsilí, času vývoje a optimálního nasazení počtu pracovníků. Díky malým rozsahům aplikace budeme počítat v organickém módu a základním modelu. Pro tyto případy jsou definovány tyto konstanty: $a = 2{,}4;\ b = 1{,}05;\ c = 2{,}5;\ d = 0{,}38$

$$E = a * KSLOC^{b} = 2{,}4 * 2^{1{,}05}$$

$$E = 4{,}97$$

V tuto chvíli je nutné zohlednit okolnosti vývoje, ty jsou vybrány z výše uvedené tabulky a výslednou hodnotu upravují součinem.

$$E = 4{,}97 * 0{,}3944 = \boldsymbol{1,96}$$

$$T = c * E^{d} = 2{,}5 * 1{,}96^{0{,}38} = \boldsymbol{3,23}$$

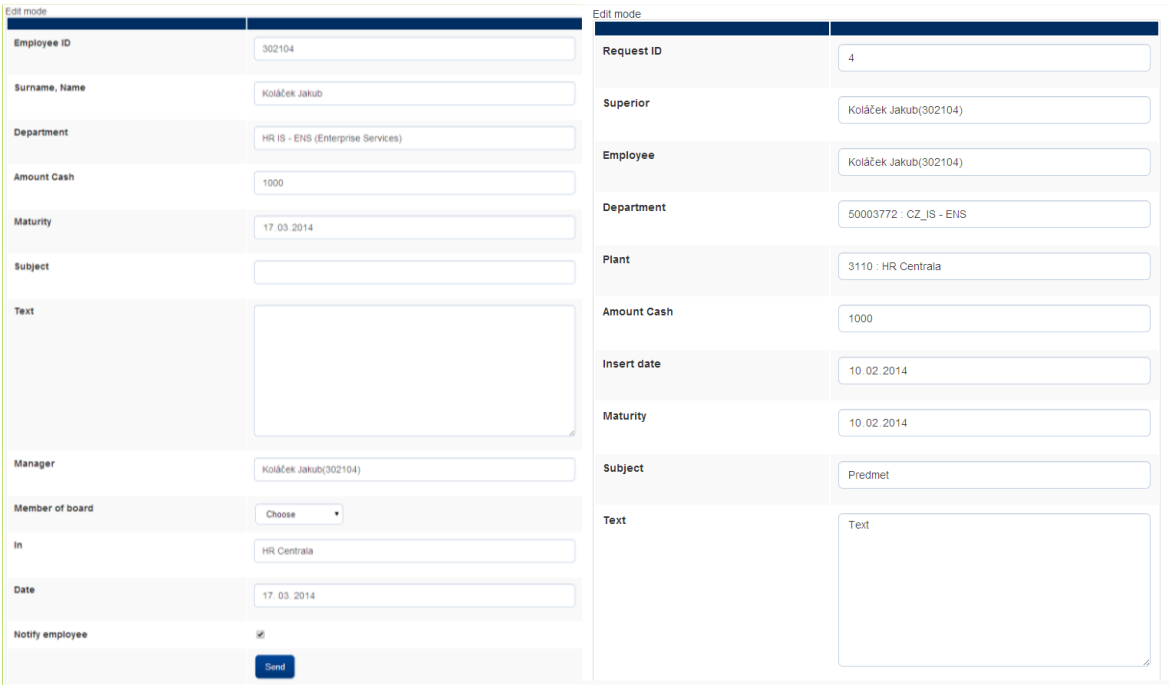
$$P = \frac{E}{T} = \frac{1{,}96}{3{,}23} = \boldsymbol{0,6}$$

Doba trvání jednotlivých fází vývoje

Výpočet závisí na mnoha proměnných, dle kterých se výsledná doba vývoje může měnit na obě strany. Výsledný výpočet byl proveden pomocí ručních propočtů a stanovil výslednou dobu vývoje komponent na 3,23 měsíce.

5.1.2.4 Implementace

Implementace byla provedena obdobně jako v předchozí aplikaci. Výsledky jednotlivých komponent jsou viditelné v níže zobrazených obrázcích.



Obrázek 36: Výsledná komponenta přidání a detail žádosti o odměnu.

Edit mode

Waiting

Supervizor	Employee	Subject	Date of post	Detail	Options
Koláček Jakub (302104)	Semotán Ondřej (301116)	dddd	12. 02. 2014	Show	Approve Reject
Koláček Jakub (302104)	Semotán Ondřej (301116)	Test	12. 02. 2014	Show	Approve Reject
Koláček Jakub (302104)	Koláček Jakub (302104)	Subject - test	12. 02. 2014	Show	-
Koláček Jakub (302104)	Koláček Jakub (302104)	Test	10. 02. 2014	Show	-
Koláček Jakub (302104)	Koláček Jakub (302104)	Predmet	10. 02. 2014	Show	-
Koláček Jakub (302104)	Koláček Jakub (302104)	Subject1	10. 02. 2014	Show	-

Obrázek 37: Výsledná komponenta zobrazení žádostí o odměnu.

5.1.2.5 Testování a nasazení

Aplikace byla testována a nasazena obdobně jako v kapitole 5.1.1.5.

5.1.2.6 Metriky

V poslední kapitole shrneme základní údaje o aplikaci pro práci se stížnostmi z několika pohledů. Budeme sledovat stejné parametry jako u předchozí aplikace.

Počet komponent: 3

Počet souborů a řádků:[6]

Jazyk	soubory	komentářů	Zdrojový kód
Java	**21**	**507**	**2156**
JSP	**3**	**110**	**570**
XML	**12**	**0**	**155**
Celkem	**36**	**617**	**2881**

Tabulka 18: Základní data o metrikách aplikace „Mimořádné odměny“.

Počet funkčních bodů: 59

Doba vývoje (odhad): 3,23 měsíce

Doba vývoje (realita): 2,9 měsíce

5.2 Návrh interních pravidel

Tato část má za cíl definovat některá interní pravidla pro zkvalitnění vývoje komponent uvnitř firmy. Jedná se především o návrh struktury databázové vrstvy, pravidel pro vývoj, nástrojů pro kvalitnější komunikaci a tvorbu refaktoringu.

5.2.1 Návrh struktury vrstev

V této části budeme definovat některé nové vrstvy včetně popisu uložení v jednotlivých složkách systému Adobe CQ5.

Základní, již existující, vrstvy:

- `Components` – obsahuje definice komponent (jsp, xml soubory)
- `Bundles` – obsahuje základní business logiku, datovou a aplikační vrstvu (java soubory)
- `Content` – obsahuje obsah a částečnou systémovou konfiguraci

5.2.1.1 Složka Library

Umístění: `bundles/src/main/java/info/hartmann/`

[6] Statistika je generována programem cloc-1.60.exe. Ve výčtu jsou zahrnuty i některé soubory vytvořené systémem Adobe CQ5 nutné k vnitřnímu nastavení. Jedná se o soubory typu XML.

Složka je určena pro uložení vytvořených modelů a tříd, které mohou využívat i ostatní programátoři při vývoji jiných aplikací. Složka je dále rozdělena do následujících podsložek:

- **exceptions** – slouží pro definici potřebných a neexistujících výjimek.
- **general** – slouží pro uložení knihoven a modelů využitelných napříč systémy
- **intranet** – definuje knihovny a modely pro práci s databází intranet – 1 složka zajišťuje práci s jednou tabulkou v databázi.
- **rhstru** – definuje knihovny a modely pro práci s databází rhstru – 1 složka zajišťuje práci s jednou tabulkou v databázi.
- **servlets** – uložiště servletů poskytujících podpůrné služby například pro autocomplete nápověd v jQuery, atd.

5.2.1.2 Složka Intranet

Umístění: `bundles/src/main/java/info/hartmann/`

Jsou zde uloženy v jednotlivých složkách třídy, servlety a další části zdrojového kódu, které se týkají pouze samotné aplikace a není možné je použít pro aplikace jiné. Většinou se jedná o servlety, případně třídy definující komunikaci se samotnou komponentou.

5.2.1.3 Struktura tříd a rozhraní v datovém modelu

Rozhraní manažera: Tato položka je nepovinná a definuje komunikační rozhraní výsledného manažera. V budoucnu může sloužit k lepší škálovatelnosti datových modelů. Základním obsahem rozhraní jsou tzv. CRUD operace:

- **Create** – funkce slouží pro vytvoření objektu v databázi
- **Read** – funkcionalita poskytující pro čtení objektů z databáze
- **Update** – model obsahuje funkce pro editaci objektu v databázi
- **Delete** – funkce pro odstranění objektu z databáze

Rozhraní objektu: Jedná se o další nepovinnou položku definující strukturu objektů obsahující data o dané entitě. V budoucnu může sloužit podobně jako rozhraní manažera pro lepší škálovatelnost systému. Případně vytváření odlišných objektů s podporou stejné funkcionality od současných funkcí.
Manažer: V případě, že existuje rozhraní manažera, pak toto rozhraní implementuje. Jedná se o třídu poskytující stejnou funkcionalitu jako rozhraní manažera. U některých metod nemusí být implementována vnitřní logika – například metody `create, update`. Tato položka je povinná.
Model: Model je třída korespondující s tabulkou v databázi. Definuje formát dat získávané z databáze a poskytované aplikační vrstvě. Tato položka je povinná.

V rámci diplomové práce byly vytvořeny následující modely:

- **`Division`**
  - `IDivisionManager`
  - `DivisionManager`
  - `Division`
- **`Department`**
  - `IDepartmentManager`
  - `DepartmentManager`
  - `Department`
- **`User`**
  - `IUserManager`
  - `UserManager`
  - `User`
- **`BonusRequest`**
  - `IBonusRequestManager`
  - `BonusRequestManager`
  - `BonusRequest`
- **`Plant`**
  - `IPlantManager`
  - `PlantManager`
  - `Plant`

Manažer daného modelu pracuje se stejnojmennými tabulkami v cílové databázi a poskytuje základní CRUD funkcionality. Jedná se především o metody:

- `createObject(Object)`
- `deleteObject(Object)`
- `findAllObjects()`
- `updateObject(Object)`
- `findObjectById(Id)`

K tomu mohou být vytvořeny libovolné metody potřebné pro další práci s definovanou tabulkou. Díky vytvořeným dokumentačním komentářům je možné vytvořit html dokumentaci, která bude ostatním programátorům poskytovat informace o existujících třídách a rozhraních možných k opětovnému použití.

5.2.2 Návrh „Code Rules“

Pravidla pro vývoj jsou základní příručkou, která se stává nutností pro každou softwarovou firmu, protože jasně definují základní situace včetně doporučených postupů a řešení. Firma Hartmann – RICO a.s. žádná taková pravidla neměla. Ani své vývojáře nezavazovala k dodržování zavedených pravidel uvedených v dokumentu „Code Convention“ na adrese http://www.oracle.com/technetwork/java/codeconv-138413.html.

Přínosy použití „Code Rules“ :

- Zpřehlednění kódu
- Stejný styl kódu od různých programátorů
- Kvalitnější a levnější vývoj

Pravidla pro vývoj jsou vypracována dle výše zmíněné specifikace a přiložena v příloze označené písmenem C. Do pravidel byla vložena ještě některá pravidla, která nejsou ve specifikaci řešeny a dle vlastního uvážení jsou pro vývoj důležitá. Dalším důvodem, proč nebyl dokument „Code Convention“ jednoduše převzat je důležitost přizpůsobení na konkrétní podmínky ve společnosti.

Vytvořená pravidla mají rozsah 14 stran, z toho dvě poslední se spíše věnují vytvořené architektuře na uložišti GIT.

5.2.3 Návrh wiki stránek

Wiki stránky jsou navrhovány jako podpora pro vývoj a výměnu informací mezi programátory. Zmíněný nástroj je možné rozšířit do celé firmy a vytvořit tak přehledný sklad informací poskytující ucelená data o vnitřních procesech, systémech, databázích, atd.

5.2.3.1 XWiki

Jedná se o profesionální wiki s podporou pro podnikové potřeby – přidělování autorizačních práv, tvorba blogů, ověřování vůči LDAP, exporty dokumentů do PDF, atd. Má kvalitní a propracovaný design a podporuje přidávání nových vzhledů.

Funkcionality nástroje XWiki je možné rozšiřovat. Jde totiž o vysoce modulární architekturu s podporou pluginů. Tento nástroj je možné využít zdarma, je dostupná pod LGPL licencí. [52, 53]

5.2.3.2 Confluence

Confluence je wiki nástroj pro podporu komunikace a uchovávání informací v rámci firmy. Jedná se o korporátní systém používaný nadnárodními společnostmi. Díky tomu je systém velice komplexní a poskytuje všechny základní prostředky pro týmovou podporu.

Dlouhodobé použití nástroje je možné pouze po zaplacení licenčního poplatku, který je rozdělen na několik fází dle objemu uživatelů. Licenční podmínky pro cenotvorbu jsou definovány v následující tabulce. [51]

Počet uživatelů	10	15	25	50	100	500	2000
Cena za měsíc ($)	10	50	100	200	300	500	1000

Tabulka 19: Licenční podmínky pro cenotvorbu za jednotlivé licence. [51]

Používají například firmy: [51]

- Facebook
- Skype
- Microsoft
- Adobe
- LinkedIn
- Nike
- Twitter
- HubSpot

5.2.4 Návrh refaktoringu

Refaktoring je pro tvorbu vrstev poskytujících služby více subjektům téměř nutností. Pomáhá lepší čitelnosti a snadnějšímu pochopení vlastního i cizího zdrojového kódu. Je možné odstranit neefektivní a zavádějící konstrukce a tím dosáhnout většího výkonu. To odpovídá části Demingova PDCA cyklu, který napomáhá kontrole postupů, případně nápravě vzniklých chyb pro další cykly.

Refaktoring je nutnou záležitostí pro složku „`library`“ obsahující modely pro celý systém. Pokud budou dodržována definovaná pravidla a vývoj bude prováděn svědomitě, pak rozsah nutných refaktoringu nebude tak zásadní.

5.3 Ekonomické zhodnocení navrhovaného řešení

V této části bude ekonomicky zhodnocen přínos pro firmu a finanční náročnost řešení návrhů uvedených v této diplomové práce. Hned v úvodu bychom rádi podotkli složitost tohoto zhodnocení pro danou oblast. Velká část ekonomického zhodnocení může vést k diskuzím všech čtenářů o tom, zda je zhodnocení realistické. Tento problém nastává ihned z několika důvodů. První problém nastává v oblasti s kvantifikací přínosů v porovnání s nákladovostí na zavedení řešení. Druhým problémem je frekvence použití daného řešení. Ten je potřebné zohlednit u tvorby modelů a vnitřní architektury komponent. Nicméně se pokusíme toto zhodnocení provést co nejobjektivněji a učinit tak výsledné závěry.

5.3.1 Ekonomické zhodnocení vývoje komponent

Zhodnocení jednotlivých aplikací bude prováděno pouze z jednoho pohledu – finanční náročnosti na provedený vývoj. Vývoj bude vycházet z dosažené reality, i získaných výpočtů z metody COCOMO II. Pro výpočty budeme předpokládat průměrný hodinový plat na vývoj komponenty pro Adobe CQ5 ve výši 250 Kč na hodinu.

Ekonomické zhodnocení se netýká pouze programátorské činnosti. Vypočtené částky jsou odrazem celkových nákladů všech zúčastněných pracovníků a zahrnují fáze od nápadu danou aplikaci realizovat, až po finální nasazení. Proto finální cena aplikace není počítána cenou programátora, ale průměrnou cenou práce všech kooperujících osob.

5.3.1.1 Aplikace „Stížnosti“

Parametr	Hodnota
Čistá doba vývoje aplikace	2,6 měsíce
Průměrná cena vývoje za „člověkoměsíc“	43 000 Kč
Celková cena	111 800 Kč

Tabulka 20: Ekonomické zhodnocení aplikace „Stížnosti“.

5.3.1.2 Aplikace „Mimořádné odměny“

Parametr	Hodnota
Čistá doba vývoje aplikace	1,96 měsíce
Průměrná cena vývoje za „člověkoměsíc“	43 000 Kč
Celková cena	84 280 Kč

Tabulka 21: Ekonomické zhodnocení aplikace „Mimořádné odměny“.

5.3.2 Ekonomické zhodnocení zavedení interních pravidel

Vyjádření finančních a časových úspor pro společnost v těchto navržených pravidlech je zřejmé, nicméně těžko uchopitelné. Ve své podstatě závisí na několika základních bodech:

- Dodržování stanovené architektury
- Frekvence použití stávajících funkcionalit
- Kvalita tvorby wiki stránek
- Kvalita refaktoringu

5.3.2.1 Pravidla pro vývoj

Díky stanoveným pravidlům je definován styl zdrojového kódu a jednotná architektura, kterou je možné opětovně používat. S každou novou aplikací jsou vytvářeny nové modely do databázové vrstvy poskytující funkcionality pro další projekty. To umožní rychlejší, levnější a kvalitnější vývoj v budoucnosti se snesitelnější udržovatelností. Tímto zavedením je možné dosáhnout úspor v rozmezí 20-50% v závislosti na velikosti vrstvy a složitosti aplikace.

Parametr	Hodnota
Dodržování architektury	10%
Psaní kvalitního zdrojového kódu	10%
Úspora času při dalším vývoji	20-50%

Tabulka 22: Ekonomické zhodnocení pravidel pro vývoj.

5.3.2.2 Zavedení wiki stránek

Zavedení wiki stránek společnosti přinese profesionální nástroj pro uchovávání a sdílení informací mezi členy v týmu, případně mezi jednotlivými firemními odděleními. Přesnější finanční zhodnocení je uvedeno v následující tabulce.

Parametr	Hodnota
Tvorba wiki stránek	15%
Údržba wiki stránek	5%
Úspora času při zaučení pracovníka, či dohledání informací	20-25%

Tabulka 23: Ekonomické zhodnoceni wiki stránek.

5.3.2.3 Refaktoring

Tvorba refaktoringu je velmi důležitou a prospěšnou činností poskytující levnější vývoj, který vychází i lepší kvality zdrojového kódu. Nicméně je velmi obtížné určit, v jakém poměru tento nástroj tyto úspory přináší.

6 Závěr

Cílem této práce bylo vytvoření několika komponent pro systém Adobe CQ5 ve společnosti Hartmann – RICO a.s. sídlem ve Veverské Bítýšce. Dílčím podcílem bylo vytvoření práce, která pomůže dalším programátorům s tvorbou komponent pro tento systém.

V kapitole „Teoretická východiska“ byly jednotlivé metody rozděleny na tři logické celky – metody použité k analýze současného stavu, analytické metody a metody ekonomické. V posledním bodu byl popsán systém Adobe CQ5, jeho struktura a metodická pravidla pro vývoj komponent. Dále byl čtenář seznámen s dostupnými vývojovými nástroji CRXDELite, CRXDE a Eclipse a zabudovanými nástroji uvnitř systému.

V analýze současného stavu jsme rozebrali postavení celé společnosti na trhu a postavění IT oddělení uvnitř firmy. K tomu byly využity metodiky a nástroje SWOT, RACI matice, případně GE matice. Analýza zachytila silné postavení společnosti na trhu zdravotních materiálů a doplňků. Dále byl analyzován systém Adobe CQ5 a porovnávám s konkurenčními nástroji, mezi nimiž je považován za nejlepší. V tomto ohledu byl zjištěn problém s dostupnou dokumentací k systému a malou uživatelskou komunitou, což se velmi negativně projevuje na kvalitu a dobu vývoje jednotlivých komponent.

Vlastní řešení této diplomové práce se zaměřuje na vytvoření kvalitních komponent dle potřeb a zadání firmy Hartmann-RICO a.s. Postupně byly vypracovány komponenty pro zadávání a přehled stížnosti a odpověď na stížnost. V další aplikaci pak byla vypracována aplikace „Mimořádné odměny“ a v ní komponenty poskytující vložení, přehled a detail žádostí. Další komponenta byla tvořena pro synchronizaci skupin mezi systémy Adobe CQ5 a interní databází RHstru. Ta není v této práci z kapacitních důvodů popsána a je uložena na přiloženém nosiči. V průběhu seznamování s firmou a firemním prostředím byly zjištěny některé chyby ve vývoji a bylo doporučeno jejich řešení. Jednalo se především o neexistenci jednotných vývojových pravidel a nevhodně používané architektury aplikací. V dalších bodech bylo doporučeno využití firemních wiki stránek a aplikace refaktoringu.

Všechny komponenty byly otestovány a nasazeny do firemního prostředí k použití. Testování probíhalo dle standardních metodik a na všech zařízeních, kde byla aplikace nasazena. V poslední kapitole bylo provedeno ekonomické zhodnocení všech výhod a nevýhod, která tyto návrhy s sebou přináší. Některá zhodnocení jsou prováděna pomocí sofistikovaných nástrojů, jiná jsou tvořena pouze odhadem, neboť není možné u nich tuto hodnotu určit jiným způsobem.

Některá navrhovaná řešení se podařilo již během tvorby diplomové práce uvést do firemní praxe. Především se jedná o psaní komentářů, lepší tvorba architektury a jednotnou úpravu zdrojového kódu.

Literatura

[1] FINAREA. *Historie obchodu: část 1/3* [online]. 25.3.2013. [cit. 2014-02-02]. Dostupné z: http://www.finarea.cz/ekonomicke-jevy/41-historie-obchodu-cast-1-3/

[2] LOON, Henrik Willem Van. *Dějiny lidstva*. 1.vyd. Praha: Nakladatelství Lidové noviny, 2000, 411 s. ISBN 80-710-6417-3.

[3] GRASSEOVÁ, Monika, Radek DUBEC a David ŘEHÁK. *Analýza v rukou manažera: 33 nejpoužívanějších metod strategického řízení*. Vyd. 1. Brno: Computer Press, 2010, 325 s. ISBN 978-80-251-2621-9.

[4] *PDCA cyklus*. Dostupné z: http://upload.wikimedia.org/wikipedia/commons/7/7a/PDCA_Cycle.svg

[5] Plan-Do-Check-Act (PDCA): Implementing New Ideas in a Controlled Way. *Http://www.mindtools.com/* [online]. 2014 [cit. 2014-04-08]. Dostupné z: http://www.mindtools.com/pages/article/newPPM_89.htm

[6] How to Develop AEM Projects Using Eclipse. *Dev.day.com* [online]. 2012 [cit. 2014-04-07]. Dostupné z: http://dev.day.com/docs/en/cq/aem-how-tos/development/howto-develop-aem-projects-using-eclipse.html

[7] Demingův cyklus PDCA a norma ISO/IEC 20000-1:2011. [online]. 12/2011. [cit. 2014-02-02]. Dostupné z: http://www.systemonline.cz/sprava-it/deminguv-cyklus-pdca.htm

[8] Matice odpovědnosti RACI. [online]. 1.5.2013. [cit. 2014-02-02]. Dostupné z: https://managementmania.com/cs/matice-odpovednosti-raci

[9] RACI tabulky a jak na ně. [online]. 25.10.2012. [cit. 2014-02-02]. Dostupné z: http://www.cleverandsmart.cz/raci-tabulky-a-jak-na-ne/

[10] Wireframe - drátěný model webu. *Shopsys* [online]. 2010 [cit. 2014-02-02]. Dostupné z: http://www.shopsys.cz/clanky/wireframe-drateny-model-webu/

[11] VONDRÁK, Ivo. *Úvod do softwarového inženýrství* [online]. Ostrava, 2002 [cit. 2014-02-02]. Dostupné z: http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

[12] Diagram tříd. *ORCA* [online]. [cit. 2014-02-02]. Dostupné z: http://orca.xf.cz/ooms/001/001.htm

[13] Agregace a kompozice. [online]. [cit. 2014-02-02]. Dostupné z: http://www.cs.vsb.cz/benes/vyuka/upr/texty/objekty/ch01s02s03.html

[14] SIKHA BAGUI, Richard Earp. *Database Design Using Entity-Relationship Diagrams*. London: Auerbach Publications, 2003. ISBN 978-020-3486-054.

[15] VALENTA, Michal. *DBS – Databázové modely* [online]. [cit. 2014-02-02]. Dostupné z: https://users.fit.cvut.cz/~valenta/doku/lib/exe/fetch.php/bivs/dbs_02_databazove_modely.pdf. Prezentace. FIT ČVUT.

[16] ZENDULKA, Jaroslav. *Konceptuální modelování a návrh databáze* [online]. [cit. 2014-02-02]. Dostupné z: http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_2.pdf. Prezentace. FIT VUT.

[17] RÁČEK, Jaroslav. *Funkční body*. Brno, 2013. Prezentace. Fakulta informatiky, MU.

[18] *COCOMO II Model Definition Manual* [online]. 2013 [cit. 2014-02-02]. Dostupné z: ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/modelman.pdf

[19] *What is Adobe CQ5?*. 2.2.2014. 2011. Dostupné z: http://www.youtube.com/watch?v=SJzGKSb-_JQ

[20] Day Software. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-02-02]. Dostupné z: http://en.wikipedia.org/wiki/Day_Software

[21] Adobe CQ. *Adobe CQ – Adobe Marketing Cloud* [online]. [cit. 2014-02-02]. Dostupné z: http://www.adobe.com/sea/products/cq.html

[22] Past, Present and Future of Web Content Management. ADOBE SYSTEMS. *Adobe Peak* [online]. 2012 [cit. 2014-02-02]. Dostupné z: http://peek.adobe.com/technology/past-present-and-future-of-wcm.html

[23] Exploring the CQ and its Platform. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/5-5/exploring/concepts.html

[24] CQ DAM Supported Formats. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/dam/documented_platforms.html

[25] User Administration and Security. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/administering/security.html

[26] How to Work With Packages. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/administering/package_manager.html

[27] Reporting. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/administering/reporting.html

[28] Segmentation. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/administering/segmentation.html

[29] Developing with CRXDE Lite. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/core/developing/development_tools/developing_with_crxde_lite.html

[30] CQ5 WCM Architect Guide. ADOBE SYSTEMS. [online]. 2008 [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/v5_1/html-resources/cq5_guide_architect/cq5_guide_architect.pdf

[31] Architecture Overview. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/current/exploring/architecture-overview.html

[32] The Basics. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/5-4/developing/the_basics.html

[33] Developing with CRXDE. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/crx/current/developing/development_tools/developing_with_crxde.html

[34] Components. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: https://dev.day.com/docs/en/cq/5-5/developing/components.html

[35] BURNETTE, Ed. *Eclipse IDE: pocket guide*. 1st ed. Sebastopol, CA: O'Reilly, c2005, ix, 117 p. ISBN 05-961-0065-5.

[36] RÁČEK, Jaroslav. *Plánovací a odhadovací nástroje*. Brno, 2013. Prezentace. Fakulta informatiky, MU.

[37] RÁČEK, Jaroslav. *Odhadování ceny SW - původní COCOMO*. Brno, 2013. Prezentace. Fakulta informatiky, MU.

[38] Developing with CRXDE. ADOBE SYSTEMS. [online]. [cit. 2014-02-02]. Dostupné z: http://dev.day.com/docs/en/cq/5-5/developing/developmenttools/developing_with_eclipse.html

[39] Management informačních systémů. *Vysoké učení technické v Brně* [online]. 2014 [cit. 2014-03-20]. Dostupné z: https://www.vutbr.cz/www_base/priloha.php?dpid=78217

[40] O nás. *Hartmann* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://cz.hartmann.info/o_nas.php

[41] Historie společnosti HARTMANN - RICO. *Hartmann* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://cz.hartmann.info/23098.php

[42] Výrobní závody. *Hartmann* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://cz.hartmann.info/19605.php

[43] Management kvality: Optimální kvalita výrobků a služeb. *Hartmann* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://cz.hartmann.info/management_kvality.php

[44] Ocenění: Zdraví v dobrých rukou. *Hartmann* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://cz.hartmann.info/oceneni.php

[45] LLC, Books. *Software Development Process: Waterfall Model, Computer Programming, Extreme Programming, Capability Maturity Model, Software Testing, Software Architecture, Code and Fix, Revision Control, Spiral Model, Iterative and Incremental Development*. Books Group: General Books, 2011. ISBN 9781156607671.

[46] IBM Lotus Notes / Domino Environment. *Geniusproject* [online]. 1997-2013 [cit. 2014-03-13]. Dostupné z: http://www.geniusproject.com/ibm-lotus-notes-domino

[47] Magic Quadrant for Web Content Management. *Gartner* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://www.gartner.com/technology/reprints.do?id=1-1HYYYL0&ct=130801&st=sg

[48] FOWLER, Martin. *Refactoring: improving the design of existing code*. Boston: Addison-Wesley, 1999, xxi, 431 s. ISBN 978-0-201-48567-7.

[49] Rigorózní práce: Odhady času a úsilí workflow projektů. SEDLÁČKOVÁ, Jana. *Informační systém Masarykovy univerzity* [online]. 2005 [cit. 2014-04-07]. Dostupné z: http://is.muni.cz/th/39202/fi_r/Rigorozni_prace_Sedlackova.txt

[50] Function Point Languages Table. *Www.qsm.com* [online]. 2014 [cit. 2014-04-07]. Dostupné z: http://www.qsm.com/resources/function-point-languages-table

[51] Confluence. *Atlkassian* [online]. 2013 [cit. 2014-04-07]. Dostupné z: https://www.atlassian.com/software/confluence/Confluence. *Atlassian* [online]. 2013 [cit. 2014-04-07]. Dostupné z: https://www.atlassian.com/software/confluence/

[52] XWiki Enterprise. *XWIKI* [online]. 2013 [cit. 2014-04-07]. Dostupné z: http://enterprise.xwiki.org/xwiki/bin/view/Main/WebHome

[53] XWiki Enterprise: License. *XWIKI* [online]. 2013 [cit. 2014-04-07]. Dostupné z: http://www.xwiki.org/xwiki/bin/view/Main/License

[54] Calibrating the COCOMO II Post-Architecture Model. SUNITA DEVNANI-CHULANI, Sunita, Bradford CLARK a Barry BOEHM. *Sunset* [online]. 1998 [cit. 2014-04-07]. Dostupné z: http://sunset.usc.edu/csse/TECHRPTS/1998/usccse98-502/CalPostArch.pdf

[55] Concepts of the AEM Touch-Optimized UI. BOEHM. *http://dev.day.com/* [online]. 2012 [cit. 2014-04-07]. Dostupné z: http://dev.day.com/docs/en/cq/5-6/touch-ui/touch-ui-concepts.html

[56] HAJÍČEK, Tomáš. *Matice GE* [online]. 19. 9. 2012 [cit. 2014-05-21]. Dostupné z: http://www.vseomarketingu.estranky.cz/clanky/marketing/matice-ge.html

Seznam příloh

Příloha A: Analýza současného stavu 1

Příloha B: Analýza současného stavu 2

Příloha C: Pravidla pro vývoj – Code Rules

Příloha D: Obsah DVD

Seznam obrázků

Obrázek 1: SWOT analýza. [3] 11
Obrázek 2: Demingův cyklu (PDCA cyklus). [4] 12
Obrázek 3: Příklad GE matice (Zpracováno dle [56]). 13
Obrázek 4: Příklad drátěného modelu a reálného webu. [10] 14
Obrázek 5: Příklad diagramu tříd (Zpracováno dle [12]). 17
Obrázek 6: Úvodní stránka Adobe CQ5. [23] 22
Obrázek 7: Funkcionalita pro administraci internetových stránek - Websites. [23] 23
Obrázek 8: Funkcionalita Digital assets. [23] 24
Obrázek 9: Funkcionalita Users & Groups. [25] 25
Obrázek 10: Tools – UI classic. [23] 26
Obrázek 11: Tools - Touch-Optimized UI. [55] 26
Obrázek 12: Integrované vývojové prostředí CRXDELite. [29] 28
Obrázek 13: Architektura Adobe CQ5. [31] 29
Obrázek 14: Vývoj aplikace pomocí prostředí CRXDE. [33] 30
Obrázek 15: Vývoj aplikace pomocí prostředí Eclipse. [6] 31
Obrázek 16: Dialog komponenty. [34] 33
Obrázek 17: Organizační struktura IT oddělení. 37
Obrázek 18: Zjednodušené schéma produkční architektury. 38
Obrázek 19: Etapy vývoje komponent. 39
Obrázek 20: Schéma nedokonale použité třívrstvé architektury. 41
Obrázek 21: EPC diagram pro testovací proces. 43
Obrázek 22: Postavení systému v konkurenčním prostředí. [47] 47
Obrázek 23: Diagram případů užití pro aplikaci „Stížnosti“. 52
Obrázek 24: Formulář vložení stížnosti a odpověď na stížnost. 53
Obrázek 25: Seznam vložených stížností do systému. 53
Obrázek 26: Diagram tříd aplikace „Stížnosti“. 54
Obrázek 27: ER diagram výsledného řešení. 55
Obrázek 28: Přiřazení objektů jednotlivým vrstvám a vzájemná komunikace mezi nimi. 58
Obrázek 29: Výsledná implementace komponenty „Přidání stížnosti“. 58
Obrázek 30: Výsledná implementace komponenty „Zobrazení vložení stížnosti“. 59
Obrázek 31: Výsledná implementace komponenty „Odpověď na stížnost“. 59
Obrázek 32: Diagram aktivit pro aplikaci „Mimořádné odměny“. 64
Obrázek 33: Formulář vložení žádosti o odměnu a detail žádosti o odměnu. 65
Obrázek 34: Návrh komponenty pro zobrazení vložených žádostí. 65

Obrázek 35: ER-diagram aplikace „Mimořádné odměny“. 66
Obrázek 36: Výsledná komponenta přidání a detail žádosti o odměnu. 68
Obrázek 37: Výsledná komponenta zobrazení žádostí o odměnu. 68

Seznam zdrojových kódů

Box 1: Příklad práce s výjimkami a logováním chyb. 60
Box 2: Příklad komentáře pro třídu EvaluationManager. 61
Box 3: Příklad komentáře metody pro vytvoření stížnosti v rozhraní IEvaluationManager. 61
Box 4: Příklad inline komentáře pro synchronizaci skupin ve třídě GroupSynchronzationManager...62

Seznam tabulek

Tabulka 1: GE matice pro jednoduchý případ (Zpracováno dle [8, 9]). 13
Tabulka 2: Parametry ovlivňující výpočet celkového úsilí pro vývoj aplikace. [54] 19
Tabulka 3: Vlastnosti komponenty. [34] 32
Tabulka 4: Vyhodnocení vlivů na atraktivitu trhu 35
Tabulka 5: Vyhodnocení vlivů na konkurenční prostředí 35
Tabulka 6: Tržní postavení společnosti v konkurenčním prostředí. 36
Tabulka 7: RACI matice odpovědností jednotlivých etap vývoje. 40
Tabulka 8: Vyhodnocení vlivů na atraktivitu trhu. 48
Tabulka 9: Vyhodnocení vlivů na konkurenční prostředí 48
Tabulka 10: Tržní postavení společnosti v konkurenčním prostředí. 48
Tabulka 11: SWOT analýza firmy Hartmann – RICO a.s. v oblasti IT 50
Tabulka 12: Výpočet funkčních bodů. 55
Tabulka 13: Přepočet jednoho funkčního bodu na počet příkazu daného jazyka. [50] 56
Tabulka 14: Zvolené parametry pro výpočet metody COCOMO. 56
Tabulka 15: Základní data o metrikách aplikace „Stížnosti“ 63
Tabulka 16: Výpočet funkčních bodů. 66
Tabulka 17: Zvolené parametry pro výpočet metody COCOMO. 67
Tabulka 18: Základní data o metrikách aplikace „Mimořádné odměny“. 69
Tabulka 19: Licenční podmínky pro cenotvorbu za jednotlivé licence. [51] 72
Tabulka 20: Ekonomické zhodnocení aplikace „Stížnosti“. 74
Tabulka 21: Ekonomické zhodnocení aplikace „Mimořádné odměny“. 74
Tabulka 22: Ekonomické zhodnocení pravidel pro vývoj. 75
Tabulka 23: Ekonomické zhodnoceni wiki stránek. 75

Příloha A

Analýza současného stavu 1

Analýza současného stavu

*Povinné pole

Struktura IT oddělení *
Kolik členů obsahuje IT oddělení?
20

Z toho vývojářů *
Kolik programátorů IT oddělení má k dispozici (+- včetně agentur(přibližně))
6

Jaká je průměrná cena hodiny programátora při vývoji komponenty?
Pokud není tajné, bude použito v ekonomických zhodnocení, stačí hrubý odhad. (není povinná)
400

Jsou při projektovém řízení použity nějaké nástroje? *
Jedná se o nástroje pro řízení času a financí nutné pro vývoj projektu. Vetšinou se používají pro velké projekty

- [ ] Nepoužíváme
- [ ] MS Project Studio
- [ ] Open Proj
- [ ] Gantter
- [ ] TeamBox
- [ ] Rally Community Edition
- [ ] FreedCamp
- [x] Jiné: P4D (Project4Domino)

Jsou ve firmě vypracována jednotná pravidla pro vývoj? *
Jedná se většinou o dokument, který všechny pracovníky (programátory) zavazuje k jednotným pravidlům pro vývoj.

- ( ) Ano
- ( ) Ne
- (•) Jiné: Pouze z části

Které etapy vývoje jsou v Hartmannu obsaženy? *
Seznam vychází z vodopádového modelu.

- [x] Specifikace požadavků
- [x] Návrh
- [x] Implementace
- [x] Integrace
- [x] Testování a ladění (validace)
- [x] Instalace
- [x] Údržba
- [ ] Jiné:

Jsou při vývoji využity návrhové vzory *
Návrhový vzor je řešení, které se v historii v dané problematice nejvíce osvědčilo. Tzv. "Best-practise".

- (•) Ano
- ( ) Ne
- ( ) Jiné:

Jakou dominantní architekturu v systému používáme?

- ( ) MVC/MVP
- (•) Třívrstvou architekturu
- ( ) Architekturu neřešíme, přebíráme systémové.
- ( ) Jiné:

Provádíme refaktoring zdrojového kódu? *
Refaktoring je zpětná úprava zdrojového kódu. Používá se v průběhu vývoje v případech, kdy se objeví analytická chyba. Zabraňuje tvorbě nesystematických záplat.

- ( ) Ano
- ( ) Ne
- (•) Jiné: Ano, s tímto v rámci webc

Jaké prvky pro komunikaci a sdílení informací jsou použity? *

- [x] Email
- [x] Interní Komunikátor
- [ ] Skype
- [ ] Wiki stránky
- [x] Tiketovací systém
- [x] Sdílení dokumentů na síťovém disku
- [x] Jiné: SKYPE pouze neoficiálně

Jaké typy dokumentací jsou tvořeny a udržovány? *

- [ ] Wiki dokumentace
- [x] JavaDoc dokumentace
- [x] Dokumenty na sdíleném disku
- [ ] Jiné:

V jakém rozsahu je tvořena JavaDoc dokumentace v komponentách? *

- ( ) <10%
- (•) 11%<25%
- ( ) 26%<50%
- ( ) 51%<75%
- ( ) více než 75%

V jakém rozsahu je dokumentace tvořena? *

Jedná se o dokumentace potřebné pro vývoj a nastavení prostředí pro vývoj (nastavení prostředí, serverů, dokumentace komponent, interních pravidel, atd..).

- ( ) <10%
- (•) 11%<25%
- ( ) 26%<50%
- ( ) 51%<75%
- ( ) více než 75%

Odeslat

Příloha B

Analýza současného stavu 2

Analýza současného stavu 2

*Povinné pole

Definujte jednoduše strukturu IT oddělení. *
Například 1 vedoucí, 1 projektový manažer, 6 vývojářů, XX administrativních pracovníků... manažer spadá pod vedoucího a řídí všechny programátory, atd..

1 vedoucí IS
3 manažeři (spadající pod vedoucího IS)

- každý ze 3 manažerů má pod sebou 6-8 lidí (pracovníky IS - programátoři, správci systému, HW podpora, SAP podpora)

Nejčastějšími zadavateli aplikací/komponent do CQ5 jsou? *
Zadavatel je zároveň i plátce zakázky.

- [x] Ostatní organizační jednotky firmy (obchodní, finanční, personální)
- [x] Jiné firmy - nemocnice, lékárny, atd...
- [ ] Jiné:

Kolik procent práce IT oddělení tvoří aplikace pro jiné firmy (pokud jsou tvořeny)? *
Stačí hrubý odhad.

40-50%

Kdo nese přímou odpovědnost za vývoj komponent? *
Vedoucí IT oddělení, projektový manažer - otázka souvisí se strukturou oddělení.

projektový manažer

Mezi jakými systémy bylo vybíráno při výběru CQ5? *
Jaké jiné systémy/frameworky byly zvažovány pro nazasení do firmy?

"RedDot (OpenText)
Microsoft

- víceméně všechna velká korporátní řešení

http://www.gartner.com/technology/reprints.do?id=1-1HYYYL0&ct=130801&st=sg"

Definujte procesní etapy vývoje komponenty vůči zadavateli. *

Například - zadání od konkrétního oddělení, počáteční konzultace nad přínosem, finanční rozvaha, konečné rozhodnutí, vývoj, konzultace na postupech, dopracování změn, předání, údržba, atd...

"procesy nelze zcela popsat, protože IT již dostává od ostatních oddělení připravené materiály pro vývoj

za IT tedy proběhne analýza (technická), sepsání manažerské a technické dokumentace a následně probíhá vývoj, testování a předávání produktu, následně údržba a případné rozšiřování"

Jaké prvky a nástroje jsou používány při testování? *

- [x] Aplikace testuje sám vývojář
- [x] Máme vlastního testera (testery)
- [ ] Používáme automatické testy (jUnit)
- [ ] Používáme automatické testy integrované v prohlížeči (Selenium)
- [x] Jiné: před spuštěním testuje za

Pro které závody v ČR tvoří IT oddělení ve Veverské Bítýšce technologickou podporu?

Má každý závod svoje IT oddělení, nebo závod ve Veverské Bítýšce dělá IT podporu i ostatním závodům?

- [x] Celá Česká republika
- [ ] Veverská Bítýška
- [ ] Havlíčkův Brod
- [ ] Chvalkovice
- [x] Jiné: Slovensko a Bulharsko

Odeslat

Příloha C

Pravidla pro vývoj – Code Rules

Pravidla pro vývoj aplikací v jazyce Java

ve společnosti Hartmann – RICO a.s.

verze 1.0

Autor: Bc. Jan Pešava

Email: xpesav00@fbm.std.vutbr.cz

Datum: 27. 3. 2014

Úvod

Tento materiál vychází z velké části ze standardního dokumentu vydaného firmou Sun – „Java Code Convention“, který popisuje základní pravidla pro formální úpravu zdrojového kódu v jazyce Java. Tato pravidla jsou přebrána, doplněna a upravena na technologické zázemí firmy.

1.1 Cíle jednotných pravidel

- Sjednotit výsledný design zdrojových kódů
- Zpřehlednit a zjednodušit jejich další používání
- Snížení nákladů na údržbu současných zdrojových kódů
- Definování jednotného jazyka pro pojmenování entit a komentářů

2 Úprava zdrojového kódu

V této kapitole budou probrány základní prvky požadovaného vzhledu zdrojového kódu. Jedná se například o pojmenování souborů, jejich organizace, odsazení jednotlivých bloků uvnitř zdrojového kódu, psaní komentářů, atd.

2.1 Pojmenování souborů

Soubory jsou pojmenovány tak, aby jasně vyjadřovaly svůj účel. Stanoveným jazykem pro pojmenovávání souborů je **anglický jazyk.**

Oddělování slov v názvech bude prováděno velkými písmeny.

- `TestedMainClassName – dobře`
- `TMCN - špatně`
- `Tested_Main_Class_Name – špatně`
- `testedmainclassname - špatně`
- `Tested-Main-Class-Name – špatně`

Pojmenování souboru by mělo již po přečtení názvu naznačovat, o jaký typ souboru se jedná, zda se jedná o třídu, rozhraní, výčet, servlet, atd.

2.1.1 Pojmenování tříd

Třídy jsou pojmenovány vždy velkým počátečním písmenem a dále se jejich pojmenování řídí dle výše stanovaných pravidel.

- `TestedMainClassName.java – dobře`
- `testedMainClassName.java – špatně`

2.1.2 Pojmenování rozhraní

Z názvu souboru by mělo být patrné, že se jedná o rozhraní. To by mělo být uvedené případným znakem v názvu souboru. Většinou se uvádí písmeno „I“ jako „Interface“.

- `ITestedInterfaceName.java – dobře`
- `TestedInterfaceName.java – špatně`
- `itestedinterfacename.java – špatně`

2.1.3 Pojmenování výčtu

Výčet je pojmenován dle níže uvedeného příkladu.

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
```

2.1.4 Pojmenování balíčků

Název jednotlivých balíčků je psán pouze malými písmeny. Vychází to ze zavedených standardů a programátorských zvyků.

- `packagename – dobře`
- `packageName – špatně`
- `PackageName – špatně`

Výsledná cesta k balíčku je pak definována například takto:

- `import basepackage.nextpackage.packagename;`

2.2 Organizace souborů

Soubory by se měly skládat z oblastí oddělených prázdnými řádky, případně komentáři identifikujícími každou sekci.

Soubory delší než 2000 řádků jsou „težkopádné“ a je vhodné se jim vyhnout. Zde je na místě uvažovat o lepším architektonickém návrhu.

2.2.1 Zdrojové soubory

Každý zdrojový soubor obsahuje pouze jednu třídu nebo rozhraní. Dle specifikace je možné použít soukromou třídu nebo rozhraní ve stejném souboru s veřejnou třídou. Tento postup je nutné dělat s obezřetností a jen ze závažných důvodů.

2.2.1.1 Každý soubor má následující uspořádání:

2.2.1.1.1 Počáteční komentář

Každý soubor by měl začínat komentářem ve stylu jazyka c a obsahovat základní popis o verzi, autorovi, datu vytvoření, editace, atd… Tyto komentáře nejsou dokumentační.

```
/*
* Classname
*
* Version info
*
* Copyright notice
*/
```

2.2.1.1.2 Název balíčku a sekce importů

Jedná se o první nekomentovaný příkaz, který musí soubor obsahovat. Ihned za ním následuje příkaz importů. Příklad:

```
package java.awt;
import java.awt.peer.CanvasPeer;
```

2.2.1.1.3 Deklarace třídy, rozhraní

Následující tabulka popisuje části třídy nebo rozhraní tak, jak by měly být navrženy.

pořadí	Část třídy/rozhraní	Poznámka
1	Dokumentační komentář třídy, rozhraní	Detailněji popsána v kapitole 2.4.
2	Příkaz class nebo interface	-
3	Definice statických proměnných	Na začátku jsou definovány veřejné proměnné, následně chráněné a soukromé.
4	Definice proměnných objektu	Na začátku jsou definovány veřejné proměnné, následně chráněné a soukromé.
5	Konstruktor	-
6	Metody	Definovány metody s patřičnými komentáři.

2.3 Odsazení

Pro odsazení vnořených bloků je stanovena jednotka 4 mezer. V případě potřeby je možné používat pro odsazení i tabulátor, který má velikost 8 mezer.

2.3.1 Velikost řádků

Velikost řádku je maximálně stanovena na 80 znaků. Větší délka tvoří výsledný zdrojový kód nepřehledným.

2.3.2 Zalamování řádků

Pokud je výraz delší než 80 znaků, je možné ho zalomit na více řádků. Zalamování se řídí těmito základními principy:

- Zalomení za čárkou
- Zalomení před operátorem
- Zarovnání nového řádku se začátkem výrazu na stejné úrovni s předchozím řádkem

Příklady:

```
function(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

longName1 = longName2 * (longName3 + longName4 - longName5)
           + 4 * longname6; // PREFER

//CONVENTIONAL INDENTATION
someMethod(int anArg, Object anotherArg, String yetAnotherArg,
         Object andStillAnother) {
         ...
}
```

2.4 Komentáře

Syntaxe jazyka Java má dva druhy komentářů:

- **Implementační komentáře:** slouží pouze pro vysvětlení implementačních detailů ve zdrojovém kódu, nejsou použiti pro tvorbu generované dokumentace
- **Dokumentační komentáře:** komentují zdrojový kód a jsou při generování dokumentace do ní zahrnuty.

Povinností je komentovat každý zdrojový kód.

2.4.1 Formát implementačních komentářů

Blokové komentáře slouží ve zdrojovém kódu k popisu vnitřní logiky jednotlivých funkcionalit při jejich nahlížení. Můžeme je dělit na několik částí. Vystačíme však se dvěma základními – blokový komentář a řádkový komentář. Není možné je používat pro komentování public funkcí, tříd a rozhraní. Pro tyto případy jsou k dispozici dokumentační komentáře. Více v kapitole 2.4.2.

2.4.1.1 Blokový komentář

Je uvozen znaky `/*` a ukončen `*/`. Všechny znaku uvnitř jsou považovány za komentář. Je možné ho použít i jako vložený řádkový komentář.

```
/*
* Here is a block comment with some very special
* formatting that I want indent(1) to ignore.
*
* one
* two
* three
*/
```

Případně

```
/* Here is a block comment with some very special */
```

2.4.1.2 Řádkový komentář

Řádkový komentář je uvozen znaky „//" a platí vždy až do konce řádku. Všechno za ním je považováno za komentář. Jedná se o rychlou variantu tvorby komentářů. Vhodné pro interní komentování funkcí. Nepoužívat pro komentování více řádků. K tomu slouží blokové komentáře.

Příklady:

```
// Here is a line comment
// Variables Declaration
// Get data from database
// Send Email to users
```

2.4.2 Formát dokumentačních komentářů

Dokumentačními komentáři budou komentovány všechny vytvořené třídy včetně všech metod označené jako veřejné, případně chráněné. Silně doporučeno je však komentovat všechny metody bez ohledu na viditelnost.

Tyto komentáře jsou uvozeny znakem `/**` a ukončeny `*/`. Vše mezi tím je bráno jako dokumentační komentář.

Příklad:

```
/**
* The Example class provides...
*/
```

Do komentářů je možné vkládat notace s předem danou sémantikou.

- **@author** (třída nebo rozhraní, **povinné**)
- **@version** (třída nebo rozhraní, **povinné**.)
- **@param** (metoda nebo konstruktor, **povinné**)
- **@return** (jen u metod, **povinné, pokud vrací hodnotu**)
- **@exception** (@throws je synonymum přidané v Javadoc 1.2)
- **@see** (určuje relaci k jiným komponentám, třídám, atd…)
- **@since**
- **@serial** (nebo @serialField nebo @serialData)
- **@deprecated** (označování zastaralých funkcí)

2.4.2.1 Komentování tříd, rozhraní

Do komentářů je také možné vkládat HTML znaky. Například se v komentářích může hodit vytváření seznamů značkami <ul> a <li>, případně tvorba odstavce <p> a jiné.

```
/**
 * All coordinates which appear as arguments to the methods of this
 * Graphics object are considered relative to the translation origin
 * of this Graphics object prior to the invocation of the method.
 * All rendering operations modify only pixels which lie within the
 * area bounded by both the current clip of the graphics context
 * and the extents of the Component used to create the Graphics
object.
 *
 * @author      Sami Shaio
 * @author      Arthur van Hoff
```

```
 * @version      %I%, %G%
 * @since        1.0
 */
public abstract class Graphics {
}
```

`%I%` - definuje verzi třídy, kterou při každé editace zvýší. Například z 1.1 na 1.2, atd.

`%G%` je datum ve formátu `mm/dd/yy`.

2.4.2.2 Komentování metod

Povinností je komentovat dokumentačními komentáři všechny veřejné a chráněné metody. U soukromých metod je možné si vybrat z obou možných forem komentování – implementační a dokumentační komentáře.

```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param  url  an absolute URL giving the base location of the
 *         image
 * @param  name the location of the image, relative to the url
 *         argument
 * @return the image at the specified URL
 * @see    Image
 */
 public Image getImage(URL url, String name) {
        try {
            return getImage(new URL(url, name));
        } catch (MalformedURLException e) {
            return null;
        }
 }
```

2.4.3 Dodatečné ustanovení

Pro detailnější seznámení s komentováním v jazyce Java je možné navštívit web společnosti ORACLE: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

2.5 Deklarace

Tato kapitola popisuje pravidla pro definici příkazů, jejich pojmenování a umístění v rámci celého zdrojového kódu.

2.5.1 Příkaz na řádek

Doporučením je dávat jeden příkaz na jeden řádek. Tyto příkazy pomáhají lepší orientaci v kódu.

Příklad:

```
int level; // indentation level
int size; // size of table
```

místo

```
int level, size;    // raději nepoužívat
```

2.5.2 Umístění deklarace

Jednotlivé proměnné jsou deklarovány ihned na začátku bloku. Blok je definován složenými závorkami – „{}". Není vhodné čekat až na jejich první použití. Toto pravidlo slouží pro kvalitnější přehlednost proměnných v kódu.

```
void MyMethod() {
      int int1; // beginning of method block
      if (condition) {
            int int2; // beginning of "if" block
            ...
      }
}
```

Pozor na deklaraci stejných proměnných na různých vrstvách. Této konstrukci kódu se pokuste vyhnout.

```
int count;
...
func() {
      if (condition) {
            int count; // špatně!
            ...
      }
      ...
}
```

2.5.3 Inicializace

Pokud je to možné, proměnná je inicializována ihned po její deklaraci. Jedinou výjimkou jsou případy, ve kterých inicializace závisí na předchozích výpočtech.

2.5.4 Deklarace tříd a rozhraní

Pro formátování deklarací tříd a rozhraní jsou stanovena následující pravidla:

- Nejsou vloženy žádné mezery mezi názvem metody a kulatou závorkou. „emptyMethod()"
- Otevírající složená závorka bloku třídy/rozhraní začíná na stejném řádku jako deklarace.
- Uzavírající závorka bloku třídy/rozhraní končí na samostatném řádku a ve stejné horizontální úrovni.

```
class Sample extends Object {
      int ivar1;
      int ivar2;

      Sample(int i, int j) {
            ivar1 = i;
```

```
            ivar2 = j;
      }

      int emptyMethod() {}
      ...
}
```

2.5.5 Pojmenování atributů a metod

Atributy a metody budou pojmenovány anglickým jazykem a malými písmeny mimo oddělení slov písmeny velkými. Pojmenování atributů by mělo odrážet obsah proměnné. Pojmenování metod by mělo odrážet jejich funkční aktivitu.

Správný příklad:

```
int cislo1;
int cislo2;

public int soucetCisla(int cislo1, int cislo2) {
      return cislo1 + cislo2;
}
```

Nesprávný příklad:

```
int c1;
int c2;

public int sc(int c1, int c2) {
      return c1 + c2;
}
```

2.6 Příkazy

V této části jsou definovány jednotlivé příkazy v jazyce Java a určeno jejich výsledné formátování.

2.6.1 Jednoduché příkazy

Jeden příkaz je uveden na jednom řádku.

Příklad:

```
argv++; argc--; // špatně!
argv++;         // dobře
argc--;         // dobře
```

2.6.2 Složené příkazy

Složené příkazy vždy dávat do bloku – složených závorek.

2.6.3 Příkaz „return“

Příkaz return nemusí být uveden mezi složenými závorkami.

Příklad:

```
return;
return myDisk.size();
```

```
return (size ? size : defaultSize);
```

2.6.4 Větvení

Příkazy vykonávané po každém větvení musí být vždy uvedeny ve složených závorkách.

Příklad:

```
if (condition) //nepovoleno, nutno použít - {}!
      statement;
```

2.6.4.1 Příkaz if

Příklad:

```
if (condition) {
      statements;
}
```

2.6.4.2 Příkaz if-else

Příklad:

```
if (condition) {
      statements;
} else {
      statements;
}
```

2.6.4.3 Příkaz if-else-if-else

Příklad:

```
if (condition) {
      statements;
} else if (condition) {
      statements;
} else if (condition) {
      statements;
}
```

2.6.4.4 Příkaz switch

Příklad:

```
switch (condition) {
      case ABC:
            statements;
            /* falls through */
      case DEF:
            statements;
            break;
      case XYZ:
            statements;
            break;
      default:
            statements;
            break;
}
```

```
switch (condition) {
case ABC:
      statements;
      /* falls through */
case DEF:
      statements;
      break;
case XYZ:
      statements;
      break;
default:
      statements;
      break;
}
```

2.6.5 Cykly

V této části je popsána konvence cyklu – for, while a do-while.

2.6.5.1 Cyklus for

Příklad:

```
for (initialization; condition; update) {
      statements;
}
```

2.6.5.2 Cyklus while

Příklad:

```
while (condition) {
      statements;
}
```

2.6.5.3 Cyklus do-while

Příklad:

```
do {
      statements;
} while (condition);
```

2.6.6 Try-catch

Příkaz try-catch by měl mít následující formát:

Příklad:

```
try {
      statements;
} catch (ExceptionClass e) {
      statements;
}
```

2.7 Odřádkování

Vkládání prázdných řádků mezi jednotlivé příkazy ve zdrojovém kódu pomáhá k tvorbě sekcí a tím zlepšuje výslednou přehlednost. Toto stylování je ponecháno na estetickém uvážení každého programátora.

2.8 Konvence pro pojmenování

Typ identifikátoru	Pravidla pro pojmenování	Příklad
Třída	Třída musí být pojmenována podstatným jménem s velkým prvním písmenem každého slova v názvu. Ostatní jsou malá, mimo zavedených zkratek – URL, HTML, atd…	`class Raster;` `class ImageSprite;`
Rozhraní	Podobná pravidla jako u tříd. Uvozeno písmenem „I“.	`interface IRasterDelegate;` `interface IStoring;`

Metoda	Metoda musí být pojmenována slovesem, první písmeno je malé. Jednotlivá slova jsou rozčleněna velkým písmenem.	`run();`
		`runFast();`
		`getBackground();`

Proměnná	Proměnná musí být pojmenována podstatným jménem, jinak stejná pravidla jako u pojmenování metod.	`int i;`
		`char *wordPointer;`
		`float cubeWidth;`

Konstanta	Všechna písmena jsou psána velkými písmeny, oddělenými podtržítkem.	`int MIN_WIDTH = 4;`
		`int MAX_WIDTH = 999;`
		`int GET_THE_CPU = 1;`

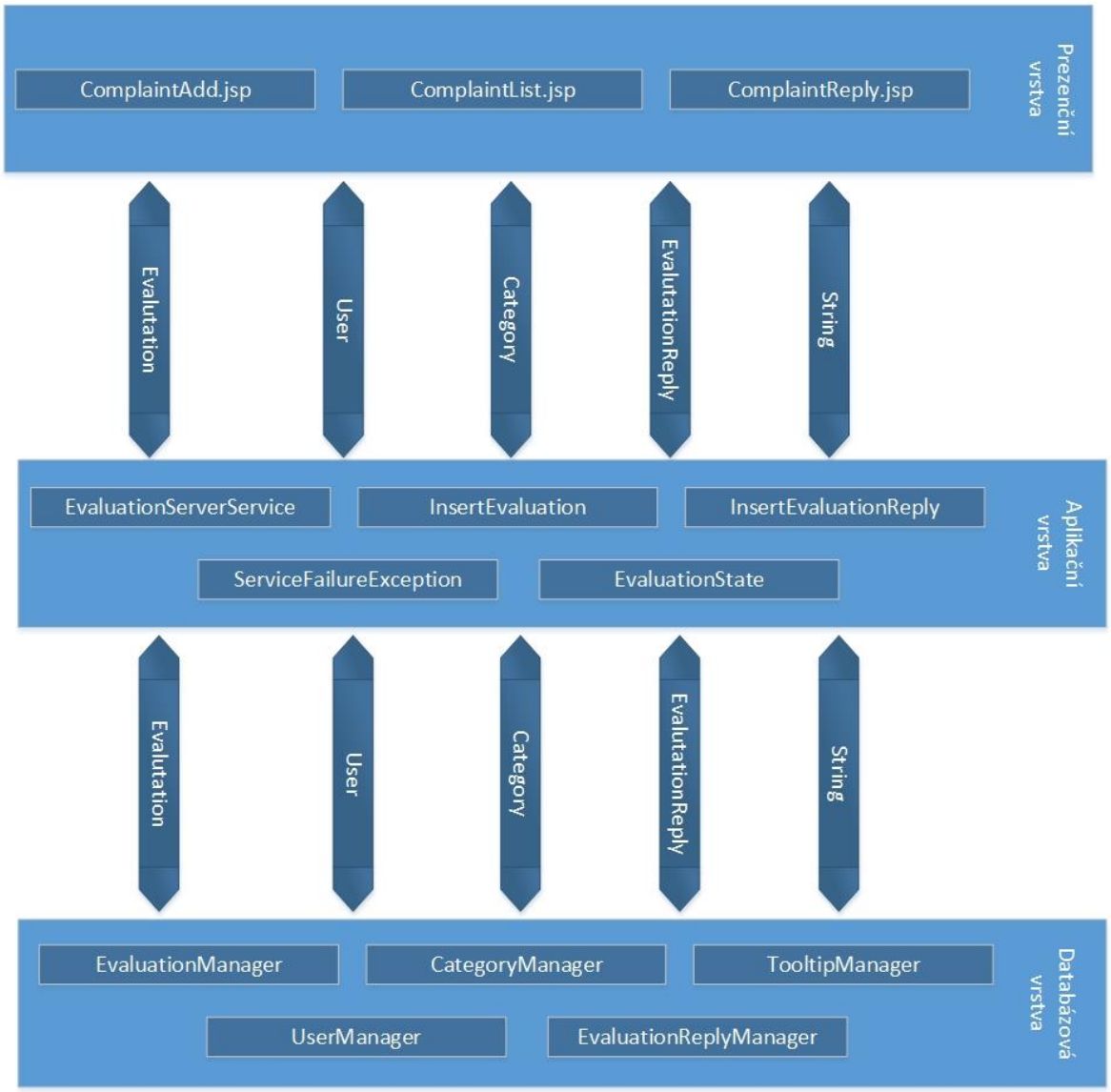
3 Architektura systému

V této kapitole bude popsána architektura systému a jednotlivých vrstev komponent na uložišti GIT.

3.1 Popis architektury

Cílem architektury je udržování následujících vrstev:

- **Prezentační vrstva:** obsahuje soubory pro zobrazování dat z databáze. Jedná se o soubory s příponou „jsp“.
- **Aplikační vrstva:** Business logika systému. Získává data z databázové vrstvy a poskytuje je vrstvě prezenční.
- **Databázová vrstva:** získává data z relační databáze a poskytuje je vrstvě aplikační.



Obrázek 38: Příklad rozdělení vrstev v rámci komponenty

Cílem tvorby jednotlivých vrstev je definování povinností a odpovědností jednotlivým celkům. Ty mají v gesci určité funkcionality a je možné tak definovat služby, které budou k dispozici jiným vrstvám. Tím je možné zabránit replikacím zdrojového kódu, lepší správě a kvalitnějšímu a komfortnějšímu vývoji.

3.2 Popis stromové struktury složek

V této kapitole budou probrány některé důležité složky v uložišti.

3.2.1 Hartmann-default-bundle

V této složce by měly být uloženy všechny soubory v napsané v programovacím jazyce Java.

3.2.1.1 Složka Library

Umístění: /src/main/java/info/hartmann/

Složka je určena pro uložení vytvořených modelů a tříd, které mohou využívat i ostatní programátoři při vývoji jiných aplikací. Složka je dále rozdělena do následujících podsložek:

- **exceptions** – slouží pro definici potřebných a neexistujících výjimek.
- **general** – slouží pro uložení knihoven a modelů využitelných napříč systémy
- **intranet** – definuje knihovny a modely pro prací s databází intranet – 1 složka zajišťuje práci s jednou tabulkou v databázi.
- **rhstru** – definuje knihovny a modely pro prací s databází rhstru – 1 složka zajišťuje práci s jednou tabulkou v databázi.
- **servlets** – uložiště servletů poskytujících podpůrné služby například pro autocomplete nápověd v jQuery, atd.
- **wrappers** – bude upřesněno.

3.2.1.2 Složka Intranet

Umístění: /src/main/java/info/hartmann/

Jsou zde uloženy v jednotlivých složkách třídy, servlety a další části zdrojového kódu, které se týkají pouze samotné aplikace a není možné je použít pro aplikace jiné. Většinou se jedná o servlety, případně třídy definující komunikaci se samotnou komponentou.

3.2.2 Hatman-default-components

V této složce jsou uloženy definice samotných komponent. Jedná se o xml a jsp soubory. Jiné soubory, zvláště java soubory, by zde uloženy být neměly.

3.2.3 Hartman-default-content

Specifikace této složky bude ještě upřesněna.

Příloha D

Obsah DVD

Popis DVD

soubor `README.txt`

Zdrojové kódy

adresář `/source-app`

Diplomová práce

soubor `Master's Thesis.pdf`