



Příloha ZD14 - Požadavky na popis systému

k č.j. ČÚZK-10685/2020

Obsah

1	Účel dokumentu	4
2	Obecné požadavky na dokumentaci	4
2.1	Průběžná aktualizace	4
2.2	Správa verzí dokumentace	4
2.3	Typy dokumentů	4
3	Základní popis aplikace	5
3.1	Účel aplikace	5
3.2	Testovací a produkční prostředí	5
3.3	Architektura	5
3.4	UI Design	5
4	Analýza	5
4.1	Procesní model (BMPN)	5
4.2	Případy užití (Use Cases, UML 2)	6
4.3	Diagramy aktivit (UML 2)	8
4.4	Doménový model (Class Diagram, UML 2)	8
4.5	Sekvenční diagramy (UML 2)	9
4.6	Stavové diagramy (UML 2)	9
4.7	Diagramy komponent (UML 2)	10
4.8	Diagramy nasazení (UML 2)	10
4.9	Požadavky na UI	11
4.10	Požadavky na výkonnost aplikace	11
4.11	Požadavky na HW a SW	11
4.12	Doplňující informace	11
5	Databáze	11
5.1	Datový model	11
5.2	Komentáře	12
5.3	Uložené procedury, funkce, sekvence	12
5.4	Aktualizace dokumentace	13
6	Bezpečnost	13
6.1	Autentizace	13
6.2	Autorizace	13
6.3	Zabezpečení infrastruktury	13
6.4	Audit přístupu k datům	13
7	Programátorská dokumentace	13
7.1	Použité technologie	13

7.2	Nástroje používané pro vývoj	13
7.3	Lokální vývojové prostředí	13
7.4	Organizace kódu	14
7.5	Pravidla pro psaní kódu	14
7.5.1	Jmenné konvence	14
7.5.2	Komentáře	14
7.5.3	Formátování	16
7.5.4	Internacionalizace a lokalizace	16
7.6	Pravidla pro tvorbu UI	16
7.7	Verzování	16
7.8	Testování	16
8	Verzování	17
8.1	Verzování zdrojových kódů	17
8.2	Verzování databázových změn	17
8.3	Verzování dokumentace	17
9	Buildy a nasazování aplikace (Release and Deployment)	17
9.1	Nasazování verzí na testovací a produkční prostředí	17
10	Integrace s dalšími systémy	17
11	Popis API	18
11.1	Popisy vystavených webových služeb	18
11.2	Popisy dalších poskytnutých rozhraní	18

1 Účel dokumentu

Účelem tohoto dokumentu je popsat souhrn požadavků na dokumentaci z hlediska popisu a údržby systému.

2 Obecné požadavky na dokumentaci

Zadavatel požaduje, aby Dodavatel vytvářel, aktualizoval a kontroloval dokumentaci IS DMVS podle následujících principů:

- dokumentace celého systému bude komplexní, kompaktní a konzistentní,
- dokumentace bude vytvářena strukturovaně podle podrobnosti, každá úroveň struktury bude obsahovat přiměřenou úroveň detailů k popisované problematice.
- navigace a orientace v dokumentaci musí být jednoduchá a srozumitelná,
- v dokumentaci musí jít vyhledávat a musí obsahovat rejstříky pojmů,
- v dokumentaci musí být integrovány všechny části, včetně datového modelu, UML, ArchiMate apod.,
- dokumentace musí být provázána aktivními odkazy, umožňující jednoduchou navigaci mezi jejími částmi.

2.1 Průběžná aktualizace

Aby se kdokoli s příslušnou kvalifikací mohl v aplikaci zorientovat a provádět v ní změny, je nezbytně nutné průběžně dokumentaci udržovat. Dokumentace bude v jednotlivých úrovních aktualizována tak, aby byla v dané podrobnosti stále aktuální. Pro aktualizace musí být stanovena vnitřní pravidla tak, aby nebylo možné je opomíjet nebo obcházet.

2.2 Správa verzí dokumentace

Změny v dokumentaci budou kvůli zachování celistvosti dokumentů zapisovány přímo do jednotlivých sekcí dokumentů a nikoli do dodatků.

Hlavička každého textového dokumentu bude zahrnovat historii změn obsahující číslo verze, datum, autora a stručný popis změny.

Příklad historie změn:

Verze	Datum	Autor	Popis
1.0	23. 5. 2020	Jiří Němec	Úvodní verze dokumentu
1.1	27. 6. 2020	Petra Poláková	Doplnění popisu webové služby XY
2.0	10. 7. 2020	Karel Uher	Kompletní revize dokumentu

Verze dokumentů budou zálohovány spolu se zdrojovými kódy aplikace v repozitáři softwaru použitého pro správu verzí (Git, TFS, ..., viz kapitola [Verzování](#)).

2.3 Typy dokumentů

Všechny dokumenty budou vytvořeny ve standardních běžně používaných formátech, aby je mohl Zadavatel bez problémů prohlížet nebo editovat:

- standardní formáty dle projektové kanceláře,
- obrázky nezahrnuté v dokumentech (PNG, JPG),
- ostatní – Enterprise Architect, WIKI stránky apod. – po dohodě Dodavatele se Zadavatelem.

3 Základní popis aplikace

3.1 Účel aplikace

V této kapitole bude popsán účel aplikace spolu s hlavními scénáři jejího použití. Musí být jasně a přehledně uvedeno, jaké služby/funkce aplikace uživatelům poskytuje.

Slovní popis bude pro snadnější porozumění doplněn ilustrujícími schématy a screenshoty.

3.2 Testovací a produkční prostředí

V tomto oddíle budou uvedeny odkazy na spuštění aplikace v testovacím a produkčním prostředí se stručným popisem rozdílů obou režimů.

3.3 Architektura

Popis architektury dodávaného systému bude vytvářen ve všech vrstvách architektury dle platné metodiky OHA MVČR a s použitím prakticky ověřených a uznávaných standardů: mezinárodně uznávaného rámce pro řízení tvorby enterprise architektury TOGAF, jazyka ArchiMate® pro modelování architektonického obsahu a dále jazyka UML užitého pro detailní návrh řešení. Dále bude popsáno, jakým způsobem a přes jaké protokoly spolu jednotlivé vrstvy komunikují.

3.4 UI Design

Kapitola o UI designu bude popisovat způsob realizace uživatelského rozhraní (web, desktop, ...). Dále budou specifikovány principy použité při návrhu uživatelského rozhraní (např. Material Design, responzivita v případě webové aplikace atd.). Bude shrnuto ovládání aplikace a její základní ovládací prvky (pohyb mezi jednotlivými funkcemi, menu, dialogy, filtry, globálně sdílené prvky, atd.).

Popis bude doplněn screenshoty uživatelského rozhraní.

4 Analýza

Analýza změn i nových funkcí musí být prováděna pomocí modelovacího jazyka UML 2.0. Pro tvorbu analýzy se předpokládá využití standardních modelovacích nástrojů, např. Enterprise Architect. Volba vhodného analytického nástroje bude konzultována se Zadavatelem.

Zadavatel požaduje předávání výstupů analýzy a designu i ve formě, která umožní interaktivní procházení (například export repository pro EA v XML formátu), nebo musí umožnit Zadavateli vzdálený přístup do repository Dodavatele a bezplatně zajistit prostředky k procházení analýzy a designu.

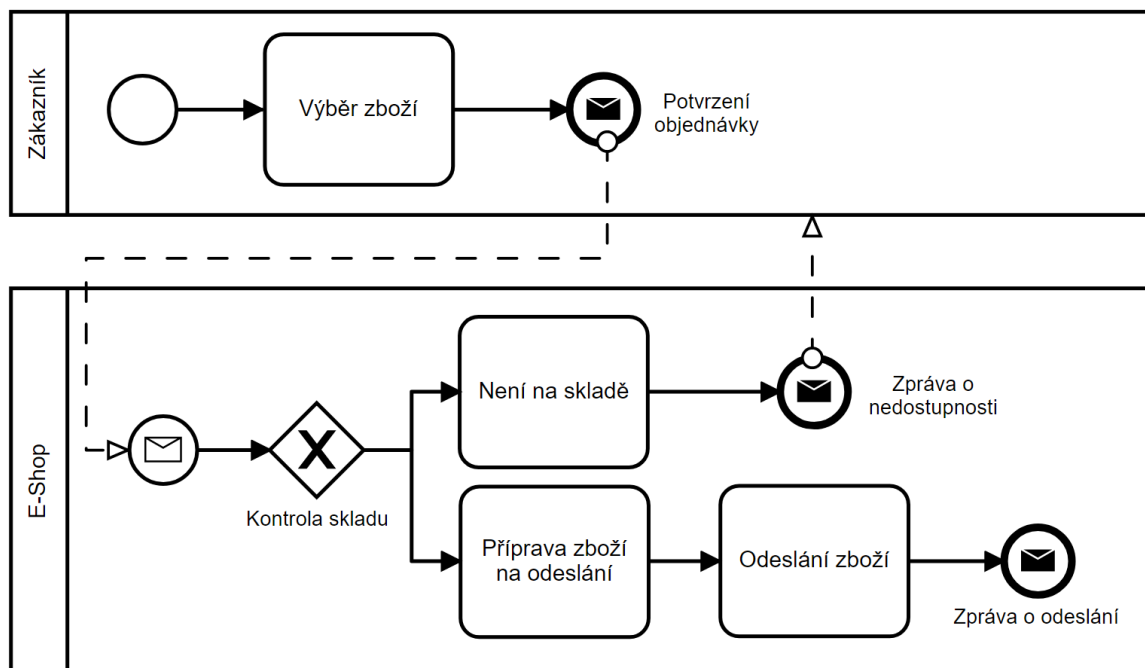
Analýza musí být v průběhu životního cyklu systému průběžně udržována tak, aby byla vždy v aktuálním stavu – tzn. při dílčích změnách funkčnosti systému musí být upravena i analýza. Vzhledem k tomu, že v současné době neexistuje „konzistentní“ analýza, provede Dodavatel její postupné vytvoření.

Dokument analýzy slouží k podrobnému analytickému popisu aplikace a musí jasně a přehledně popisovat záměry a principy pro vytvoření aplikace z hlediska funkčnosti, použitelnosti a možností dalšího rozvoje.

4.1 Procesní model (BPMN)

Pro základní systémové procesy, které aplikace řeší, budou vytvořeny procesní modely v notaci BPMN. Jednotlivé diagramy budou doplněny slovními popisy procesů.

Příklad – Objednávka v e-shopu:



- Zákazník si vybírá zboží v e-shopu
- Po výběru zboží potvrzuje zákazník objednávku
- Prodejce je informován o potvrzení objednávky
- Prodejce provede kontrolu dostupnosti zboží ve skladu
- Pokud zboží není na skladě, informuje o tom prodejce zákazníka
- Pokud je zboží skladem, zboží je připraveno k odeslání a odesláno. Zákazník je informován o odeslání zboží.

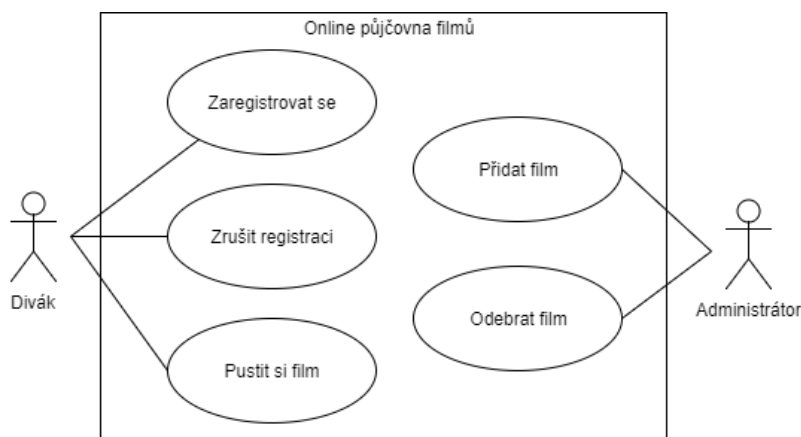
4.2 Případy užití (Use Cases, UML 2)

Všechny klíčové části interakce aplikace s uživateli a ostatními systémy budou popsány pomocí případů užití. Výčet interakcí bude znázorněn pomocí diagramu v notaci UML 2 a jednotlivé případy užití budou doplněny slovními popisy.

Každý popisovaný případ užití bude obsahovat diagram a slovní popis obsahující tyto prvky:

- Název
- ID případu užití
- Cíl
- Hlavní aktéři
- Vedlejší aktéři
- Vstupní podmínky
- Výstupní podmínky
- Základní scénář
- Alternativní scénáře.

Příklad:



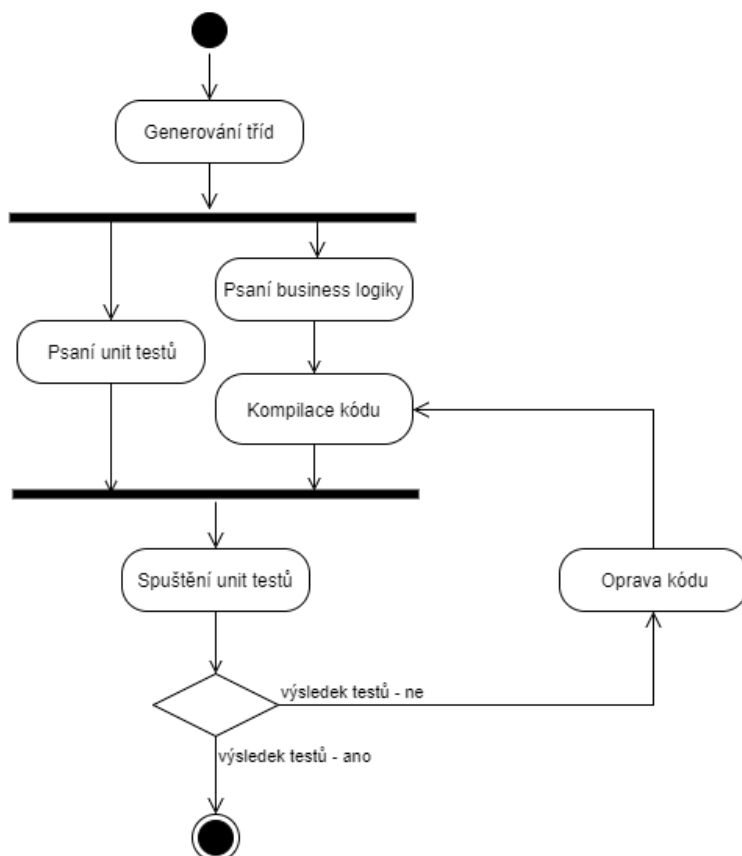
Název	Zaregistrovat se		
ID případu užití	UC1		
Cíl	Zaregistrování uživatele do systému		
Hlavní aktéři	Uživatel		
Vedlejší aktéři	-		
Vstupní podmínky	-		
Výstupní podmínky	Uživatel je zaregistrován do systému		
Základní scénář	Krok	Aktér	Akce
	1	Uživatel	Otevře v prohlížeči úvodní stránku a zvolí „Nový uživatel“.
	2	System	Zobrazí stránku registrace nového uživatele.
	3	Uživatel	Zapíše e-mailovou adresu a heslo.
	4	System	Zjistí, jestli už zadaná e-mailová adresa v systému existuje a zda zadané heslo splňuje požadovaná kritéria.
	5	System	Pokud adresa v systému není a heslo je v pořádku, systém uživatele zaregistruje a přesměruje ho na dashboard.
Alternativní scénář	Krok	Aktér	Akce
	5a	System	Pokud adresa v systému existuje nebo heslo nespĺňuje kritéria, uživateli je na stránce registrace zobrazena chybová hláška a výzva k opětovnému zadání (krok 2 základního scénáře).

4.3 Diagramy aktivit (UML 2)

Pro ilustraci u vybraných případů užití a algoritmů (zejména těch procesů, u kterých dochází k paralelnímu průběhu aktivit) budou zhotoveny diagramy aktivit podle standardu UML 2. Každý diagram aktivit bude mít stručný popis znázorňovaného případu.

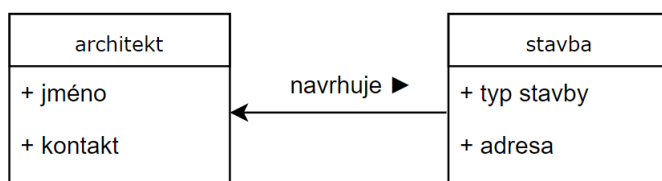
Příklad popisu: Diagram znázorňuje aktivity potřebné k dosažení spustitelného automaticky otestovaného zdrojového kódu.

Příklad diagramu s paralelním během aktivit:



4.4 Doménový model (Class Diagram, UML 2)

Ke znázornění vztahů mezi entitami vyskytujícími se v aplikaci bude sloužit doménový model. Ten bude popsán pomocí diagramu tříd v notaci UML 2 a bude obsahovat všechny hlavní entity (třídy) použité v programech aplikace. Jednotlivé třídy v diagramu budou mít z důvodu přehlednosti uvedeny pouze nejdůležitější atributy. Vztahy mezi entitami (asociace, agregace, atd.) budou mít v diagramu kromě spojnice i jednoduchý slovní popis, např.:

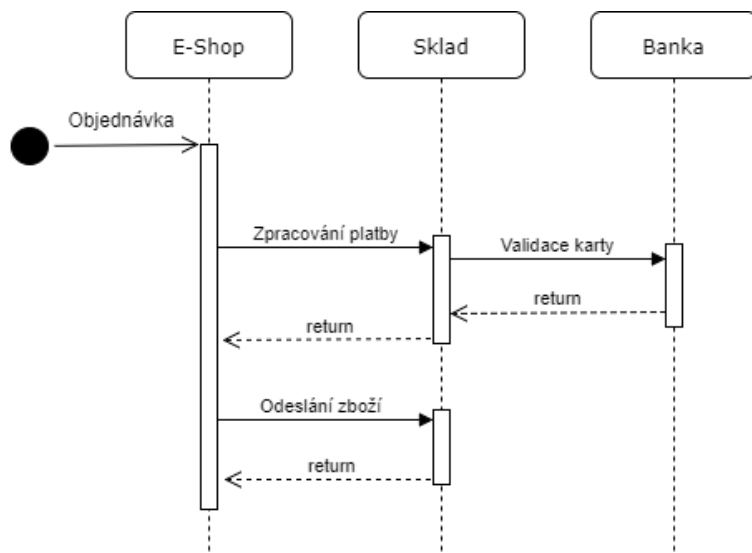


K diagramu bude formou tabulky připojen slovní výčet a popis významu jednotlivých entit.

4.5 Sekvenční diagramy (UML 2)

Pro případy užití, u kterých dochází v rámci jednoho případu užití k výměně zpráv mezi objekty a je důležitý pohled z hlediska časové posloupnosti této výměny, budou vytvořeny sekvenční diagramy podle standardu UML 2. Diagramy budou doplněny slovními popisy posloupnosti.

Příklad sekvenčního diagramu – objednání zboží v e-shopu:



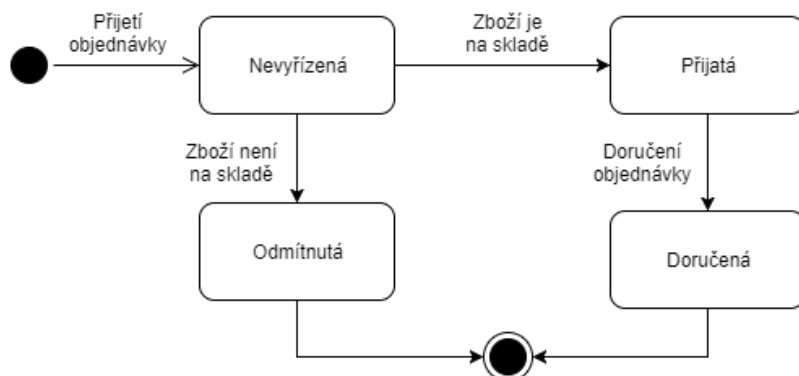
Příklad popisu:

1. Uživatel zadává do E-shopu objednávku
2. E-shop odesílá požadavek na Sklad, kde proběhne rezervace zboží
3. Sklad kontaktuje Banku, kde proběhne vlastní platba
4. Banka nazpět informuje Sklad, že zboží bylo zaplaceno
5. Sklad nazpět informuje E-shop, že rezervace a platba proběhly v pořádku
6. E-shop posílá do Skladu požadavek na odeslání zboží
7. Sklad potvrzuje E-shopu přijetí požadavku na odeslání.

4.6 Stavové diagramy (UML 2)

Pro všechny procesy, v jejichž průběhu dochází u entit k přechodu z jednoho stavu do jiného, například zpracování objednávky (objednávka mění svůj stav z nevyřízené na přijatou nebo odmítnutou atd.), budou vypracovány stavové diagramy podle standardu UML 2.

Příklad – Změny stavu objednávky:

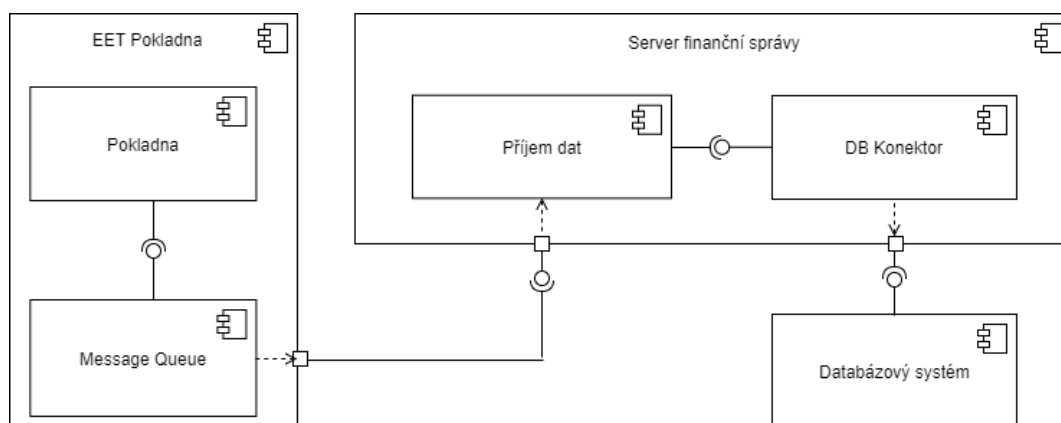


4.7 Diagramy komponent (UML 2)

Vztahy mezi samostatnými komponentami v rámci celé aplikace budou názorně vysvětleny pomocí diagramů komponent v notaci UML 2. Každá komponenta poskytuje a/nebo požaduje od jiných komponent služby, které jsou přístupné pouze přes rozhraní. Každou komponentu lze teoreticky nahradit novou komponentou, pokud poskytuje stejné rozhraní.

Každý diagram komponent bude doplněn slovním popisem vztahů mezi jednotlivými komponentami.

Příklad diagramu komponent – Evidence tržeb:



Příklad popisu:

Tento diagram zobrazuje vztahy komponent při odesílání dat o tržbách z EET pokladny na Server finanční správy a následné uložení dat do Databázového systému.

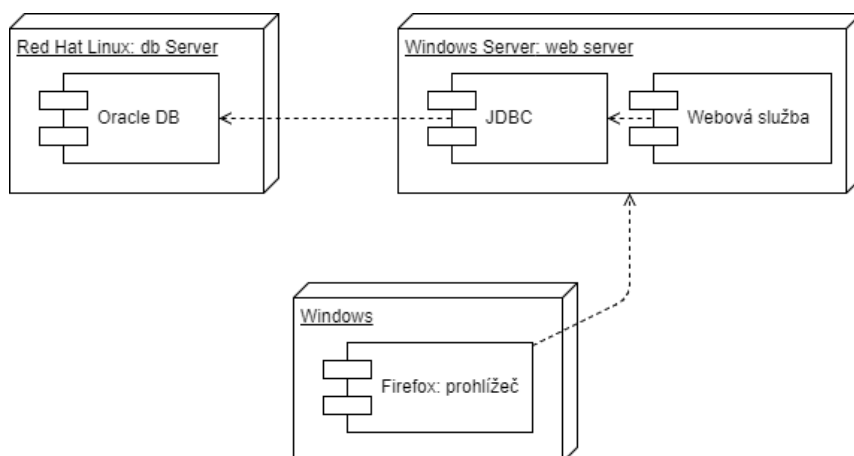
Pokladna se připojuje přes komponentu fronty zpráv (Message Queue) na systém Server finanční správy. Pro případy, že mezi frontou zpráv a serverem není internetové spojení, je fronta zpráv nastavena tak, aby uměla pozdržet odeslání informací až do chvíle, kdy je spojení přes internet znovu zprovozněno.

Systém Server finanční správy má dvě hlavní komponenty, webovou službu Příjem dat a DB Konektor. Webová služba Příjem dat poskytuje rozhraní, na které se připojuje komponenta Message Queue ze systému EET Pokladna. Přijatá data jsou předána na rozhraní komponenty DB Konektor, která je následně odesílá k uložení do komponenty Databázový systém.

4.8 Diagramy nasazení (UML 2)

Pro znázornění rozložení jednotlivých softwarových komponent na hardwarových zdrojích (uzlech) a jejich spolupráce budou pro všechny scénáře využití aplikace vypracovány diagramy nasazení (deployment diagrams) v notaci UML 2.

Příklad diagramu nasazení:



4.9 Požadavky na UI

V této sekci budou popsány jak základní požadavky na uživatelské rozhraní (web, desktop, mobilní aplikace atd.), tak i podrobnější informace (přizpůsobení velikosti uživatelských obrazovek velikosti displeje, fonty, barvy atd.).

Také je třeba detailně specifikovat způsob ovládání (klávesnice, myš, dotyk, hlas, atd.) a navigace (např. menu, ribbon atd.).

Kapitola bude doplněna grafickými ukázkami požadovaných podob uživatelských rozhraní.

4.10 Požadavky na výkonost aplikace

Všechny zadavatelem požadované výkonové parametry (např. maximální časy odezvy načítání obrazovek, databázových dotazů nebo webových služeb) musí být v analýze zdokumentovány pro potřeby průběžné kontroly proti stávajícím verzím aplikace.

4.11 Požadavky na HW a SW

Dalším bodem analýzy budou požadavky aplikace na hardware a software, v případě vícevrstvé aplikace pro každou vrstvu zvlášť. Je třeba uvést minimální konfigurace počítačů, operační systémy, síťová napojení a veškerý pomocný software nutný k provozování aplikace.

4.12 Doplnující informace

Tato kapitola bude obsahovat ostatní výše nepopsané důležité informace, které jsou potřebné k pochopení fungování aplikace.

5 Databáze

5.1 Datový model

Pokud aplikace používá databázi, bude tento dokument obsahovat její kompletní logický datový model. Podle typu databáze půjde o model relační, objektový, atd. Model bude pokud možno rozdělen na dílčí celky podle jednotlivých agend aplikace pro větší přehlednost. Příklad diagramu relačního datového modelu:

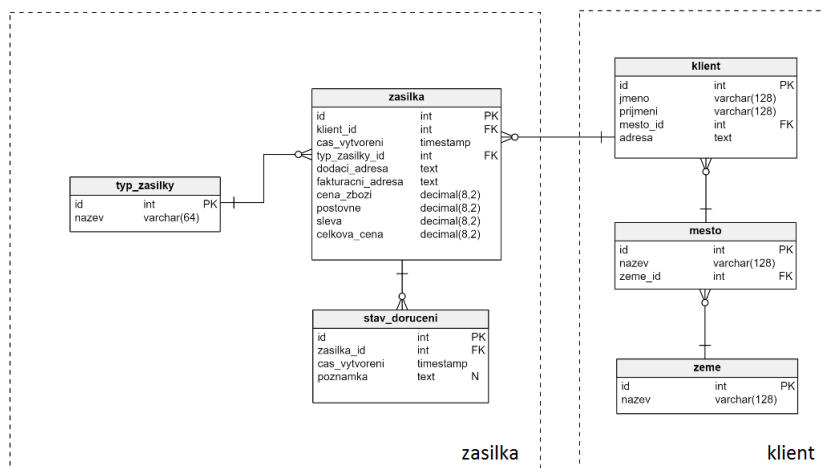


Diagram bude doprovázen kompletním seznamem entit a jejich položek s popisy.

Příklad popisu databázové entity:

Položka	Datový typ	Nullable	Defaultní hodnota	Popis
id	int	Ne	null	Jednoznačný identifikátor zásilky
nazev	varchar(64)	Ne	null	Název zásilky

5.2 Komentáře

Pokud to databáze umožňuje, budou komentáře k obsahům jednotlivých entit a jejich položek uloženy přímo ve schématu databáze. Tento způsob uložení komentářů umožní jejich čtení přímo při práci s databází a současně také automatické generování dokumentace pomocí k tomu určených nástrojů.

Komentáře ke sloupcům tabulky ponesou vysvětlující text k účelu sloupce, v komentáři celé tabulky bude vysvětlen účel tabulky v rámci IS DMVS a popis dat v ní uchovávaných a označení verze tabulky.

Pokud databáze obsahuje vlastní procedury, pak každá z nich musí mít minimálně komentář popisující její účel a vstupní a výstupní parametry. V případě delšího kódu procedury budou komentářem opatřeny i jednotlivé logické bloky kódu.

Příklad komentáře v hlavičce uložené procedury:

```

/* *****
Objekt:      addSetting
Typ:         procedura
Účel:        Procedura pro zápis nového záznamu v tabulce nastavení

PARAMETERS:
    pSettingCode:      kód nastavení
    pSettingValue:     hodnota nastavení
***** */
  
```

5.3 Uložené procedury, funkce, sekvence

Dokument bude obsahovat seznamy a stručné popisy všech uložených procedur, funkcí, sekvencí a dalších důležitých elementů definovaných v databázi.

5.4 Aktualizace dokumentace

Dokumentace musí být aktualizována při každém nasazení databázových změn na produkční prostředí. Bude popsáno, jakým způsobem bude přítomnost aktualizované dokumentace kontrolována (automatické kontroly, checklisty atd.).

6 Bezpečnost

Tento dokument bude popisovat všechny způsoby a zásady použité pro zabezpečení aplikace proti neoprávněnému užití.

6.1 Autentizace

V této kapitole budou uvedeny mechanismy použitého způsobu autentizace (jednoznačné určení uživatele přistupujícího do systému). Pokud jsou k autentizaci používány speciální aplikace (např. adresářové služby), služby operačního systému (např. Windows Authentication) nebo hardwarová zařízení (čipové karty, snímače otisků prstů), musí být vysvětlen způsob jejich využití.

Pro všechny autentizační prvky (hesla, certifikáty, PIN) budou popsána pravidla pro jejich vytváření, ukládání, doba jejich platnosti a způsoby obnovy. U ukládaných údajů (hesla, PIN) bude uveden způsob jejich zabezpečení (šifrování) v uložené podobě.

6.2 Autorizace

Popis principů autorizace (ověření oprávnění uživatele provádět v systému příslušnou akci) bude obsahovat výčet rolí s návazností na seznamy přístupných funkcí (agend).

6.3 Zabezpečení infrastruktury

V této kapitole bude kompletně popsáno zabezpečení infrastruktury, na které je aplikace provozována (servery, komunikační protokoly, firewally, certifikáty, klientské počítače atd.).

6.4 Audit přístupu k datům

Zde bude uvedeno, jakým způsobem je v průběhu používání aplikaci zaznamenáván (auditován) přístup k datům pro prevenci a případné kontroly možnosti úniku těchto dat.

7 Programátorská dokumentace

Programátorská dokumentace popíše aplikaci z pohledu užitečného pro vývojáře.

Pokud to bude pro pochopení výhodné, budou jednotlivé popisy obsahovat odkazy na [Dokument 2 - Analýza](#).

7.1 Použité technologie

Tato sekce bude obsahovat kompletní výčet technologií použitých pro vývoj (programovací jazyky, běhová (runtime) prostředí, frameworky) včetně jejich verzí. V případě vlastních frameworků nebo méně obvyklých technologií budou uvedeny odkazy na jejich oficiální dokumentace.

7.2 Nástroje používané pro vývoj

Zde bude poskytnut detailní výčet zahrnující editory, překladače (compilery), debuggery, testovací frameworky a IDE (Integrated Development Environment) prostředí používané v rámci vývojového týmu.

7.3 Lokální vývojové prostředí

Pro vývojáře bude vytvořen podrobný návod vysvětlující konfiguraci prostředí pro lokální vývoj. Musí být jednoznačně určeno, co a jak je potřeba mít nainstalováno na lokálním prostředí pro umožnění

místního spouštění aplikace, přidávání změn a jejich ladění. Podrobnější informace o správě verzí zdrojových kódů, databází a dokumentace bude v samostatném dokumentu, viz [Verzování](#).

7.4 Organizace kódu

V této sekci budou vysvětleny způsoby organizace zdrojových souborů do sdružujících logických a fyzických celků (adresářů, projektů, balíčků, jmenných prostorů (namespaces)).

7.5 Pravidla pro psaní kódu

Tato kapitola bude podrobně specifikovat pravidla pro psaní kódu v rámci celého projektu. Dále bude popsáno, jakým způsobem budou tato pravidla vynucována automatickými kontrolami kódu při buildech a verzování.

7.5.1 Jmenné konvence

Oddíl jmenných konvencí bude obsahovat kompletní popis pravidel používaných pro vytváření názvů projektů, balíčků (packages), programů, tříd, rozhraní, konstant, proměnných atd.

7.5.2 Komentáře

Zde bude popsáno, jakým způsobem budou okomentovány jednotlivé pasáže kódu a možnosti využití těchto komentářů k automatickému generování dokumentace (např. JavaDoc). Dále bude specifikováno, jakým způsobem dochází k automatickému generování dokumentace a kde je vygenerovaná dokumentace zpřístupněna. Dodržování zvolených konvencí bude automaticky kontrolováno při buildu aplikace.

Komentáře musí splňovat následující minimální kritéria:

- pro každou třídu musí být nad její definicí uveden popis jejího účelu,
- všechny metody kromě privátních musí mít v komentáři nad definicí minimálně:
 - popis účelu,
 - popis parametrů,
 - popis návratové hodnoty,
- formát komentářů musí umožňovat automatické generování dokumentace pomocí standardních nástrojů (JavaDoc, XML Doc pro .NET, atd.).

Příklad okomentované třídy a metody v jazyce Java:

```
/**
 * Třída pro práci s řetězcí
 */
public class StringHelper {

    /**
     * Vytváří objekt typu string z předaného pole znaků
     * @param chars Pole znaků
     * @return Objekt typu string
     */
    public String CreateString(char[] chars)
    {
        return new String(chars);
    }
}
```

Ukázka části vygenerované dokumentace (JavaDoc):

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method and Description
java.lang.String	CreateString(char[] chars) Vytváří objekt typu string z předaného pole znaků

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

StringHelper

public StringHelper()

Method Detail

CreateString

public java.lang.String CreateString(char[] chars)

Vytváří objekt typu string z předaného pole znaků

Parameters:

chars - Pole znaků

Returns:

Objekt typu string

PACKAGE

CLASS

TREE

DEPRECATED

INDEX

HELP

Příklad okomentované metody v jazyce C#:

```
/// <summary>
/// Třída pro práci s řetězci
/// </summary>
public class String
{
    /// <summary>
    /// Vytváří objekt typu string z předaného pole znaků
    /// </summary>
    /// <param name="chars">Pole znaků</param>
    /// <returns>Objekt typu string</returns>
    public string CreateString(char[] chars)
    {
        return new string(chars);
    }
}
```

Ukázka části vygenerované dokumentace (.NET XML Doc/Sandcastle):

▲Syntax

```
CF VB C++ F#  
public class StringHelper
```

The `StringHelper` type exposes the following members.

▲Constructors

Name	Description
 StringHelper	Initializes a new instance of the <code>StringHelper</code> class

[Top](#)

▲Methods

Name	Description
 CreateString	Vytváří objekt typu <code>string</code> z předaného pole znaků

7.5.3 Formátování

Tato sekce bude popisovat pravidla pro formátování zdrojového kódu (odsazování bloků, použití mezer a tabelátorů, maximální délky řádků). V případě použití nástrojů pro automatickou kontrolu formátování (tzv. „lint“ nástroje) budou doplněny jejich popisy včetně aktuálních konfigurací.

7.5.4 Internacionalizace a lokalizace

Přizpůsobení aplikace pro použití podle místních zvyklostí z hlediska psaní zdrojového kódu bude popsáno včetně způsobů vytváření, ukládání a využívání zdrojů (resources).

7.6 Pravidla pro tvorbu UI

Tato část programátorské dokumentace bude sloužit jako referenční soupis pravidel pro všechny vývojáře podílející se na vývoji nových uživatelských rozhraní a prvků. Bude obsahovat celkový koncept ovládání aplikace přes UI a odůvodnění jeho výběru. Dále bude uveden kompletní výčet prezentačních, grafických a jiných ovládacích prvků, jejich podoba, rozměry a barvy včetně zvyklostí jejich nasazování pro konkrétní scénáře.

Budou uvedeny příklady použití vybraných prvků z pohledu programátora včetně ukázek zdrojového kódu a/nebo screenshotů. Jednoznačně bude vysvětleno, jakým způsobem budou definice grafické podoby implementovány programátorem (např. kaskádové styly, XML definice atd.). Dále budou popsány možnosti tvorby vlastních komponent a elementů.

Pokud aplikace podporuje provoz na různých zařízeních, bude uveden jejich výčet a pravidla pro vytváření a testování změn v aplikaci z pohledu optimalizace pro různá zařízení (tzv. responzivity).

V případě existence grafického manuálu bude tento k programátorské dokumentaci přiložen jako příloha.

7.7 Verzování

V této kapitole bude uveden stručný popis verzování z hlediska vývojáře. Podrobně se správou verzí zdrojových kódů, databázových změn a dokumentací bude zabývat samostatný dokument, viz [Verzování](#). Na konkrétní části tohoto dokumentu, které se vztahují na vývojářské práce, zde budou uvedeny odkazy.

7.8 Testování

Bude jasně specifikováno, jakým způsobem se vývojáři podílejí na testování aplikace, ať už psaním a spouštěním jednotkových (unit) testů, používáním strojových integračních testů nebo ruční kontrolou funkcionality z pohledu uživatele.

8 Verzování

V tomto dokumentu bude podrobně rozepsáno, jakým způsobem dochází při tvorbě aplikace k zaznamenávání změn souborů v průběhu času, jak často a podle jakých pravidel se vydávají nové verze aplikace a jakým způsobem se lze vrátit k předchozím verzím za účelem obnovy funkčního stavu aplikace nebo identifikace chyb.

Popsaný způsob musí umožňovat:

- porovnání jednotlivých verzí souborů,
- zjištění kdo, v jakém čase a proč dané změny provedl,
- vrácení souboru nebo celého projektu do předchozího stavu.

8.1 Verzování zdrojových kódů

Tato kapitola bude obsahovat deskripci nástroje pro správu verzí zdrojového kódu, popis procesů aktualizace lokální kopie zdrojových kódů, zaverzování změn do lokálních i centrálních repozitářů, jejich zpětné odrolování, vytváření vývojových větví, jejich slučování (merge) s hlavní linií a označování verzí.

8.2 Verzování databázových změn

Oddíl verzování databázových změn bude vytvořen s ohledem na použití vývojáři i databázovými administrátory. Musí být jednoznačně vysvětleno, jakým způsobem jsou vytvářeny a spouštěny migrační skripty, archivovány změny a také kdo je za jednotlivé kroky zodpovědný. Dále musí být uveden způsob práce se všemi verzovacími a migračními nástroji, které jsou ke správě verzí databáze využívány.

8.3 Verzování dokumentace

Verzování dokumentace bude popsáno obdobným způsobem jako verzování zdrojových kódů (viz výše) s ohledem na specifika ukládaného typu obsahu.

9 Buildy a nasazování aplikace (Release and Deployment)

Tento dokument bude kompletně popisovat plánování a řízení sestavování (buildů) verzí aplikace a jejich nasazování na testovací a produkční prostředí.

9.1 Nasazování verzí na testovací a produkční prostředí

Bude podrobně popsáno, jakým způsobem se úspěšně vytvořený build dostává na testovací nebo produkční prostředí a jakým způsobem je možné vrátit se k předchozímu buildu v případě zjištění kritických chyb.

10 Integrace s dalšími systémy

V tomto dokumentu budou popsány integrace s dalšími informačními systémy, které aplikace využívá v průběhu svého provozu. U každé integrace bude uveden její účel, četnost využití, datové toky, podrobnosti o použité technologii napojení (webové služby, soubory, atd.), možnosti konfigurace a odkazy na dokumentace API externích systémů.

Pokud jsou využívány jakékoli integrační platformy (message bus, datové adaptéry, nástroje na orchestrace procesů, schedulery, atd.), budou vysvětleny principy jejich použití v rámci aplikace.

Popisy integrací budou pro větší přehlednost doplněny diagramy komponent v notaci UML 2.

11 Popis API

Pokud aplikace poskytuje rozhraní pro přístup dalších systémů, budou popsána veškerá poskytovaná rozhraní (API – Application Programming Interface). Dokument bude zpracován tak, aby bylo možné jej v případě potřeby předat třetím stranám jako návod pro implementaci napojení jejich aplikací.

11.1 Popisy vystavených webových služeb

Pro všechny webové služby, které aplikace vystavuje pro využití externími systémy, budou vytvořeny popisy se seznamy všech atributů a způsobem konzumace služeb. Tyto popisy budou doplněny o konkrétní příklady volání s ukázkami dotazů a odpovědí služeb.

11.2 Popisy dalších poskytnutých rozhraní

Pro všechna další poskytnutá rozhraní, která aplikace nabízí k využití, budou vytvořeny popisy vstupů a výstupů. Popisy budou doplněny o konkrétní příklady volání s ukázkami dat.

Pro všechna rozhraní (webové služby, souborový systém, atd.) bude podrobně popsán způsob jejich zabezpečení proti neoprávněnému užití (protokoly, certifikáty, omezení přístupů přes VPN, atd.).