

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

Diplomová práce

Docházka a výkazy práce pro systém IMIS na platformě Android

Prohlášení

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

V Plzni dne 28. června 2013

Maxipes Fík

Abstract

Text of abstract.

Obsah

1	Úvod	1
2	Současný systém	2
2.1	Oracle forms	2
2.2	Architektura	2
2.3	Komponenty formuláře	3
2.3.1	Moduly	3
2.3.2	Triggery	5
2.3.3	Seznam hodnot	5
2.4	Uživatelské rozhraní	6
2.5	Datový model	7
2.6	Důležité formuláře	7
2.6.1	Zápis příchodů a odchodů	7
2.6.2	Výkaz práce	8
3	Analýza	9
3.1	Architektura	9
3.1.1	Přímé připojení k databázi	9
3.1.2	Oracle Database Mobile Server	9
3.1.3	Webová služba	10
3.2	Výběr typu webové služby	11
3.2.1	Representational State Transfer	11
3.2.2	Simple Object Access Protocol	11
3.3	Datová vrstva	11
3.3.1	Práce s datumem a časem	11
3.3.2	Kritika datové vrstvy	12
3.4	Business logika	12
3.4.1	Triggery	12
3.4.2	Databázové balíčky a uložené procedury	13
3.4.3	Forms knihovny	13
3.5	Synchronizace dat	13

3.5.1	Obousměrná synchronizace	13
3.5.2	Obousměrná synchronizace s úpravou databáze	13
3.5.3	Srovnání	14
3.6	Uživatelské rozhraní	14
3.6.1	LOV	14
4	Zabezpečení	15
4.1	Autentizace a autorizace	15
4.2	VPN pro vzdálený přístup	16
4.3	HTTP Basic autentizace	17
4.4	Použité řešení	17
4.5	Alternativní řešení	18
4.5.1	LDAP	18
4.6	Shrnutí	20
5	Webová služba	21
5.1	REST	21
5.2	Filtry	24
5.3	web.xml	24
6	Android aplikace	25
6.1	Funkcionalita	25
6.1.1	Nastavení a konfigurovatelnost	26
6.1.2	Uživatelská přívětivost	26
6.2	Architektura	26
6.3	Business logika	26
6.4	Android komponenty	27
6.5	Ukládání dat	27
6.6	Struktura projektu	27
6.7	Manifest + oprávnění	27
6.8	9png grafika	27
6.9	Návrhy na vylepšení	27
6.10	SQLite	28
6.11	REST	29
6.12	Synchronizace	29
6.13	Zabezpečení	29
6.14	Oprávnění	29
6.15	Zpětná kompatibilita	29
6.16	Budoucí rozšiřitelnost	29
6.17	Vytváření grafů	29
6.18	Chybové reporty	30

6.19	Distribuce	30
7	Testování	31
7.1	O čem psát...	31
7.2	Zásady pro vypracování	31

1 Úvod

2 Současný systém

IMIS = Integrovaný manažerský informační systém

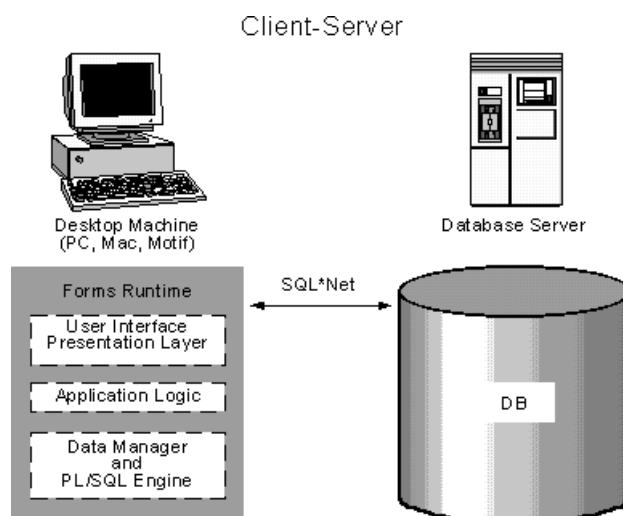
2.1 Oracle forms

Oracle forms je softwarový produkt vyvinutý společností Oracle. Slouží k vytváření formulářů, které interagují s Oracle databází. Jako programovací jazyk využívá PL/SQL. Produkt byl původně požíval terminálové rozhraní pro komunikaci se serverem. Později byl přepracován do architektury klient-server.

Prostředí běhu zajišťuje defaultní správu transakcí. Díky tomu je Oracle Forms silný nástroj pro efektivní vývoj aplikací, jejichž primárním cílem je přístup k datům uložených v databázi.

PL/SQL PL/SQL (Procedural Language/Structured Query Language) je procedurální nadstavba jazyka SQL od firmy Oracle založená na programovacím jazyku Ada.

2.2 Architektura



TODO asi vyrobit vlastní

2.3 Komponenty formuláře

Z hlediska architektury se Oracle Forms aplikace skládá z těchto celků:

2.3.1 Moduly

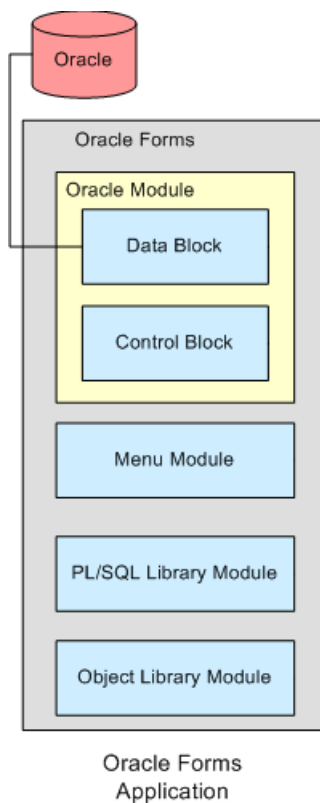
Modul formuláře Modul formuláře je hlavní komponenta aplikace. Poskytuje kód nezbytný pro interakci s úložištěm a uživatelským rozhraním. Data poskytovaná databází jsou reflektovaná v prvcích uživatelského rozhraní jako jsou textová pole, zaškrtačací políčka, přepínače, tlačítka atd. Formulář je logicky organizován do bloků. Existují dva typy bloků:

- **Datový blok**
Datový blok zobrazuje zdrojová data a poskytuje abstrakci pro způsob jakým jsou tato data získávána. Blok může být asociován s databázovou tabulkou, databázovým pohledem, uloženou procedurou, dotazem do databáze nebo transakčním triggerem. Asociace datového bloku a databázových dat standartně umožňuje přístup k těmto datům a jejich modifikaci. Datové bloky mohou být navzájem svázány vztahem "rodič - potomek". Takový vztah představuje relaci 1:N databázových tabulek. Oracle Forms zajišťuje to, že při spojení mezi master a detail bloky se zobrazí pouze ty detail bloky, které jsou vázány na master blok přes cizí klíč.
- **Řídící blok**
Představuje blok, který nemá vztah k databázové tabulce. Řídící blok může obsahovat jakékoli prvky uživatelského rozhraní. Prvky mohou sloužit k uložení dočasných proměných nebo k zobrazení dat, které nemají přímou vazbu s databází.

Modul menu Modul obsahuje hierarchii menu. Každé menu obsahuje zvolitelné položky. Každý formulář obsahuje defaultní menu obsahující příkazy pro základní DML operace s databází CRUD.

Modul PL/SQL knihovny Modul obsahuje znovu využitelný kód, který může být využit jinými formuláři, menu či knihovnami. Programové jednotky knihovny mohou být funkce, procedury a balíčky. Programové jednotky jsou spouštěny na straně klienta. Mohou obsahovat business logiku. Knihovny jsou nezávislé na formuláři, jsou zaváděny dynamicky a mohou být zároveň využívány více formuláři.

Modul knihovny objektů Modul obsahuje znovu využitelné objekty. Řeší uskladnění, správu a distribuci těchto objektů, které mohou být využity jinými formuláři, menu či knihovnami. Využívání tohoto modulu přináší přínosy v podobě úspory paměti při běhu aplikace.



TODO obrazek upravit - prepsat Oracle Module na Form Module

2.3.2 Triggery

Aplikace v Oracle pracuje s následujícími typy triggerů:

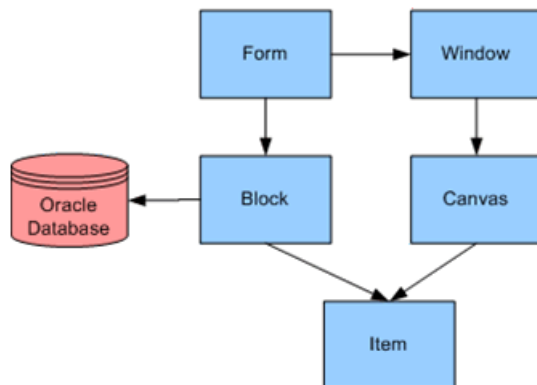
- Block-processing triggers - jsou spouštěny při události na položce patřící tomuto bloku.
- Interface event triggers - jsou spouštěny při události v uživatelském rozhraní formuláře.
- Master-detail triggers - jsou spouštěny při události související se vztahem "rodič - potomek" na daných blocích. Např. při změně položky rodiče příslušný trigger zobrazí správné položky v bloku potomka.
- Message-handling triggers - zpracovávají zobrazení chybových či informačních zpráv.
- Navigational triggers - jsou spouštěny při navigaci po položkách formuláře.
- Query-time triggers - jsou spouštěny na úrovni bloku před a po dotazu do databáze.
- Validation triggers - jsou spouštěny při validaci záznamu v položce.
- Transactional Triggers - vyvolají se při různých událostech související s interakcí s datovým úložištěm.

Pokud se jedná o datový blok, který je svázan s tabulkou v databázi, prostředí běhu automaticky zajišťuje DML pro tyto bloky. Pokud vývojář požaduje nestandardní akci při těchto úkonech, provede překrytí těchto triggerů s vlastní definovanou akcí.

2.3.3 Seznam hodnot

A (LOV) is a pop-up window that provides the user with a selection of values. The values can be static or populated by querying the database. LOVs are populated using columns returned by record groups. Check the Record Group property of the LOV for the record group which is used to provide values.

2.4 Uživatelské rozhraní



Plátno je objekt, na který je nakresleno celé GUI formuláře, tedy všechny viditelné objekty. Může mít prakticky jakoukoli velikost. Okno ohraničuje plochu plátna, která bude zobrazena. View řídí, jak bude plátno v určité době zobrazeno v okně.

TODO přepsat

2.5 Datový model

2.6 Důležité formuláře

2.6.1 Zápis příchodů a odchodů

[illegible]

+popsat z pohledu uzivatele

2.6.2 Výkaz práce

VX42 - Výkaz práce (01.03.2012) CCA

VÝKAZ PRÁCE

Uživatel 06/2012

Zaměst. **1520** **MARTIN KADLEC BC** **Kód** **KDA** **Stav** : **V** **P** **S** **Z**
 Vedoucí **1429** **JANSA JIŘÍ** **Střed** **PRG** (počet zázn.) **0** **15** **0** **0**
 Celkem za období z docházky (vyhodn.) **056:40** z docházky **060:54** z výkazu **056:40**

Výkaz práce - hodiny

Datum	S	Zakázka	Prac.	Hodin	Požad.	Hláš.	Organizace	Popis činnosti
PÁ 01/06/2012	P	R-CCA-FIRMA-	1	1	KDAREZC	02:00		sm
SO 02/06/2012	P	K-VV-N-2012	4	1	KDAREZC	01:00		firemni smernice
NE 03/06/2012								
PO 04/06/2012								
ÚT 05/06/2012								
ST 06/06/2012								
ČT 07/06/2012								
PA 08/06/2012	S	Zakázka	Auto	Km	Organizace	Popis činnosti		
SO 09/06/2012	V							
NE 10/06/2012								
PO 11/06/2012								
ÚT 12/06/2012								

Výkaz práce - Km

S	Zakázka	Auto	Km	Organizace	Popis činnosti
V					

Celkem z doch. **03:00** **A** **03:00** **00:00** **14:00(P)-15:00(P),16:59(P)-17:00(P),**

Alt1=PŘÍSTUPY Alt2=ZMĚNA OBDOBÍ Alt3=ODMĚNY OSV Alt4=SUMA Alt5=POTVRZENÍ
 Alt6=IMP:KNIHY J. Alt7=IMP:VÝKAZU ^P=TISK

+popsat z pohledu uzivatele

3 Analýza

3.1 Architektura

Při návrhu architektury jsem se rozhodoval mezi třemi variantami: přímé spojení Android aplikace ke vzdálené databázi pomocí JDBC, synchronizaci dat se vzdálenou databází pomocí Oracle Database Mobile Server a nakonec využití webové služby, která by sloužila jako rozhraní mezi klientskou aplikací a databázovým serverem.

3.1.1 Přímé připojení k databázi

Přestože přímé připojení k Oracle databázi pomocí JDBC je možné, tuto variantu jsem zamítl. Připojení pomocí JDBC je primárně určeno pro stabilní síťové připojení, které má malou odezvu a nízkou ztrátu paketů. Využití JDBC by přineslo problémy v podobě špatné odezvy aplikace, kvůli znovu navazování spojení a vytváření nových databázových relací, které musely být v důsledku ztráty konektivity ukončeny.

Vzhledem k tomu, že původní Forms aplikace funguje jako tlustý klient, provádí veškerou bussines logiku. Tato logika je zapotřebí ke správné funkci systému. Bylo by tedy nutné přenést tuto logiku na stranu klienta a potřeba komunikace se vzdálenou databází by byla větší než k pouhému přenesení dat.

3.1.2 Oracle Database Mobile Server

Oracle Database Mobile Server 11g je server zajišťující synchronizaci dat mezi Oracle databází a mobilními zařízeními. Klíčovou vlastností tohoto produktu je synchronizační jádro, které je schopné zajistit synchronizaci velké počtu mobilních zařízení se vzdálenou databázovým systémem. Přestože bylo toto synchronizační jádro navrženo pro stabilní připojení, je schopné zajistit spolehlivou funkci i při nestabilním připojení. V případě, že je spojení přerušeno synchronizace je pozastavena a po navázání spojení pokračuje v místě přerušení. Dále umožňuje šifrování dat, jak pro přenos tak i pro jejich persistenci.

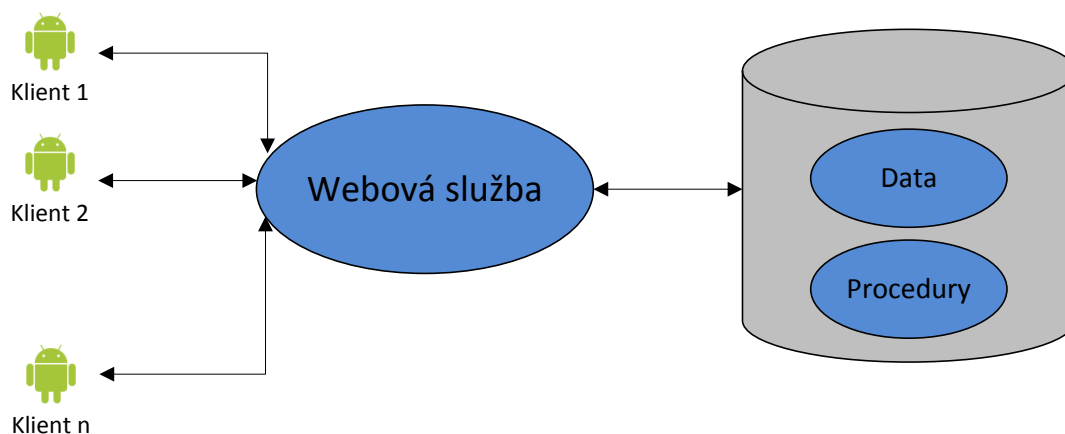
Tato varianta byla zamítnuta protože řeší pouze synchronizaci dat a neumožňuje zajistit provedení business logiky. Dalším důvodem je skutečnost, že její použití by vyžadovalo zakoupení licence pro tento server.

Server je možné spustit na serverech Oracle WebLogic Server a Oracle Glassfish. Mobilní klient, který běží na straně mobilního zařízení zajišťuje správu zařízení nutnou k synchronizaci. Tento klient je dostupný pro platformy Java, Android, Blackberry, Windows a Linux. (<http://www.oracle.com/technetwork/products/data-mobile-server/overview/index.html>)

3.1.3 Webová služba

Jako použitou architekturu jsem zvolil použití webové služby, která bude fungovat jako rozhraní mezi klientskou aplikací a databázovým serverem. Android klient v této architektuře funguje jako tenký klient spravující jen část funkčnosti z původního tlustého klienta. Business logika je umístěna na straně webové služby. Díky tomu že, webová služba bude umístěna v blízkosti firemní databáze, dojde k minimalizaci odezev při zajištění business logiky systému. Mezi klientem a webovou službou se přenášejí pouze data, která jsou opravdu nutná.

Z pohledu rozšiřitelnosti systému o další mobilní platformy se toto řešení jeví rovněž výhodně. Business logika by nebyla implementována ani na klientských aplikacích jiných platform. Při změně logiky bude potřeba úpravy v kódu pouze na straně webové služby. Cílová architektura na vyobrazena na obrázku 3.1



Obrázek 3.1: Zvolená architektura

3.2 Výběr typu webové služby

3.2.1 Representational State Transfer

3.2.2 Simple Object Access Protocol

3.3 Datová vrstva

3.3.1 Práce s datumem a časem

Při návrhu datového modelu jsem řešil problém pomocí jakého datového typu vyjadřovat údaj o čase či datu. V Oracle databázi je použit datový typ Date. SQLite databáze nabízí tři způsoby jako ukládat informaci o čase:

- **TEXT** podle ISO8601 normy ve formátu "YYYY-MM-DD HH:MM:SS.SSS".
- **REAL** podle Juliánského kalendáře, počet dní od poledne 24. Listopadu roku 4714 před křtstem (Greenwichského času).
- **INTEGER** jako Unix Time, počet sekund 1970-01-01 00:00:00 UTC.

Pro uložení v SQLite databázi jsem zvolil typ INTEGER. V aplikaci (Android klient, webová služba) jsem se rozhodl reprezentovat časový údaj pomocí primitivního typu long. Měl jsem k tomu řadu dobrých důvodů:

- odpadá starost s formátem datumu při serializaci a deserializaci JSON řetězce
- snadné porovnávání hodnot pomocí relačních operátorů
- sníží se počet konverzí v aplikaci (např. pro výpočet pozice pro vykreslení komponenty v UI)

Také jsem se ujistil, že rozsah typu long je pro potřeby aplikace dostatečný. Srovnání použitých datových typů je znázorněno v tabulce 3.1.

Datový typ	Minimální hodnota	Maximální hodnota	Přesnost
Oracle Date	January 1, 4712 BCE	December 31, 4712 CE	sekundy
SQLite INTEGER			sekundy
Java long	2.12.292269055 BC	17.8.292278994 AD	milisekundy

Tabulka 3.1: Datové typy reprezentující časový údaj

3.3.2 Kritika datové vrstvy

co se mi nelíbilo a co bych navrhl jinak a jak, návrh prichody/odchody - jeden radek, chybi primarni klic - ROWID jako unikatni identifikator, problemy ktere to prinasi, format casu - problemy s prevodem

3.4 Business logika

existuje nekajá možnost převodu formsů do javy - oracle adf - co to je, co to resi, proc to neresi muj problem

3.4.1 Triggery

jen ty, jejichž funkčnost bude muset být implementována.

- On-Delete, On-Insert, On-Update, Pre-Delete, Pre-Insert, Pre-Update
- When-Validate-Item

3.4.2 Databázové balíčky a uložené procedury

3.4.3 Forms knihovny

3.5 Synchronizace dat

V současném systému uživatel zadává data prostřednictvím příslušného formuláře. Změny jsou aplikovány bezprostředně po uložení během databázové transakce. Mobilní klient přináší nový způsob použití - data lze zadat i v režimu offline, kdy mobilní klient není v dosahu webové služby. Tyto data jsou uložena persistentně na straně klienta a jsou synchronizována až ve chvíli kdy je možná komunikace s webovou službou.

Synchronizace se týká pouze dat pro docházku uživatele. Ostatní data jsou prostřednictvím mobilní klienta pouze zobrazována. Je třeba počítat s tím, že záznamy přidané na straně klienta v režimu offline nemusí být přijaty při synchronizaci z důvodu porušení business pravidel a uživatel by měl být o této skutečnosti vhodně informován.

3.5.1 Obousměrná synchronizace

Při obousměrné synchronizaci se odesílají data ze strany klienta na server tak i opačným směrem ze serveru na klienta. Klienta lze navrhnout tak, aby si uchovával informaci o změnách na svojí straně. Při analýze databázového schématu pro docházku v současném systému jsem zjistil, že databáze neuchovává informaci o změnách na svojí straně. Beze změny této skutečnosti není možné sledovat změny na straně databáze. Výsledkem je poněkud neefektivní způsob synchronizace, kdy klient odesílá na server pouze změny, zatímco ze serveru stahuje všechna data pro daného uživatele a období.

3.5.2 Obousměrná synchronizace s úpravou databáze

Jiná varianta řešení problému synchronizace dat, která se snaží eliminovat nedostatky předchozí varianty, by vyžadovala změny v databázovém schématu současného systému. U každého záznamu by byla přidána informace o poslední změně záznamu s vhodnou časovou přestní - na sekundy. Pokud by

došlo k požadavku na smazání záznamu, nebyl by záznam skutečně smazán, ale pouze nastaven příznak smazaného záznamu. Při použití tohoto řešení by bylo možné synchronizovat oběma směry pouze změny ze strany klienta i serveru.

3.5.3 Srovnání

V obou případech řešení je iniciátorem synchronizace klient.

3.6 Uživatelské rozhraní

3.6.1 LOV

jaka alternativa v androidu

4 Zabezpečení

V následující kapitole se zabývám zabezpečením aplikace. Popisuji několik možných variant. Na závěr vysvětluji výběr zvoleného řešení.

4.1 Autentizace a autorizace

Při analýze současného systému jsem zjistil, že informace o docházce a výkazech zaměstnanců jsou dostupné všem ostatním uživatelům (údaje týkající se nadřízených pracovníků jsou dostupné i podřízením). Dalším zájmovostí je, že heslo používané k zadání docházky je pro uživatele nepovinné (má ho jen ten uživatel, který si ho nastavil) a je prakticky možné zadat docházku jinému uživateli. Není předmětem mé práce hodnotit tuto skutečnost, která je vlastností současného systému.

Autentizace

Autentizace je proces ověření proklamované identity subjektu. Vzhledem k tomu, že heslo je nepovinné a mají ho nastavené jen někteří uživatelé, nelze ověřit autenticitu pro každého uživatele.

Autorizace

Autorizace je proces získávání souhlasu s provedením nějaké operace. Současný systém neřeší autorizaci týkající se dat pro docházku a výkazy práce. Vzhledem k tomu, že nelze ověřit autenticitu pro každého uživatele, nelze autorizaci seriózně řešit.

Riziko poškození systému

Webová služba umožňuje čtení a úpravu docházkových dat a dále čtení dat o výkazech práce a zaměstnancích. Dále používá některé databázové objekty

jako jsou procedury a funkce, které pracují s těmito daty. Je vhodné aby aplikace měla přístup pouze k těm databázovým objektům, které jsou relevantní pro navrženou funkčnost aplikace. To je vhodné pro maximální zabezpečení okolního systému a minimalizaci případných rizik při zneužití či chybě v aplikaci. Toto je zodpovědností databázového administrátora organizace a v této práci se touto problematikou dále nezabývám.

4.2 VPN pro vzdálený přístup

Virtuální privátní síť je prostředek pro propojení počítačů v prostředí nedůvěryhodné sítě. Díky VPN spojení mohou počítače komunikovat tak, jako by byly součástí důvěryhodné sítě.

Vlastnosti připojení VPN

- Zapouzdření
Při použití technologie VPN jsou data zapouzdřena pomocí hlavičky obsahující směrovací informace, které umožňují průchod dat přes tranzitní síť.
- Ověřování
Klient a VPN server se vzájemně ověřují na úrovni počítače. Ověřování se děje pomocí PPTP protokolu.
- Šifrování dat
Pro utajení dat během jejich přenosu sdílenou nebo veřejnou tranzitní sítí jsou data na straně odesílatele zašifrována a na straně příjemce dešifrována. Šifrování a dešifrování je založeno na tom, že odesílatel i příjemce používají společný šifrovací klíč.

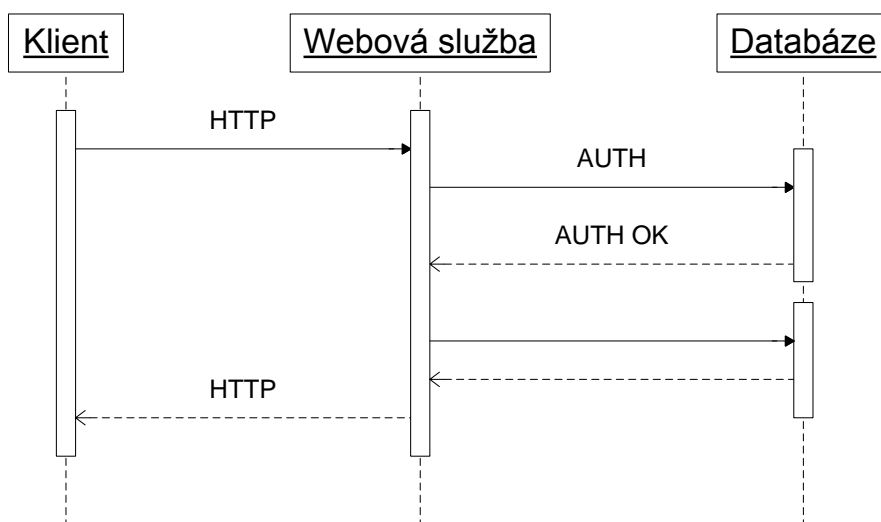
Webová služba je tedy umístěna na serveru uvnitř firemní sítě. Pokud klient chce komunikovat s webovou službou musí tak činit prostřednictvím sítě VPN.

4.3 HTTP Basic autentizace

Klient posílá autentizační hlavičku jako součást HTTP požadavku na server. Jméno a heslo je zasláno jako jeden textový řetězec oddělený dvojtečkou. Výsledný řetězec je poté zakódován metodou Base64. Uživatelské jméno a heslo se tedy posílá v zakódované podobě. Nejedná se ale o kryptografické zabezpečení přihlašovacích údajů. Použití této metody předpokládá použití zabezpečeného komunikačního kanálu mezi klientem a serverem.

4.4 Použité řešení

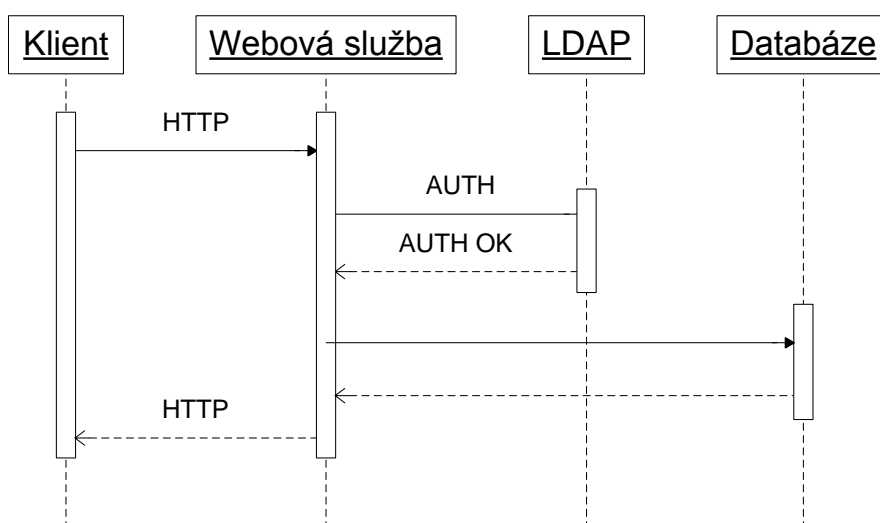
Klient i server jsou součástí jedné VPN sítě, která zajišťuje důvěryhodný komunikační kanál. Klient posílá na server HTTP požadavek jehož součástí je autentizační hlavička nesoucí uživatelské jméno a heslo. Webová služba ověří jméno a heslo pomocí databázové procedury. Jméno a heslo uživatele je uloženo v databázi. Implementované řešení je znázorněno na obr. 4.1 který zobrazuje sekvenční UML diagram v případě úspěšné autentizace.



Obrázek 4.1: Diagram aktivit při úspěšné autentizaci

4.5 Alternativní řešení

Alternativním řešením by bylo ověřování uživatelů pomocí LDAP adresáře. Organizace již LDAP používá v některých dalších firemních systémech. Toto řešení se liší v tom, že dotaz na ověření identity uživatele probíhá k LDAP adresáři a nikoli k databázi. Implementované řešení je znázorněno na obr. 4.2, který zobrazuje sekveční UML diagram v případě úspěšné autentizace.



Obrázek 4.2: Diagram aktivit při úspěšné autentizaci

4.5.1 LDAP

Directory Access Protocol (LDAP) je internetový protokol definující přístup k distribuované adresářové službě. Podle tohoto protokolu jsou jednotlivé položky na serveru ukládány formou záznamů a uspořádány do stromové struktury. Protokol LDAP je byl navržen v souladu se sadou standartů X.500 vyvinutých pro adresářové služby v počítačových sítích. Protokol LDAP je jejich odlehčenou verzí.

Aplikace funguje na bázi klient-server. Klient se při komunikaci se serverem autentizuje. Prostřednictvím klienta lze přidávat, modifikovat a mazat

záznamy na serveru.

Schéma

Úkolem informačního modelu LDAP je definovat datové typy a informace, které lze v adresářovém serveru ukládat. Data jsou uchovávána ve stromové struktuře pomocí záznamů. Záznam představuje souhrn atributů (dvojice jméno - hodnota). Atributy nesou informaci o stavu daného záznamu. Záznamy, uložené v adresáři, musí odpovídat přípustnému schématu. Schéma představuje soubor povolených objektových tříd a k nim náležících atributů. Ukázka schématu definující strukturu záznamu zaměstnance:

```
objectclass ( 1.1.2.2.2 NAME 'zamestnanec'
              DESC 'zamestnanec firmy'
              SUP osoba
              MUST ( jmeno $ identifikacniCislo )
              MAY zkratkaZamestnance )
```

Objekt popisující zaměstnance dědí od objektu osoba, vyžaduje povinný atribut 'jmeno' a 'identifikacniCislo' a nepovinný atribut 'zkratkaZamestnance'.

Funkční model

Funkční model umožňuje pomocí základních operací manipulovat a přistupovat k záznamům v adresáři a měnit či zjišťovat tak jejich stav.

- Autentizační operace: Slouží k přihlášení a odhlášení uživatele pro komunikaci s adresářovým serverem. Jsou jimi míněny především operace bind a unbind. Na úspěšném provedení operace bind závisí výsledky aktualizacích a dotazovacích operací nad adresářem.
- Aktualizační a dotazovací operace: Každý adresářový server podporuje základní operace s daty, jako je vyhledávání, přidávání, mazání, porovnávání a modifikace záznamů. Tyto operace bývají často spjaté s nastavením bezpečnostního modelu.

LDAP URL

Umístění zdroje je v LDAP specifikováno pomocí URL, které má následující tvar:

`ldap://host:port/DN?attributes?scope?filter?extensions`

- host - doména nebo IP adresa
- port - síťový port (defaultně 389)
- DN - význačné jméno použité jako základ pro vyhledávání
- attributes - seznam atributů
- scope - specifikuje vyhledávací rozsah
- filter - filtrovací kritérium
- extensions - rozšíření

4.6 Shrnutí

Použil jsem první řešení protože je to implementačně jednodušší. Stávají řešení pomocí Oracle Forms aplikace používá rovněž ověření uživatele dotazem k databázi tzn. ověření se děje na aplikační vrstvě.

Použití alternativní varianty by přineslo lepší zabezpečení ze strany mobilního klienta. Každý uživatel by musel použít svoje heslo, které nyní používá v ostatních systémech používající ověření pomocí LDAP. Vzhledem k tomu, že současný systém bude nadále fungovat, tak silnější zabezpečení mobilního klienta nemá praktický přínos.

5 Webová služba

5.1 REST

Při návrhu REST služby jsem nejprve identifikoval všechny zdroje, které webová služba zpřístupňuje. Jedná se o údaje o docházce zaměstnanců, výkazech práce a zaměstnancích samotných. Posledním zdrojem je možnost testovat komunikaci s webovou službou.

Zdroj pro docházku zaměstnanců tzn. jejich příchody a odchody (označované jako události) a poskytované služby zobrazuje tabulka 5.1. Služba umožňuje operace CRUD na datovém zdroji zaměstnanců a dále zjištění součtu doby v zaměstnání.

GET	events/{icp}?from={from}&to={to}
	Získá všechny události zaměstnance za dané období Parametry: <ul style="list-style-type: none"> • icp - identifikátor zaměstnance • from - datum začátku období • to - datum konce období
DELETE	events/{rowid}
	Smaže danou událost Parametry: <ul style="list-style-type: none"> • rowid - identifikátor události
POST	events
	Vytvoří událost, používá se bez parametrů protože identifikátor pro událost vytváří server
PUT	events/{rowid}
	Aktualizuje danou událost Parametry: <ul style="list-style-type: none"> • rowid - identifikátor události
GET	events/sum/{icp}?from={from}&to={to}
	Získá součet přítomnosti zaměstnance za dané období Parametry: <ul style="list-style-type: none"> • icp - identifikátor zaměstnance • from - datum začátku období • to - datum konce období

Tabulka 5.1: Služby pro události docházky

Zdroj pro údaje o zaměstnancích

GET	employees/{icp}
	Získá údaje o zaměstnanci identifikováno pomocí parametru Parametry: <ul style="list-style-type: none"> • icp - identifikátor zaměstnance
GET	employees/all/{icp}
	Získá seznam všech zaměstnanců, kteří jsou aktuálně v zaměstnaneckém poměru, obsahuje informaci zda jsou tito zaměstnanci podřízeni, vzhledem k zaměstnanci identifikováno pomocí parametru Parametry: <ul style="list-style-type: none"> • icp - identifikátor zaměstnance
GET	employees/lastevents
	Získá poslední událost v docházce všech zaměstnanců, kteří jsou aktuálně v zaměstnaneckém poměru
GET	employees/lastevents/{icp}
	Získá poslední událost v docházce zaměstnance identifikováno pomocí parametru

Tabulka 5.2: Služby pro zaměstnance

Zdroj výkazy práce

GET	records/{kodpra}?from={from}&to={to}
	Získá všechny výkazy práce zaměstnance za dané období Parametry: <ul style="list-style-type: none">• kodpra - identifikátor zaměstnance (zkratka)• from - datum začátku období• to - datum konce období
GET	records/sum/{icp}?from={from}&to={to}
	Získá součet vykázaného času zaměstnance za dané období Parametry: <ul style="list-style-type: none">• icp - identifikátor zaměstnance• from - datum začátku období• to - datum konce období

Tabulka 5.3: Služby pro výkazy práce

5.2 Filtry

5.3 web.xml

6 Android aplikace

6.1 Funkcionalita

Na základě analýzy současného systému a potřeb zaměstnanců byla vybrána k implementaci následující funkčnost:

Docházka

- Přehledné zobrazení událostí docházky daného zaměstnance
- Uživatel má možnost přidávat, ediovat a mazat svoje události
- Aplikace zajišťuje automatickou synchronizaci těchto údajů s firemní databází
- Zobrazení poměru typů docházkových událostí za dané období

Aktuální přítomnost na pracovišti

- Zobrazení seznamu zaměstnanců aktuálně přítomných na pracovišti
- Uživatel má možnost spravovat seznam svých "oblíbených" zaměstnanců a tento seznam zobrazovat přednostně

Výkazy práce

- Zobrazení poměru typů zakázek za dané období
- Zobrazení vývoje vývoje daného typu zakázky v daném období
- Možnost zobrazení těchto údajů i za jiné zaměstnance

6.1.1 Nastavení a konfigurovatelnost

Aplikace si musí pamatovat údaje nutné pro snadnou obsluhu tzn. uživatelské jméno a heslo, adresu umístění webové služby a tyto údaje jsou konfigurovatelné.

Dále aplikace umožní uživateli konfigurovat vzhled některých komponent, jako je barva typu události v docházce a typu záznamu ve výkazech.

6.1.2 Uživatelská přívětivost

Uživatelské rozhraní aplikace klade důraz na přehlednost, ergonomii a časově efektivní obsluhu.

6.2 Architektura

Android aplikace funguje jako tenký klient, který se připojuje k webové službě. Webová služba používá REST architekturu a přistupuje k samotné databázi.

- Webová služba - Java EE 6, aplikační server GlassFish
- Databáze - Oracle 10g, obsahuje navíc databázové procedury, které se používají v současných formulářích
- Android - obsahuje persistentní úložiště, obsahuje záznamy o docházce, úložiště se bude automaticky synchronizovat ve stavu online s databázovým serverem prostřednictvím webové služby

TODO prepsat srozumitelneji TODO schema komunikace -HHTTP, JDBC

6.3 Business logika

prijde to do webove sluzby - duvody

6.4 Android komponenty

1. komponenty pro sync a auth, provazani s android ucetm
2. CursorLoader
3. Async task
4. nestandardni UI
5. modifikace adapterview
6. custom UI - viewgroup

+ nejaka ukazka konkretniho pouziti

6.5 Ukládání dat

6.6 Struktura projektu

(jen ty použijte)

6.7 Manifest + oprávnění

6.8 9png grafika

6.9 Návrhy na vylepšení

Sdílené preference

ukládá primitivní datové typy ve tvaru klíč-hodnota. Slouží k uložení nastavení specifických pro aplikaci. Toto nastavení může být uloženo jako soukromé, kdy mohou k datům přistupovat pouze aplikace sdílející stejné *Linux user ID*.

V aplikaci používám toto úložiště pro nastavení síťového připojení (doména

a port webové služby) a barevného nastavení pro typy docházkových událostí.

Načtení dat se typicky odehrává v `onCreate()` metodě aktivity:

```
%\begin{lstlisting}
SharedPreferences settings = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
int color = settings.getInt("color", defaultColor);
%\end{lstlisting}
```

Uložení dat se typicky odehrává v `onStop()` metodě aktivity:

```
SharedPreferences settings = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putInt("color", userColor);
editor.commit();
```

Interní úložiště

Externí úložiště

SQLite databáze

Cloudové úložiště

6.10 SQLite

je třeba resit delku dat napríklad stringu?, dynamic typing

V knihovnách pro Forms aplikace se nachází další kód, který bude nutné přepsat do webové služby.

6.11 REST

RestaTemplates - springframework

1. REST operace - davkove vs jednotlivé
2. REST, tabulka URI,

6.12 Synchronizace

1. sync algoritmus - 2 algoritmy (jeden ideální, druhý reálný), srovnání
2. sync architektura - komponenty

6.13 Zabezpečení

-autentikace

6.14 Oprávnění

persmission v manifestu, vypsát a vysvětlit

6.15 Zpětná kompatibilita

6.16 Budoucí rozšiřitelnost

6.17 Vytváření grafů

knihovny, cloudové řešení, vlastní komponenty

6.18 Chybové reporty

6.19 Distribuce

7 Testování

7.1 O čem psát...

1. popsat IMIS
2. připraveno webové služby na další mobilní platformy
3. činnost aplikace online/offline
4. flow diagramy pro různé činnosti
5. přístupová práva
6. úspora persistentní paměti na straně androida
7. chybové reporty a opravy na aplikaci v ostrem prostředí, obrázků + ukázka
8. jak zjistit změnu záznamu, v datech se ukládá pouze datum poslední změny, nikoli přesný čas
9. perioda automatického mazání dat
10. datový model schema

7.2 Zásady pro vypracování

1. Prozkoumejte systém IMIS pro evidenci docházky a pracovních výkazů. Vyberte činnosti, které by bylo vhodné implementovat i pro mobilní zařízení.
2. Navrhněte mobilní aplikaci pro platformu Android, které bude obsahovat vybrané funkce z předchozího bodu zadání. Zvažte aspekty zabezpečení komunikace aplikace se systémem.
3. Implementujte navržené řešení, berte přitom v úvahu možnou rozšiřitelnost o další funkce.
4. Ověřte funkcionální vytvořené aplikace.

