Obsah

* Návrh architektury
* Blokové schéma řešení
* Topologie prostředí
* Dekompozice
* Výpis použitých technologií včetně licenčních podmínek
* Portálový aplikační server Liferay
* Aplikační server Tomcat
* Základní provozní prostředí JRE
* Databázový server PostgreSQL
* Schéma fyzických komponent řešení
* Popis procesu odeslání požadavku a jeho zpracování
* Technické parametry navrženého řešení
* Požadavky na technickou infrastrukturu – návrh HW architektury
* Latence komunikujících komponent při zpracování požadavku
* Předpokládané datové toky
* Základní zjednodušený návrh datového modelu
* UML diagram
* Návrh API
* Popis datových služeb
* Popis vstupních a výstupních dat
* Třída Zákazníci
* Třída Karty
* Třída Obálky
* Popis chybových stavů
* Návrh dalších úprav a doporučení

Návrh architektury

Předkládaný návrh architektury spolu s celkovým programovým řešením vychází z těchto podmínek a omezení, definovaných zadávací dokumentací:

* Udržení kontinuity provozu (minimálně) z pohledu koncových zákazníků – tím je specifikován samozřejmý požadavek na zachování kompletních zákaznických dat spolu s jejich přenosem ze starého systému na nový. Struktura nové databáze musí být sémanticky kompatibilní s původní databází včetně migračních postupů a aktualizačních algoritmů (dostačující je jednosměrná kompatibilita).
* Zachovat v maximální možné míře současné specializované HW vybavení (tedy HW odbavovacího systému, výrobce IVAR) – požadavek na ochranu investic zákazníka. Návrh architektury nebude do tohoto HW vybavení zasahovat, data touto vrstvou produkovaná budou do nového systému zpřístupněna přes softwarová rozhraní.
* Zachovat současný odbavovací systém (IVAR), pokladní systém (IVAR) a systém jejich propojení – souvisí s předchozím bodem a týká se aplikačního vybavení nad touto technologickou vrstvou. Důsledkem těchto požadavků je nezbytnost definovat vhodná rozhraní a provést jejich implementaci na odpovídajících stranách komunikace (ESB konektor vs. IVAR).
* Zachovat současný ekonomický a účetní systém (KARAT) – jde o poskytování dat pro měsíční závěrku, příp. jiných ekonomicky významných dat, v podstatě dávkovým způsobem kupř. ve formě exportu do Excelu. Do budoucna lze uvažovat o modernější formě integrace.
* Zachovat současný web – společnost zadavatele v současné době již provozuje svůj web, požadavkem je jeho rozšíření o části, sloužící jednak klientům (registrace, klientské účty a karty, osobní přehledy apod.), jednak administrátorům systému.

Blokové schéma řešení

Topologie prostředí

Provozní prostředí zadavatele, které má být pokryto předkládaným systém, je tvořeno

* Centrem, v němž je (nebo má být) provozována centrální databáze a koncentrováno zpracování dat včetně centrální administrace.
* Sportovišti – detašovaná pracoviště, provozující lokální databázi a provádějící její aktualizaci na základě výstupů z připojených technologií (pokladna, turniket, skříňka apod.) nebo administračních aplikací.

Centrální databáze a lokální databáze na sportovištích by měly být synchronizovány, tzn. aktualizace v centru by měla být pokud možno ihned propagována do databází na sportovištích a aktualizace v databázi každého sportoviště by se měla promítnout do centrální databáze (a odtud pak zase do databází sportovišť). Jde o komunikační model *hub-and-spoke*.

Dekompozice

Z výše uvedeného, topologie prostředí zadavatele a na základě možností navrhovaných technologií chceme architekturu řešeného systému sestavit z těchto bloků nebo modulů a rozhraní:

*Modul ESB*

Modul ESB má v celkovém řešení ústřední roli – jeho úkolem je spravovat centrální databázi s údaji klientů a provádět její aktualizaci na základě komunikace (změnových zpráv) s jednotlivými sportovišti.

Z názvu, který zadavatel zvolil pro tuto předpokládanou součást řešeného systému, lze usuzovat, že jeho představou je nasazení některého z produktů z kategorie podnikových sběrnic (*Enterprise Service Bus*). Z vlastností a funkcionalit, které standard ESB klasifikují, se však v navrhovaném řešení využije jen málo – chybí kupř. potřeba řízení procesů a toků zpráv (BPM), mediace, konverze protokolů atd. Součástí našeho návrhu je využití jednodušší výkonné komponenty, pokrývající potřebné funkcionality – konkrétně aplikačního serveru, poskytujícího

* Nasazení a volání služeb – v konkrétním případě webových služeb REST
* Efektivní databázovou vrstvu
* Prostředky pro monitoring a logování
* Přiměřenou úroveň zabezpečení, nástroje pro autentizaci a autorizaci
* Správu a administraci prostřednictvím webového rozhraní
* Přiměřenou úroveň otevřenosti a rozšiřitelnosti o další funkcionality, jejichž účelnost může resultovat z detailní analýzy řešeného systému nebo z vývoje potřeb a poznatků

*Modul ESB Connector*

Úkolem tohoto modulu, provozovaném v každé lokalitě-sportovišti, je správa lokální databáze (ve smyslu lokální „kopie“ centrální databáze) a její vzájemná synchronizace s centrem. Tato synchronizace musí být schopna překlenout možné výpadky spojení s centrem. Zároveň vystavuje služby pro volání ze strany pokladního a odbavovacího systému IVAR.

*Modul WEBAPP*

Webová aplikace navázaná na modul ESB a hostovaná v rámci stávajícího webu zadavatele, umožňující správu klientských účtů a celkovou administraci klientské evidence.

*Moduly pokladního a odbavovacího systému IVAR, účetní systém KARAT*

Z pohledu řešení předmětu této zadávací dokumentace jde o *legacy* systémy. S moduly předmětného řešení (ESB, ESB Connector) mohou komunikovat pouze přes dohodnutá nebo stanovená rozhraní.

*Rozhraní ESB – ESB Connector*

Jeho obsahem budou především zprávy, zajišťující synchronizaci obsah centrální databáze na straně ESB a obsahů lokálních databází příslušných jednotlivým ESB konektorům v lokalitách-sportovištích. Významným faktorem, který je třeba při návrhu a implementaci tohoto rozhraní zohlednit, je, že spojení mezi ESB a libovolným konektorem se může kdykoli rozpadnout, dobu výpadku do opětovného navázání nelze nijak kvantifikovat. Rozhraní bude implementováno jako REST webová služba.

*Rozhraní ESB Connector – IVAR*

Toto rozhraní představuje komunikační kanál mezi lokální databází a pokladním a odbavovacím systémem IVAR na daném sportovišti. Rozhraní bude implementováno jako REST webová služba. Vzhledem k tomu, že spolu s komponentami IVAR bude součástí téže LAN, není třeba řešit otázku výpadu spojení s nimi.

*Rozhraní ESB – WEBAPP*

Jde o rozhraní mezi datovou základnou ESB a webovým portálem společnosti zadavatele. Bude tvořeno sadou asynchronních REST webových služeb (kupř. pro ajaxové volání z webových stránek).

*Rozhraní ESB – KARAT*

Jde o dávkové rozhraní pro přenos účetních dat z centrální databáze do ekonomického systému KARAT.

Výpis použitých technologií včetně licenčních podmínek

V následující části je provedena konkretizace řešených součástí (ESB, ESB Connector a jejich rozhraní) na úrovni softwarových komponent.

Portálový aplikační server Liferay

Portálový aplikační server Liferay bude využit pro realizaci modulu ESB. Splňuje veškeré požadavky plynoucí z popisu účelu a nezbytné funkcionality ESB, navíc může poskytovat další vlastnosti využitelné při případném rozšiřování záběru ESB (významná může být kupř. jeho schopnost tvorby prezentační vrstvy na úrovni standardních portletů – webových aplikací pro portály).

Liferay je javovský portálový a aplikační server, dostupný jak pod bezplatnou licencí GNU Lesser General Public Licence (https://web.liferay.com/downloads/liferay-portal/license-ce) jako komunitní edice Liferay CE, tak jako komerční produkt jako podniková edice Liferay EE. Pro účely předmětného projektu využijeme komunitní edici.

Jako javovský produkt je portovatelný jak na platformu MS Windows, tak na platformu linux.

Z vlastností a technologií standardu J2EE, které jsou významné pro záměr využít tento server jako implementaci ESB, můžeme uvést

* Umožňuje nasazení a provoz standardních REST webových služeb (i SOAP), jak v synchronním, tak v asynchronním režimu
* Jako databázovou vrstvu využívá osvědčenou technologii Hibernate. Díky tomu je při komunikaci s databází efektivní (polling spojení, databázová cache) a platformově nezávislý – podporuje všechny typy běžných databázových serverů (MS SQL, ORACLE, PostgreSQL atd.)
* Podporuje standard JMS pro asynchronní výměnu zpráv (messaging)
* Obsahuje nástroje pro scheduling a notifikování - provádění časově závislých činností a činností na základě událostí
* Obsahuje prostředky pro monitoring (kupř. implementaci standardu JMX) a logování (systémové i aplikační)
* Je otevřený – v případě rozšíření jeho záběru a výskytu potřeby využít další z technologií standardu J2EE ji lze snadno doplnit

Z dalších rysů, významných pro předmětné řešení, můžeme uvést

* Autentizace a autorizace uživatelů – server Liferay obsahuje nástroje pro ověřování uživatelů a vyhodnocování jejich definovaných oprávnění. Zdrojem uživatelských identit pro ověřování může být jak interní databáze serveru Liferay, tak externí zdroje typu LDAP, NTLM, OpenID, sociální sítě nebo některá z implementací SSO (CAS, Open SSO, SiteMinder). Tato funkcionalita se aktuálně (v kontextu zadávací dokumentace) uplatní pouze pro ověření přístupu administrátora ESB, nicméně při úvahách o budoucích rozšířeních systému je možné tyto vlastnosti brát do úvahy.
* Administrace a konfigurovat – server Liferay obsahuje webové rozhraní, přes které je možné jej plně administrovat a konfigurovat.
* Paralelní zpracování – ošetřování klientských požadavků ve více vláknech
* Prezentační vrstva – Liferay obsahuje implementace standardů JSR168 a JSR286 pro tvorbu a provoz portletů. Tato oblast je aktuálně mimo záběr řešené zadávací dokumentace, nicméně při úvahách o budoucích rozšířeních systému je možné tuto skutečnost brát do úvahy.

Aplikační server Tomcat

Javovský aplikační server využijeme pro realizaci modulu ESC Connector (přesněji řečeno, bude tvořit provozní prostředí tohoto modulu). Podobně jako v případě serveru Liferay jde opět o produkt, šířený pod bezplatnou licencí Apache License version 2 (http://www.apache.org/licenses/). Jako javovský produkt je provozovatelný jak na platformě MS Windows, tak na linuxových zařízeních.

Oproti výše uvedenému portálovému a aplikačnímu serveru Liferay představuje Tomcat jednodušší variantu aplikačního serveru, která však pokryje potřeby implementace a provozu komponenty ESB Connector a přinese i menší nároky na příslušné HW prostředí.

Z využitelných vlastností serveru Tomcat můžeme v kontextu provozu ESB Connector jmenovat především efektivní komunikaci s databází (polling spojení), příjem a zasílání požadavků ve formě JSON (příp. SOAP) a paralelní ošetřování došlých požadavků ve více vláknech.

Základní provozní prostředí JRE

Ve výčtu technologií nelze vynechat základní provozní prostředí Java Runtime Environment (JRE), neboť na něm běží jak aplikační a portálový server Liferay (ESB), tak jednotlivé instance aplikačního serveru Tomcat (ESB Connector) na lokalitách/sportovištích.

JRE je šířeno bez poplatků pod licencí ORACLE BCL (http://www.oracle.com/technetwork/java/javase/terms/license/index.html) a je dostupné jak pro platformu MS Windows, tak pro prostředí linux.

Databázový server PostgreSQL

Databázový server PostgreSQL slouží jednak jako úložiště dat v centru – centrální databáze pod správou komponenty ESB, jednak na všech lokalitách/sportovištích jako úložiště lokálních dat pod správou lokálně příslušné komponenty ESB Connector.

Server PostgreSQL je vyspělý databázový stroj, šířený pod svobodnou licencí PostgreSQL License (<https://www.postgresql.org/about/licence/>), dostupný jak na platformě MS Windows, tak v prostředí linux. Z jeho vlastností využijeme v kontextu předmětné zadávací dokumentace především možnost transakčního zpracování, nízké náklady a efektivnost databázových operací, definice triggerů a SPL a existenci zvláštního datového typu pro formát JSON. Z provozního hlediska je výhodou přítomnost grafického administračního rozhraní nebo řádkových utilit pro archivaci, obnovu z kopie apod.

Schéma fyzických komponent řešení:



Popis procesu odeslání požadavku a jeho zpracování

Výše uvedený popis odpovídá uplatnění základního principu servisně orientované architektury (SOA), kdy komponenty vystavují svá API pro konzumaci ze strany jiných komponent a mohou figurovat jak v roli poskytovatelů služeb, tak jejich konzumentů.

Předpokladem, ze kterého při návrhu rozhraní a celé komunikace vycházíme, je uplatnění REST webových služeb - na rozdíl od SOAP WS zjednodušujících proces marshallingu na transportní formát (JSON) a zpět, a tudíž méně ovlivňujících latenci zpracování zprávy, a dále umožňujících přímou komunikaci s javascriptem ve stránkách aplikace WEBAPP. Jako přenosový protokol bude využit protokol http.

To, zda budou služby komponent volány synchronně (tj. volající komponenta bude čekat na odpověď), nebo asynchronně (volající komponenta poskytne callback funkci pro příjem zpracované odpovědi), záleží na řešení příslušné klientské komponenty. Lze očekávat, že klienti z prostředí WEBAPP (nejsou předmětem našeho řešení), prezentovaní javascriptem v příslušné HTML stránce, budou volání provádět přes Ajax, tedy asynchronně. Komponenty ESB a ESB Connector budou používat synchronní volání (budou tedy „čekat na odpověď“), což s ohledem na paralelní zpracovávání požadavků (ve více vláknech) a na charakter operací (změnová data pro synchronizaci, dotazy na stavy) předkládané řešení nijak nedegraduje.

Na serverové straně (tj. u komponent v roli poskytovatele služeb) také nepředpokládáme řešení frontování došlých požadavků. Infrastruktura uvedených aplikačních serverů (Liferay, Tomcat) tuto záležitost řeší strategií přidělování zvláštních vláken pro ošetření požadavků od určitého klienta, požadavky různých klientů jsou tedy řešeny paralelně a bez vzájemného ovlivňování.

Uplatnění komunikace prostřednictvím REST WS („JSON over http“) předpokládáme rovněž ve vztahu k existujícím komponentám legacy systému IVAR.

Technické parametry navrženého řešení

Požadavky na technickou infrastrukturu – návrh HW architektury

Dosavadní stav programového vybavení zadavatele pro předmětnou oblast nepožadoval zvláštní výpočetní výkon – proto jsou použita vesměs PC s MS Windows, RAM 2-4 Gby, CPU i5 (příp. nižší), HDD SATA 320 GBy a více.

Po zhodnocení kvantitativních nároků předpokládaných SW komponent (Liferay jako ESB, Tomcat jako ESB Connector, databáze PostgreSQL) navrhujeme pro jejich provoz využít aktuální HW zadavatele.

Latence komunikujících komponent při zpracování požadavku

Latenci zpracování jako časovou prodlevu mezi vznikem požadavku klienta a okamžikem jeho vyřízení na straně serveru ovlivňují následující faktory:

* Marshalling požadavku z podoby interní datové struktury klienta do podoby řetězce ve formátu JSON
* Otevření komunikačního kanálu k adresátovi (poskytovateli služby)
* Vlastní přenos k adresátovi
* Unmarshalling doručeného požadavku z podoby řetězce ve formátu JSON na interní datovou strukturu poskytovatele
* Zpracování na straně poskytovatele, zahrnující rovněž jeden nebo více požadavků do databáze

Uvedené faktory nelze v rámci této nabídky reálně kvantifikovat – rozhodně zanedbatelný čas představují operace marshallingu a unmarshallingu. Ostatní parametry jsou přímo ovlivňovány především výkonností sítě. Dá se očekávat, že pokud půjde o komunikaci v rámci LAN (komunikace ESB Connector vs. IVAR v rámci lokality/sportoviště, komunikace ESB/ESB Connector se „svou“ databází), půjde rovněž o zanedbatelné časy. Více se na celkové latenci může projevit komunikace přes internet (ESB vs. ESB Connector) a pak pochopitelně neefektivní návrh databázové struktury nebo řešení business logiky v dodaných komponentách (ESB a ESB Connector).

Předpokládané datové toky

Datové toky budou tvořit textová data přenášená protokolem http. Kritická se pohledu kvality spojení jeví pouze komunikace mezi centrem (ESB) a lokalitami (ESB Connector), komunikace v rámci sportoviště, tj. ESB Connector vs. IVAR, je vedena v rámci LAN (příp. dokonce v tomtéž paměťovém prostoru).

Z pohledu frekvenční a objemové kvantifikace půjde za celý systém v odhadu špičkově o 100 souběžných požadavků na prodej lístků (v dodatečných informacích zadavatele se uvádí desítky) s objemem do 1 kBy / požadavek. K tomu můžeme připočíst podobný počet doplňkových dotazů na stavy a jiné ad hoc dotazy. Jako bezpečně předimenzovaný se tedy zdá požadavek na přenos 5-násobku odhadu podle odpovědi na doplňkové dotazy, tedy špičkový přenos 500 kBy/sec.

Výpadky v komunikaci centra a lokalit jsou předpokládané a budou ošetřeny v rámci analytického návrhu synchronizace dat.

Základní zjednodušený návrh datového modelu

**Upozornění: Jedná se o úvodní analytickou abstrakci, která bude při detailnějším rozpracování upravena tak, aby odpovídala požadovanému cílovému řešení.**

UML diagram

Ve zjednodušeném návrhu uvedeném níže jsou uvedeny pouze nejdůležitější atributy a metody. Datový model vychází z předpokladu, že mezi kartami a obálkami má existovat relace M:N (usuzujeme tak z obrázku „Nová obálka“ v Příloze č. 6, kapitola 4.3.2).



Pod pojmem „Nevázaný zůstatek“ máme na mysli zůstatek zákazníkova účtu, který nebyl „rozdělen“ do jednotlivých obálek. „Vázaný zůstatek“ pak představuje částku, kterou zákazník rozdělil do obálek a nemůže s ní volně nakládat, pokud neprovede změny ve správě obálek.

Návrh API

Popis datových služeb

Jednotlivé komponenty systému budou komunikovat prostřednictvím zpráv ve formátu JSON. Každá zpráva bude – mj. pro účely synchronizace dat mezi jednotlivými databázemi – opatřena:

* jednoznačným identifikátorem UUID doplněným o kód instance, která zprávu odesílá (pro snazší identifikaci „autora“ zprávy);
* časovou značkou.



Předpokládáme, že tuto formu by v budoucnu mohly využívat i webové aplikace (zákazníci – správa účtu, karet, obálek; SAREZA – administrace zákazníků). Položka Výsledek je popsána v části Popis chybových stavů.

Popis vstupních a výstupních dat

Názvy zpráv budou odpovídat názvům metod jednotlivých instancí. Z technického hlediska budou všechny zprávy ve formátu JSON odesílány na endpoint, který je – podle typu zprávy – bude zpracovávat.

Další případy užití vznikají kombinací níže uvedených metod (např. historie zůstatku na účtu bude implementována voláním funkce *UkažZůstatek*() k určitým datům).

Třída Zákazníci

|  |  |  |  |
| --- | --- | --- | --- |
| **metoda** | **popis** | **vstupní data** | **výstupní data** |
| Registruj() | Registrace nového zákazníka | * údaje o zákazníkovi dle požadavků zadavatele
 | * id zákazníka
 |
| DobijÚčetČástkou() | Dobití zákazníkova účtu finanční částkou | * id zákazníka
* způsob dobití
* pokladní
* částka (Kč)
 | * nevázaný zůstatek
 |
| DobijÚčetKódem() | Dobití zákazníkova účtu kódem (předpokládané užití – různé marketingové akce) | * id zákazníka
* kód
 | * nevázaný zůstatek
 |
| UkažZůstatek() | Vrátí zůstatek zákazníkova účtu ke stanovenému datu | * id zákazníka
* datum
 | * nevázaný zůstatek
* vázaný zůstatek
 |
| VyplaťZůstatek() | Pro případ, že by v budoucnu bylo možné zůstatek na kartě zákazníkovi vrátit. | * id zákazníka
 |  |
| UkažHistoriiTransakcí() | Zobrazení historie transakcí (dobití, platby za služby, obálky) | * id zákazníka
 | * historie transakcí ve strukturované podobě
 |
| ZaslatÚčtenku() | Údaje pro účtenku | * pokladní
* typ placení
* kolekce položek
 |  |

Třída Karty

|  |  |  |  |
| --- | --- | --- | --- |
| **metoda** | **popis** | **vstupní data** | **výstupní data** |
| VytvořKartu() | Vytvoření karty | * id zákazníka
* údaje na kartu
 | * id karty
 |
| UpravKartu() | Úprava údajů na kartě | * id karty
* nové údaje na kartu
 |  |
| BlokujKartu() | Blokace karty | * id karty
 |  |
| OdblokujKartu() | Zrušení blokace | * id karty
 |  |
| ZaplaťSlužbuKartou() | Úhrada služby | * id karty
* id obálky (pokud je)
* pokladní
* kolekce položek
 | * nevázaný zůstatek
* vázaný zůstatek
 |

Třída Obálky

|  |  |  |  |
| --- | --- | --- | --- |
| **metoda** | **popis** | **vstupní data** | **výstupní data** |
| VytvořObálku() | Vytvoření obálky | * id zákazníka
* id karty
* platnost do
* částka
* id služby
 | * id obálky
 |
| UpravObálku() | Úprava obálky | * id obálky
* id karty
* platnost do
* částka
* id služby
 | * vázaný zůstatek
 |
| SmažObálku() | Odstranění obálky | * id obálky
 | * nevázaný zůstatek
 |

Popis chybových stavů

Pro účely posílání zpráv předpokládáme vytvoření číselníku, který by popisoval nejen stav zprávy, ale i výsledek odpovědi. Možné hodnoty:

|  |  |
| --- | --- |
| -1 | Na zprávu dosud nedošla odpověď (výchozí hodnota pro nové zprávy) |
| 0 | Odpověď – úspěšné zpracování |
| >0 | Odpověď – neúspěšné zpracování (kód chyby) |

Návrh dalších úprav a doporučení

1. Do budoucna implementovat pro některé položky číselníky (například pro pokladní, způsoby dobití) – kromě textové podoby (prozatím ponechané z důvodu kompatibility) by pak ve zprávách bylo zasíláno rovněž i ID položky.
2. U zákazníků a karet navrhujeme zavést atribut Kategorie – přinese možnost budoucího rozčlenění zákazníků či karet do různých skupin a následné využití při marketingových akcích.