Slovenská technická univerzita
Fakulta informatiky a informačných technológií

Tímový projekt

Projektová dokumentácia

# Platforma pre realizovanie transakcií prostredníctvom mobilných zariadení

**Bc. Miroslav Čorba**
**Bc. Branislav Hašto**
**Bc. Lukáš Lipka**
**Bc. Matej Lipták**
**Bc. Roman Pipík**

Študijný odbor: Softvérové inžinierstvo
Vedúci tímu: Ing. Michal Čerňanský, Phd.
Ročník: 1.
Semester: Zimný
Školský rok: 2010/2011

# 1. Časť

# Softvérový systém

# Obsah

# Úvod

## *Účel dokumentu*

Predkladaný dokument obsahuje projektovú dokumentáciu k softvérovému systému, určenému pre realizovanie transakcií prostredníctvom mobilných zariadení. Dokument je výsledkom študentského tímového projektu v predmete Tímový projekt, realizovaný na Fakulte informatiky a informačných technológií a je určený pre zadávateľa projektu, ktorým je Ing. Michal Čerňanský, Phd.

## *Metodika vývoja*

Vývoj sa riadi agilnou metodikou SCRUM, z čoho vyplýva viacero dôsledkov:

- vývoj prebieha v dvojtýždňových intervaloch – šprintoch
- šprinty sú zložené z príbehov (User Story)
- príbehy sú rozbité na úlohy (task)
- zložitosť príbehov je ohodnotená bodmi
- koniec každého šprintu je sprevádzaný odovzdaním riešení jednotlivých príbehov, naplánovaných pre daný šprint
- riešenie príbehu musí byť implementované, zdokumentované a otestované
- stretnutia tímu sú pravidelne raz za týždeň

## *Forma dokumentácie*

Forma dokumentácie sa pridŕža pravidiel, určených v spomenutom predmete a obsah pozostáva z opisu jednotlivých šprintov a k nim prislúchajúcich príbehov (angl. User story). Opis každého príbehu obsahuje analýzu, návrh, opis implementácie a spôsob testovania jednotlivých príbehov. Taktiež je každý príbeh navyše rozbitý na úlohy.

## *Výber implementačného jazyka, prostredia a technológií*

Implementácia je rozdelená do 2 častí:

- serverová časť
- klientska časť

**Serverová časť**

Cloudové služby – *Google App Engine*
Programovací jazyk – *Java*
Používateľské rozhrania na serveri – *Google Web Toolkit*
Komunikácia s klientskou časťou – *architektúra REST, prenosový protokol HTTP, výmena dát vo formáte JSON, implementované pomocou frameworku Restlet*
Vývojové prostredie – *Eclipse*

**Klientska časť**

Programovací jazyk – *Objective C, Cocoa Touch*
Vývojové prostredie - *Xcode*

# Šprint č. 1

Úlohou prvého šprintu bolo najmä oboznámiť sa s technológiami, ktoré budú použité pre implementáciu a vytvoriť jednoduchú aplikáciu, ktorá umožní pomocou mobilného zariadenia odoslať informáciu (reťazec) na server, ktorý túto informáciu uloží pod identifikačným číslom. Následne po zadaní tohto identifikačného čísla bude možné pomocou ďalšieho mobilného zariadenia získať túto informáciu. Bude to realizované zadaním identifikačného čísla, ktoré bolo vygenerované serverom.

V rámci tohto šprintu boli identifikované nasledujúce príbehy:

1. Vloženie textovej informácie do transakčného systému
2. Vybratie textovej informácie z transakčného systému

## *Vloženie textovej informácie do transakčného systému*

Ako používateľ chcem uložiť informáciu do systému, aby ju iný používateľ mohol zo systému vybrať.

### Analýza problému

Aby sa informácie mohli vkladať do transakčného systému, je potrebné riešiť nasledujúce problémy:

- Prenos informácie medzi klientom a serverom – je potrebné zvoliť prenosový protokol a navrhnúť formát pre výmenu správ medzi klientom a serverom. Navrhnutý protokol by mal mať čo najširšie využitie a mal by byť podporovaný na veľkom množstve zariadení.
- Poskytnutie údajov o vloženej informácii od servera pre klienta. To je potrebné, aby klient mohol túto informáciu ďalej poskytnúť iným klientom, ktorí si ju následne môžu vyzdvihnúť.
- Ukladanie prijatých informácií od klienta na strane servera – je potrebné navrhnúť, ako sa budú prijaté informácie ukladať.
- Odosielanie informácií na strane klienta – je potrebné vytvoriť používateľské rozhranie, ktoré umožní zadať informáciu a následne ju odoslať na server.

### Návrh riešenia

*Prenos informácie (HTTP protokol)*
Ako prenosový protokol sme zvolili HTTP. Dôvodom je jeho veľké rozšírenie. Protokol funguje na základe požiadaviek (Request) a odpovedí (Response). Každá požiadavka i odpoveď okrem samotných dát obsahujú aj hlavičky s metadátami.

Klient vždy iniciuje požiadavku na server. Požiadavky môžu prebiehať rôznymi metódami (napr. GET, POST, DELETE). Metóda indikuje, ako by sa mal server zachovať pri spracovávaní požiadavky (napr. POST sa zvyčajne používa pri požiadavkách, ktoré menia vnútorný stav aplikácie, naopak GET zväčša slúži na získanie dát z aplikácie, teda iba na zmenu zobrazenia).

Po prijatí požiadavky ju server spracuje a odošle odpoveď. Každá odpoveď obsahuje aj stavový kód, ktorý indikuje stav spracovania požiadavky. Existuje niekoľko druhov kódov (potvrdzovacie, presmerovacie, chybové atď.).

Ako prenosový formát (čiže formát tela správ) sme v tejto fáze zvolili iba čistý text (plain text). Dôvodom je, že zatiaľ ide iba o prenos neštrukturovaných informácií (textové reťazce).

*Poskytnutie údajov o vloženej informácii od servera pre klienta*
Na poskytnutie týchto údajov môžeme použiť priamo HTTP odpoveď. Po spracovaní požiadavky na vloženie informácie do systému v odpovedi vrátime údaje o tom, ako sa dá uložená informácia vyzdvihnúť.

*REST*
Aby mohol klient so serverom komunikovať, je potrebné presne definovať rozhranie servera. Rozhrania, ktoré umožňujú klientom komunikovať so serverom prostredníctvom siete, sa nazývajú webové služby. Vo väčšine prípadov táto komunikácia prebieha prostredníctvom protokolu HTTP. Pomocou webových služieb pracujeme s tzv. zdrojmi (Resources). Zdrojmi sú akékoľvek objekty, ktoré v systéme reprezentujeme. Každý zdroj je jedinečne identifikovaný pomocou tzv. URI (Uniform Resource Identifier).

Existujú dva prístupy k návrhu webových služieb:

- Služba definuje operácie, ktoré klient môže vykonávať (príkladom je napr. XML-RPC, príp. SOAP).
- Operácie sú definované implicitne, služba definuje, ako pristupovať k dátam (na akých adresách, akou HTTP metódou, atď.). Takýto štýl architektúry označujeme aj ako REST.

Rozhodli sme sa použiť štýl architektúry REST. Dôvodom je, že iné prístupy (napr. SOAP) sú pre naše účely zbytočne zložité.

REST rozlišuje 4 základné metódy na prístup k dátam (CRUD – Create, Read, Update, Delete), ktoré zodpovedajú HTTP metódam (POST, GET, PUT, DELETE). V tomto príbehu použijeme iba metódu Create. Aby sme klientovi mohli potvrdiť úspešné uloženie informácie na serveri, v odpovedi použijeme HTTP kód určený pre tento prípad, 201 Created.

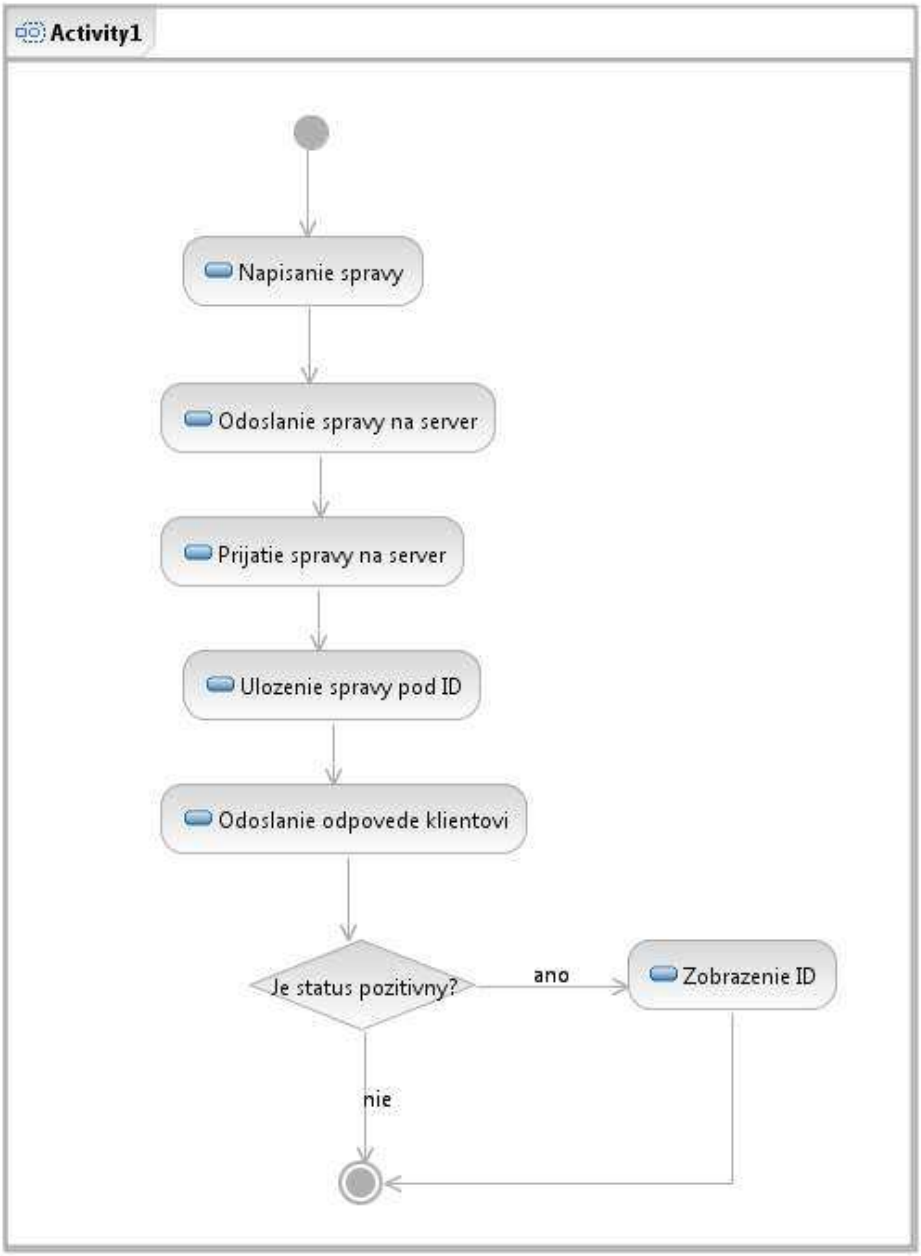
Podrobnejší opis komunikácie:
**Požiadavka**
**HTTP Metóda**: POST
**URI**: /strings/ (Pozn.: Toto URI je relatívne vzhľadom na server, kde bude REST rozhranie umiestnené.)
**Telo:** Reťazec, ktorý sa má uložiť

**Odpoveď**
**HTTP kód pri úspechu:** 201 Created
**Hlavičky:** Location: URI uloženého reťazca
**Telo:** Informácia o identifikačnom čísle, pod ktorým bol reťazec uložený

*Ukladanie prijatých informácií od klienta na strane servera*

Keďže cieľom prvého šprintu je oboznámenie sa s technológiou a vyskúšanie komunikácie medzi klientom a serverom, ukladanie informácií bude riešené iba pomocou dátovej štruktúry v pamäti servera. Problémom je, že údaje sa stratia po reštartovaní servera alebo aplikácie, teda nie sú perzistentné. Keďže tento jav nenastáva často, vzhľadom na naše ciele v prvom šprinte je to dostatočné riešenie.

*Odosielanie informácií na strane klienta*
Keď bude používateľ chcieť odoslať správu na server, napíše najskôr reťazec. Tento reťazec sa pomocou HTTP protokolu pošle na určenú URL adresu, na ktorej je server pripravený spracovať HTTP Request. Po prijatí HTTP requestu vytiahne server správu z jeho obsahu a uloží ju pod nejakým náhodne vygenerovaným číslom. Toto číslo sa potom odošle v odpovedi naspäť klientovi. Keď klient príjme odpoveď, tak skontroluje, či prišiel pozitívny potvrdzovací kód (či sa úspešne odoslala informácia). Ak bola transakcia v poriadku, tak zobrazí číslo, pod ktorým je na serveri uložená informácia. Graficky je tento postup zobrazený na obrázku č. 1.




**Obrázok 1Aktivity diagram pre proces odoslania informácie**

**Opis implementácie**

*REST*
Na implementáciu webovej služby sme použili framework Restlet. Za zdroje sa považujú reťazce (String), sú definované rozhraním StringResource. O implementáciu sa stará trieda InMemoryStringServerResource. Informácie sa ukladajú do objektu HashMap v pamäti, ktorá sa vytvorí pri štarte aplikácie. Operácia vkladania informácií je realizovaná metódou @Post store (String string). Výhodou frameworku Restlet je, že môžeme pomocou Java anotácii určiť, ktorá HTTP metóda je povolená pri volaní operácie. V tomto prípade sme určili, že je povolená iba metóda POST.

Aby sme vedeli definovať, ku ktorým zdrojom prislúchajú jednotlivé URI adresy, pri štarte aplikácie musíme nakonfigurovať tzv. smerovač (Router). Ten funguje na princípe regulárnych výrazov a umožní všetky požiadavky, ktoré zodpovedajú výrazu smerovať na určený zdroj. Pre splnenie tohto používateľského príbehu stačilo do smerovača pridať iba jednu cestu (route).

*Klientská časť*
Klientská časť bude bežať na mobilnom zariadení iPhone, preto je táto časť implementovaná v programovacom jazyku Objective C vo framework-u Cocoa Touch.

Informácia je z klientskej časti posielaná pomocou HTTP Request metódy POST. Na toto je použitá trieda z Objective-C `NSMutableURLRequest`, v ktorej sa metóda nastaví na POST, adresa kam sa posielajú tieto správy sa nastaví na "http://branovaprva.appspot.com/rest/strings/" a ako telo sa zadefinuje reťaz, ktorý je získaný z text boxu, ktorý je na obrazovke.

Po odoslaní požiadavky sa zo servera vráti odpoveď, z ktorej sa získajú údaje. Ak je status kód v rozmedzí od 200 do 300, znamená to, že správa bola doručená v poriadku. Preto sa pomocou triedy `UIAlertView` vygeneruje správa obsahujúca číslo ID, ktoré identifikuje miesto, kde je na serveri uložená správa, ktorá tam bola odoslaná.

**Testovanie**

Na otestovanie bolo potrebné implementovať aj druhý príbeh – Vybratie textovej informácie z transakčného systému. Pre opis testovanie pozri opis tohto príbehu.

## *Vybratie textovej informácie z transakčného systému*

Ako používateľ chcem vybrať informáciu zo servera, aby som vedel, čo mi tam predchádzajúci používateľ zanechal.

**Analýza problému**

Aby sme mohli informácie z transakčného systému vyberať, potrebujeme riešiť niekoľko problémov:

- Prenos informácie medzi klientom a serverom – je bližšie opísaný v časti Vloženie informácie do transakčného systému.
- Vyzdvihnutie informácie zo servera prostredníctvom jej identifikátora. Môžu nastať aj chybové stavy, keď sa informácia so zadaným identifikátorom na serveri nenachádza.
- Zobrazenie získanej informácie na klientovi.

**Návrh riešenia**

*REST*
Ako indikáciu neexistujúceho zdroja použijeme HTTP kód 404 Not Found. Podrobný opis komunikácie je nasledovný:
**Požiadavka**
**HTTP Metóda**: GET
**URI**: /strings/<identifikačné_číslo>
**Telo:** prázdne

**Odpoveď**
Úspešné vykonanie požiadavky
**HTTP kód:** 200 OK
**Telo:** Uložený reťazec

Neexistujúci reťazec
**HTTP kód:** 404 Not Found
**Telo:** prázdne

**Opis implementácie**

*REST*
Pri implementácii sme rozšírili triedu InMemoryStringServerResource o ďalšiu metódu @Get retrieve(). Okrem toho bolo potrebné rozšíriť smerovač o ďalšiu cestu, s jedným variabilným parametrom, ktorým je identifikátor reťazca.

**Testovanie**

Navrhli sme tento testovací scenár:

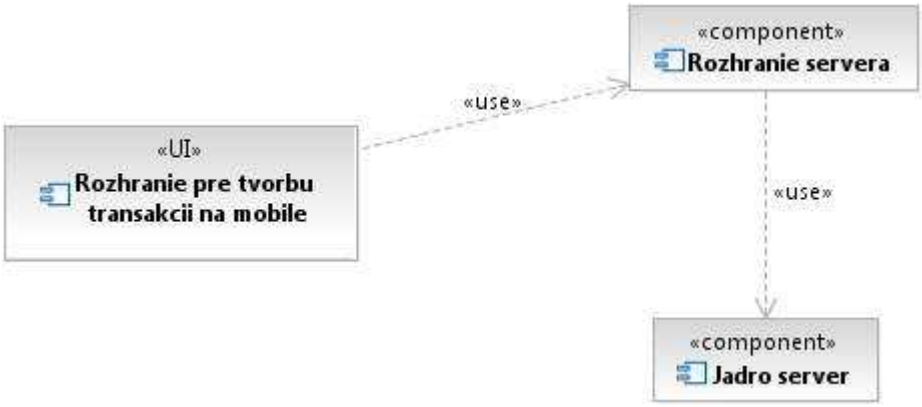
1. Na klientskom zariadení sa odošle požiadavka na vloženie vymysleného reťazca do systému. Očakávaným výstupom je odpoveď s kódom reprezentujúcim úspešné vloženie reťazca a identifikátor reťazca v tele odpovede. Zároveň je potrebné zapamätať si tento identifikátor.
2. Na klientskom zariadení sa odošle požiadavka na vybratie reťazca so zapamätamým identifikátorom zo systému. Očakávaným výstupom je odpoveď, ktorá v tele obsahuje reťazec zhodný s vymysleným reťazcom z kroku 1.

Výstup v oboch krokoch bol podľa očakávania.

## *Zhrnutie šprintu č. 1*

Cieľom tohto šprintu bolo vytvoriť takzvanú HelloWorld transakčnú aplikáciu, kde bude možné odoslať správu na server a potom ju získať naspäť. Na základe vyššie spomínaných informácií sa podarilo túto úlohu tímu splniť. Bola vytvorená aplikácia, ktorá na klientovi – mobilnom zariadení umožní zadať informáciu vo formáte reťazca. Túto informáciu pošle na server, kde sa uloží a je ju možné pomocou identifikačného čísla získať ďalším mobilným zariadením iPhone.

Po 1. Šprinte sa nám podarilo identifikovať nasledovnú architektúru systému(Obrázok  2).




**Obrázok 2Architektúra systému po 1. šprinte**

# Šprint č. 2

Cieľom druhého šprintu je posunúť prvotnú aplikáciu, vytvorenú v prvom šprinte ďalej. Potrebné je začať používať maticové kódy, v ktorých bude uložená informácia o uložení objektu na serveri. Úlohou teda bude vygenerovať maticový kód jedným zariadením a prijať druhým zariadením. Na strane servera je potrebné začať používať JSON objekty a umožniť registráciu používateľov do systému pomocou nich.
V rámci tohto šprintu boli identifikované nasledovné príbehy:

- Registrácia používateľov
- Vloženie informácie do transakčného systému a zobrazenie 2D kódu
- Vloženie komplexnej informácie do transakčného systému
- Vybratie komplexnej informácie z transakčného systému

## *Registrácia používateľov*

Ako používateľ sa chcem zaregistrovať, aby som mal prístup k funkcionalite pre registrovaných používateľov.

### Analýza problému

Identifikovali sme nasledujúce problémy:

1. Je potrebné navrhnúť, ako sa dáta budú na serveri ukladať, s dôrazom na nasledujúci bod.
2. Keďže medzi registračné údaje patria aj heslá, je potrebné zabezpečiť, aby sa na serveri neukladali ako čistý text (zaistenie aspoň minimálnej bezpečnosti).
3. Pri registrácii môže nastať niekoľko chybových stavov – zadaný používateľ je už registrovaný, alebo niektoré údaje nespĺňajú požadovaný formát. Tieto situácie je potrebné ošetriť.
4. Je veľmi pravdepodobné, že spôsob registrácie a požadované údaje sa budú v budúcnosti meniť. Preto by bolo vhodné oddeliť registráciu od mobilného klienta a registráciu realizovať prostredníctvom webovej stránky (vytvoriť a upravovať ju sa nám v tejto fáze javí vhodnejšie ako neustále upravovať používateľské rozhranie na mobilnom klientovi).

### Návrh riešenia

*REST*
REST rozhranie zohľadňuje fakt, že informácie o registrácii sú štruktúrované (pozri príbeh *Vkladanie štruktúrovanej informácie*).

Podrobný opis rozhrania:
**Požiadavka**
**HTTP Metóda**: POST
**URI**: /user
**Hlavičky:** Content-type: application/json; charset=UTF-8
**Telo:** JSON objekt vo formáte:

```
{
  “name”: “meno_pouzivatela”,
  “password”: “zvolene_heslo_pouzivatela”,
```

```
  “mail”: “user@server.tld”
}
```

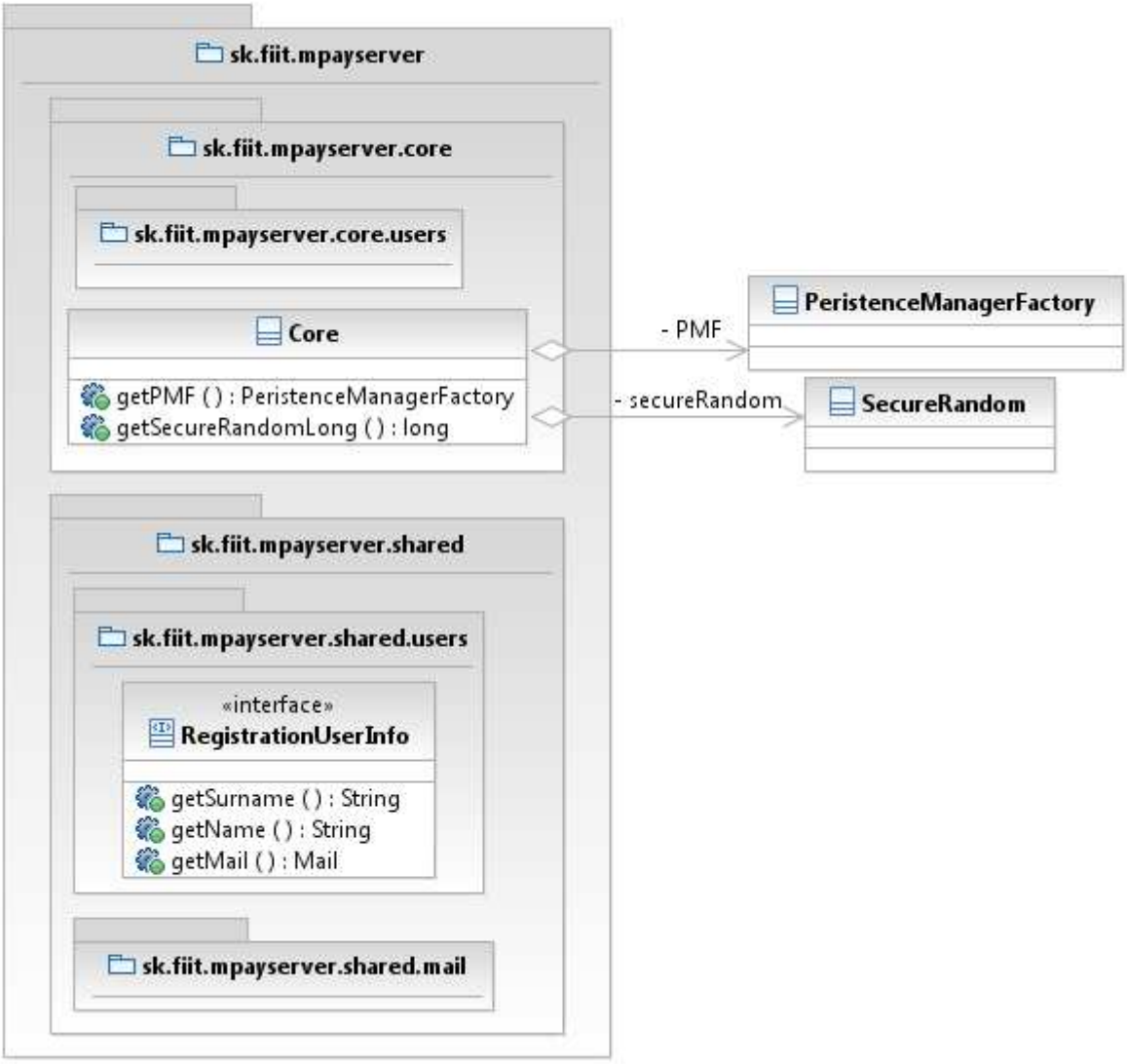
**Odpoveď**
Úspešné vykonanie požiadavky
**HTTP kód:** 201 Created
**Telo:** prázdne

Používateľ už existuje
**HTTP kód:** 409 Conflict
**Hlavičky:** Content-type: application/json; charset=UTF-8
**Telo:** JSON chybový objekt (pozri vyššie) so správou o chybe

Niektorý zo vstupných parametrov je chybný
**HTTP kód:** 400 Bad Request
**Hlavičky:** Content-type: application/json; charset=UTF-8
**Telo:** JSON chybový objekt so správou o chybe

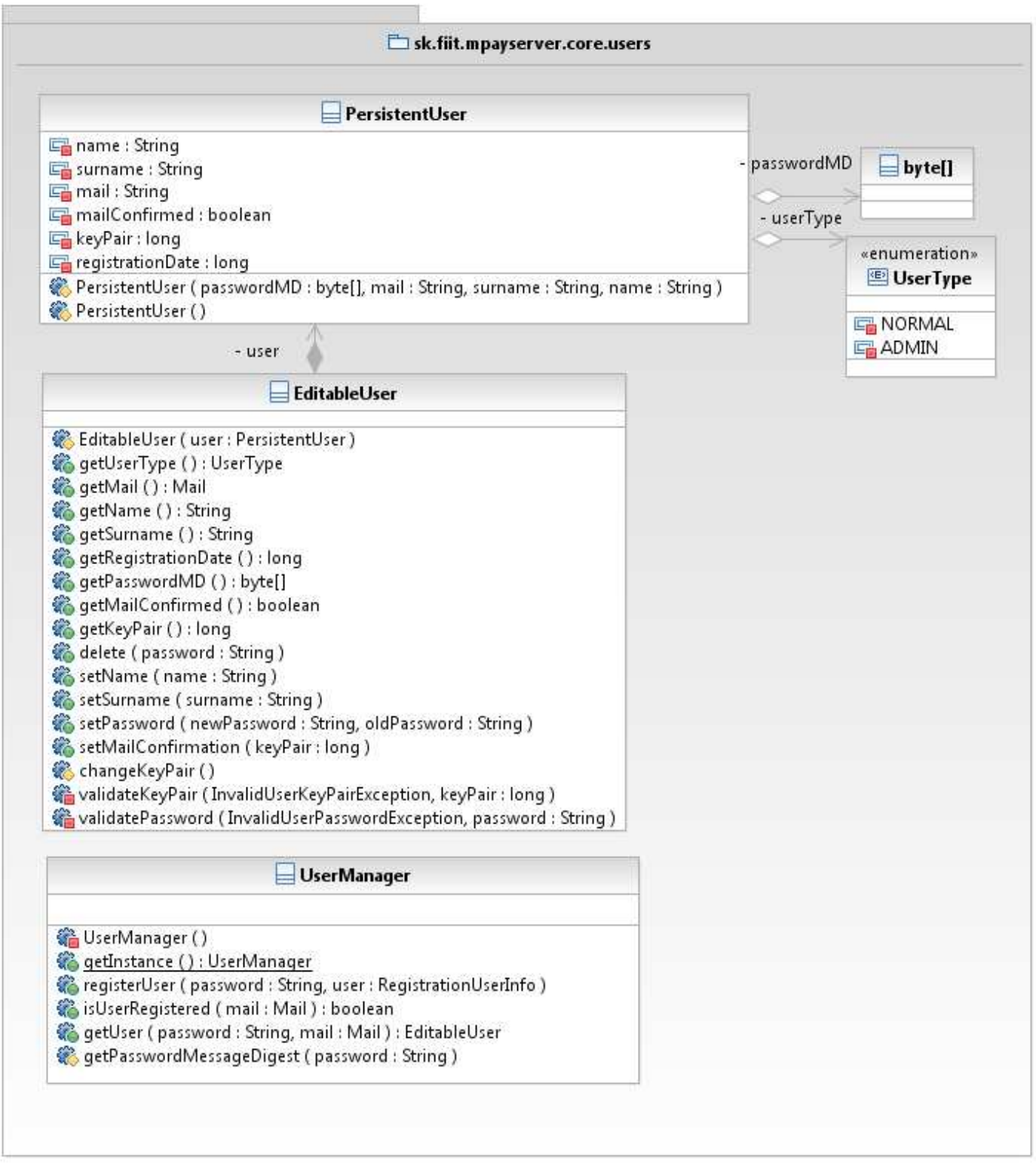
*Web rozhranie pre registráciu*
Pomocou tohto web rozhrania je potrebné získať od používateľa všetky informácie, potrebné k registrácii. Na to budú slúžiť klasické formulárové input boxy, do ktorých sa budú jednotlivé údaje vpisovať. Tieto údaje je potom potrebné z formulára získať a vytvoriť JSON objekt, ktorý bude posielaný na server na REST-ové rozhranie, kde bude ihneď vyhodnotený. To znamená, že sa zistia všetky potrebné detaily, prípadne sa odhalia konflikty, pre ktoré nemôže byť takýto používateľ registrovaný.
Zo servera sa vráti opäť JSON objekt, ktorý bude niesť informácie o možnosti registrovať používateľa. V časti REST je možné vidieť, aké kódy môžu byť vrátené. V závislosti od toho bude používateľ vytvorený alebo bude známa chyba, ktorá nastala.

*Jadro servera*
Jadro servera spracováva užívateľov, ukladá ich do perzistentnej pamäte, načítava ich a dovoľuje ich odstránenie a zmenu. V rámci jadra sa implementovala aj funkcionalita spoločná pre klienta a server.
Štruktúra jadra je zobrazená na obrázku 3. Objekty v balíku **sk.fiit.mpayserver.shared** sú použiteľné v časti klienta ako aj v časti servera. Nachádza sa tam rozhranie určujúce čo všetko musí zadať užívateľ pri registrácii (heslo je oddelené kvôli bezpečnosti). Objekty v balíku **sk.fiit.mpayserver.core** je možné použiť iba v časti servera. Nachádza sa tam trieda **Core** ktorá poskytuje základné služby pre prácu jadra a to generovanie bezpečných náhodných čísel a sprístupňovanie inštancie potrebnej pre perzistenciu objektov.



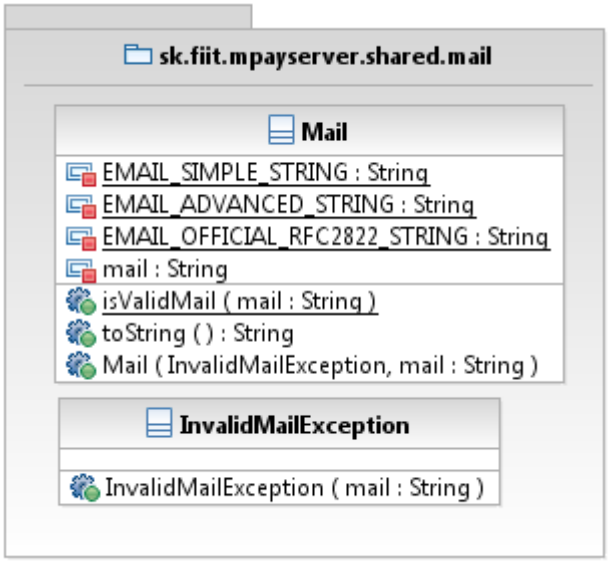

**Obrázok 3 Štruktúra balíkov jadra**

V balíku sa ďalej nachádza trieda **UserManager** (Obr.4) ktorá vykonáva samotné požiadavky na prácu s užívateľmi. Užívatelia sa ukladajú ako inštancie objektu **PersistentUser**. Následne je možné získať uloženého užívateľa vo forme inštancie triedy **EditableUser**, ktorá sa odkazuje na perzistentného užívateľa. V úložisku sa nenachádza uložené heslo ale iba jeho „MessageDigset“ podľa ktorého sa overuje správnosť hesla, čo zvyšuje bezpečnosť systému. Pre získanie užívateľa je potrebné poznať jeho heslo a mail. Z toho vyplýva že získať inštanciu užívateľa je možné iba s jeho pričinením. Triedy na získanie užívateľa bez jeho hesla zatiaľ neboli implementované, a budú implementované neskôr pri vytváraní administrátorskej funkcionality jadra. Pomocou **UserManager** objektu je možné zistiť či daný užívateľ už existuje (na základe jeho mailovej adresy). Pri registrácii je potrebné overiť mail užívateľa, táto funkcionalita zatiaľ nebola implementovaná.




**Obrázok 4 Balík sk.fiit.mpayserver.com a jeho triedy**

Server aj klient využívajú triedu **Mail** ktorá obaľuje triedu **String** a pridáva k nej podmienku validity mailovej adresy. Takto bola zaručená validita mailu v rámci systému. Taktiež je možné pomocou tejto triedy overovať validitu textového reťazca v prípade že nechceme vytvoriť inštanciu objektu **Mail**. UML diagram tohto objektu je na obrázku 5.




**Obrázok 5 Balík sk.fiit.mpayserver.shared.mail**

## Opis implementácie

*REST*
Implementácia je veľmi podobná prvému šprintu. Zdroj je definovaný pomocou rozhrania UserResource, ktoré implementuje trieda ServerUserResource. Metóda registrácia používateľa je implementovaná funkciou @Post create(RegistrationInfoDTO info). Objekt triedy RegistrationInfoDTO je automaticky deserializovaný zo vstupného JSON formátu.

*Web rozhranie pre registráciu*
Implementácia tejto časti bude realizovaná pomocou Google Web Toolkit. Pomocou neho bude vytvorený jednoduchý formulár, do ktorého bude môcť používateľ zadať všetky potrebné informácie. Následne bude implementované rozhranie ClientProxy, vďaka ktorému bude pomocou funkcie @Post create(RegistrationInfoDTO info) odoslaný JSON objekt na server. Po získaní odozvy zo servera bude zobrazená informácia o úspechu či neúspechu celého procesu.

## Testovanie

Na otestovanie tejto funkcionality sme navrhli tieto testovacie scenáre:

- Požiadavka na registráciu používateľa s vymyslenými údajmi, pričom mail je formálne správny a meno aj heslo sú vyplnené. Očakávaným výstupom je odpoveď s kódom reprezentujúcim úspešnú registráciu.
- Požiadavka na registráciu rovnakého používateľa ako v prípade 1. Očakávaným výstupom je zamietnutie registrácie z dôvodu existencie používateľa, odpoveď s chybovým kódom a správa o chybe v tele odpovede.
- Požiadavka na registráciu, kde mail nie je formálne správny. Očakávaným výstupom je odpoveď s chybovým kódom a správa o chybe v tele správy.
- Požiadavka na registráciu, kde chýba niektorý z parametrov {meno, heslo, mail}. Očakávaným výstupom je odpoveď s chybovým kódom a správa o chybe v tele správy.

Výstupy vo všetkých krokoch zodpovedali očakávaniam.

## *Vloženie informácie do transakčného systému a zobrazenie 2D kódu*

### Analýza problému

Tento príbeh pridáva funkcionalitu vygenerovania 2D maticového kódu ku príbehu "Vloženie informácie do transakčného systému" spracovávaného v šprinte 1. Na vygenerovanie 2D je potrebná knižnica, ktorá také niečo už vykonáva.

### Návrh riešenia

Návrh sa oproti predchádzajúcemu príbehu zmení iba tak, že keď používateľ pošle správu na server a príde mu odpoveď, tak okrem čísla, pod ktorým sa uchovala informácia na serveri sa zobrazí aj 2D kód, v ktorom bude toto číslo zakódované.

### Opis implementácie

Na implementáciu sme použili knižnicu Softmatic Barcode Generator. Použitím funkcií z tejto knižnice sme zakódovali informáciu, ktorá nám prišla zo servera a nastavili sme ju na objekt UIImage, ktorý predstavuje obrázok, a ktorý sa zobrazí na displeji používateľovi.

### Testovanie

Testovanie sa z časových dôvodov nestihlo vykonať špecializovanými prostriedkami. V budúcnosti sa plánuje nájsť voľne prístupný dekodér 2D kódov, uložiť vygenerovaný 2D kód do súboru a otestovať správnosť zakódovania.

## *Vloženie komplexnej informácie do transakčného systému*

## *Vybratie komplexnej informácie z transakčného systému*

Cieľom v týchto príbehoch je rozšíriť príbeh z prvého šprintu o možnosť vkladať a vyberať štruktúrované informácie. Je to najmä príprava pre ďalší šprint, aby si vývojári serverovej i klientskej časti vyskúšali prácu so štruktúrovanými dátami.

### Analýza problému

- V prvom šprinte sa do systému ukladali iba jednoduché texty, preto treba systém rozšíriť.
- Treba určiť formát, v ktorom sa budú dáta prenášať medzi klientom a serverom.
- Je potrebné navrhnúť, ako sa budú informácie na serveri ukladať.
- Je potrebné navrhnúť systém na serializáciu štruktúrovaných dát.

### Návrh riešenia

*Formát štrukturovaných informácií*
Keďže v systéme sa budú ďalej prenášať štrukturované informácie, je potrebné navrhnúť, v akom formáte sa budú prenášať. Zvolili sme formát JSON, keďže je jednoduchý, kompaktný a má dobrú podporu na strane klientov (je podmnožinou jazyka Javascript, ktorý je podporovaný v každom webovom prehliadači). Druhou alternatívou bol formát XML, ale zvolili sme JSON kvôli spomínanej kompaktnosti a jednoduchosti.

S touto zmenou súvisí aj formát chybových správ. Ten by mal obsahovať indikáciu, že došlo k chybe a tiež chybovú správu. Chybový objekt teda bude nasledovný:

```
{
  “error”: true,
  “errorMessage”: “Správa o chybe.”
}
```

V budúcnosti bude pravdepodobne potrebné pridať aj chybový kód, čo s navrhnutým formátom bude pomerne jednoduché.

*REST*
Rozhodli sme sa, že v tomto príbehu pri vkladaní informácie do systému zatiaľ nepotrebujeme iné informácie ako je vkladaný text. Preto sme nepridali ďalšie polia, iba sme pole s textom „obalili“ do JSON formátu. Pri vyberaní informácie zo systému sme pridali pole, ktoré nesie údaj o čase, kedy bola informácia vložená do systému.

Podrobný opis rozhrania:
**Požiadavka**
Vloženie štruktúrovanej informácie do systému
**HTTP Metóda**: POST
**URI**: /structured
**Hlavičky:** Content-type: application/json; charset=UTF-8
**Telo:** JSON objekt vo formáte:

```
{
  “text”: “textová informácia”
}
```

**Odpoveď**
Úspešné vykonanie požiadavky
**HTTP kód:** 201 Created
**Hlavičky:** Content-type: application/json; charset=UTF-8
Location: URI uloženej informácie
**Telo:** JSON objekt vo formáte:

```
{
  “id”: “identifikačné číslo, pod ktorým bola informácia uložená”,
  “uri”: “URI uloženej informácie”
}
```

**Požiadavka**
Vybratie štruktúrovanej informácie zo systému
**HTTP Metóda**: GET
**URI**: /structured/<identifikačné_číslo>
**Telo:** prázdne

**Odpoveď**
Úspešné vykonanie požiadavky
**HTTP kód:** 200 OK
**Hlavičky:** Content-type: application/json; charset=UTF-8
**Telo:** JSON objekt vo formáte:

```
{
  “text”: “uložená textová informácia”,
  “time”: “čas vloženia informácie do systému”
}
```

Informácia so zadaným identifikačným číslom na serveri neexistuje
**HTTP kód:** 404 Not Found
**Hlavičky:** Content-type: application/json; charset=UTF-8
**Telo:** JSON chybový objekt (pozri vyššie) so správou o chybe

*Spôsob ukladania informácií na serveri*
Keďže cieľom týchto príbehov je vyskúšať spracovanie štrukturovaných informácií na strane servera i klienta, spôsob ukladania bude rovnaký ako v prvom šprinte – do dátovej štruktúry v pamäti servera. Problémy tohto riešenia sú rovnaké ako v prvom šprinte, výhodou je veľmi rýchla implementácia.

*Klientska časť iPhone*
Keď sa bude posielať informácia na server, z klientskej časti sa odošle JSON objekt, ktorý bude obsahovať jedno pole, tak ako je to uvedené v rozhraní. Toto pole zadá používateľ a bude predstavovať správu, ktorá sa odošle na server. Naspäť príde odpoveď, ktorá bude tiež vo forme JSON objektu. Odpoveď bude obsahovať viacero polí (podľa rozhrania), ale aplikácia z toho použije iba pole "id", podľa ktorého bude možné nájsť uloženú informáciu na serveri.
Keď bude chcieť používateľ získať správu aj s časom vloženia tejto správy na server, zadá ID, to sa použije v adrese URL, na ktorú pôjde požiadavka na vybratie správy. Naspäť na klientskú aplikáciu znovu príde komplexná informácia s dvomi poľami (čas a správa), vytiahne sa z nej iba správa a tá sa zobrazí používateľovi.

**Opis implementácie**

*REST*
Keďže dáta sa prenášajú vo formáte JSON a v kóde sa pracuje priamo s Java objektmi (kvôli elegancii, čistote kódu, typovosti atď. nepracujeme priamo s JSON objektmi), je potrebné zabezpečiť deserializáciu z formátu JSON. Rozhodli sme sa použiť knižnicu Jackson, ktorá má podporu priamo vo frameworku Restlet a tým umožňuje deserializovať transparentne. Konfigurácia spočíva iba v pridaní knižnice do classpath.
Okrem toho bolo potrebné zaistiť aj to, aby boli chybové stavy tiež reprezentované vo formáte JSON. Momentálne sa do JSON formátu prevádzajú chyby, ktorých výskyt môže používateľ ovplyvniť (tzv. client errors, teda HTTP kódy 4xx). Chyby z kategórie 5xx (vnútorné chyby servera) sa zatiaľ takto neprevádzajú. K tomuto rozhodnutiu sa dospelo počas implementácie a bude ho potrebné ešte prehodnotiť. Toto sa vo frameworku Restlet realizuje pomocou tzv. StatusService. Ich úlohou je vrátiť pre daný chybový stav aplikácie vhodnú reprezentáciu. Ako reprezentácia chybového stavu sa používa objekt triedy JsonErrorRepresentation.

Konkrétne je na serveri štruktúrovaná informácia implementovaná pomocou triedy StructuredInfoServerResource. Dôležitými metódami sú @Get StructuredInfo retrieve() a @Post void store(StructuredInfo info).

*Klientska časť iPhone*
Na prácu s JSON objektmi sme použili triedu NSDictionary. Premennej tejto triedy sme priradili objekt označený ako "text" a k nemu sme priradili hodnotu z textového poľa, ktoré vyplnil používateľ. Toto sme odoslali podobne ako v predchádzajúcej časti, kedy sme posielali iba obyčajný reťazec s tým, že boli zmenené atribúty HTTP požiadavky (pozri časť návrh REST, podrobný opis rozhrania) a JSON objekt sme pomocou triedy JSONRepresentation pretransformovali do reťazca, ktorý sa mohol nastaviť ako telo požiadavky.

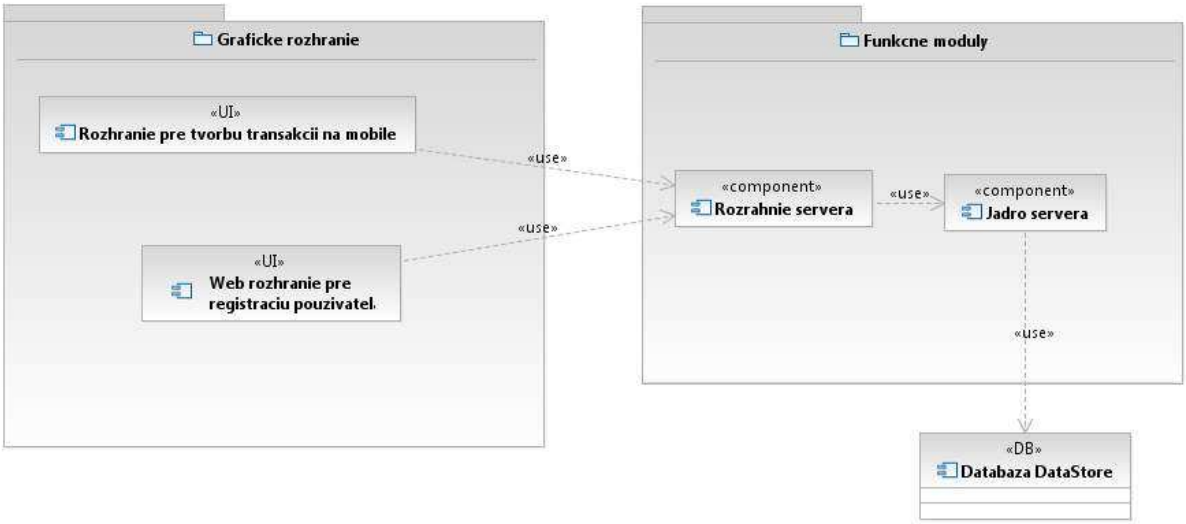
Podobne sa vykonala implementácia aj v opačnom smere, kedy sme oproti predchádzajúcemu príbehu (vybratie textovej informácie) nenastavili ako správu na zobrazenie jednoducho obsah odpovede, ale sme to museli pretransformovať pomocou JSONValue na JSON objekt, z ktorého sme pomocou NSDictionary vytiahli správu na zobrazenie používateľovi.

### Testovanie

Testovacie scenáre sú rovnaké ako v prvom šprinte, preto ich na tomto mieste neuvádzame.

## *Zhrnutie šprintu č. 2*

Cieľom tohto šprintu bolo obohatiť doteraz vytvorenú aplikáciu o používanie JSON objektov, generovania 2D kódu a odoslanie komplexnej informácie na server. Taktiež bolo potrebné vytvoriť web stránku na registráciu používateľov. Všetky tieto úlohy sa podarilo v tomto šprinte splniť.
Po tomto šprinte architektúra systému vyzerá nasledovne (Obrázok 6).




**Obrázok 6 Architektúra systému po 2. šprinte**

# 2.časť

# Riadenie projektu

# Obsah

# Úvod

## *Prehľad dokumentu*

Táto časť dokumentácie pojednáva o jednotlivých dokumentoch k riadeniu projektu, ktoré postupne vznikajú pri vývoji produktu. Zvyšok tejto dokumentácie je radený nasledovne:

Kapitola ***Plán projektu*** obsahuje predpokladaný plán pre celý projekt. Plán pre celý projekt je na abstraktnej úrovni a uvedené sú len základné obrysy plánu. Uvedený je ale konkrétny plán pre nasledujúce 4 týždne, v ktorom je možné nájsť konkrétne úlohy pre dané obdobie.

V kapitole ***Úlohy členov tímu*** sa nachádza rozdelenie úloh pre jednotlivých členov tímu, pričom sú špecifikované aj mená autorov jednotlivých častí inžinierskeho diela (1. časť Softvérový systém). Taktiež sa tu nachádzajú roly, pridelené jednotlivým členom tímu pre najbližšie obdobie.

Kapitola ***Metodiky potrebné pri vývoji*** zahŕňa všetky metodiky, použité pre tvorbu produktu. Ide najmä o štandardy kódovania, štandardy písania projektovej dokumentácie, štandardy manažmentu úloh.

V kapitole ***Manažment verzií, konfigurácií a zmien*** sa nachádzajú metodiky pre manažment verzií, konfigurácií i zmien. Kapitola je zameraná najmä procesy, súvisiace s týmito úlohami.

V prílohe A - ***Ponuka*** je ponuka tímu, vďaka ktorej bola tímu pridelená daná téma.

Príloha B - ***Záznamy zo stretnutí*** obsahuje zápisy, vytvorené po každom stretnutí. Tieto zápisy obsahujú podrobný opis jednotlivých stretnutí, pričom sú v nich špecifikované úlohy do ďalšieho šprintu a taktiež sú vyhodnotené úlohy z predchádzajúceho stretnutia.

Príloha C - ***Preberacie protokoly*** obsahuje protokoly, podpísané zákazníkom (vedúcim tímu), ktoré slúžia na potvrdenie prebratia dokumentu zákazníkom.

# Plán projektu

## *Dlhodobý plán pre celý projekt*

Dlhodobý plán je vytvorený na dvojšprintovej granularite.

**Zimný semester**

1. Koniec 2. šprintu:
   - prenos komplexnej informácie medzi serverom a klientom
   - zobrazovanie 2D kódu
   - web stránka tímu
   - možnosť registrovania používateľov
2. Koniec 4. šprintu:
   - funkčný prototyp - na server sa odošle správa, server vráti kód, z ktorého sa vygeneruje 2D kód. Tento 2D kód sa zosníma z iPhone, dekóduje a pošle ďalšiu správu na server.
   - subsystém na prezeranie transakčného systému
3. Koniec zimného semestra:
   - skompletizované rozhranie

**Letný semester**

1. Koniec 2. šprintu:
   - Autentifikácia používateľa, registrácia cez certifikáty
2. Koniec 4. šprintu:
   - Aplikácia bude  prenášať informácie zodpovedajúce realizácii transakcií
3. Koniec letného semestra:
   - Prototyp aplikácie simulujúcej mobilné bankovníctvo

## *Krátkodobý plán na 4 týždne*

1. Prvý týždeň 3. šprintu:
   - Naštudovanie knižnice na dekódovanie 2D kódu a možnosti snímania obrázkov kamerou
   - rozpracovanie systému na prihlasovanie používateľov
2. Druhý týždeň 3. šprintu:
   - implementácia snímania a dekódovania 2D kódu
   - dokončenie systému na prihlasovanie používateľov
   - implementovanie spravovania používateľov
3. Prvý týždeň 4. šprintu:
   - opravenie prípadných nedostatkov snímania a dekódovania 2D kódu

- dopracovanie prípadných nedostatkov spravovania používateľov
- vytvorenie systému na ukladanie informácií o transakciách

4. Druhý týždeň 4. šprintu:

- na zariadení, ktoré zosnímalo 2D kód sa bude môcť použiť dekódovaná informácia na ďalšiu transakciu
- dopracovanie systému na ukladanie transakcií
- vytvorenie možnosti zobrazenia transakcií

### *Porovnanie plánu s aktuálnym postupom*

Aktuálne výsledky zodpovedajú stanovenému plánu. Samozrejme, ani náš projekt sa nezaobišiel bez ťažkostí, ale od žiadnej sa neočakáva ovplyvnenie výsledkov dlhodobého plánu.

# Úlohy členov tímu

## Dlhodobé úlohy

Vedúci tímu - Matej Lipták

Manažér vývoja - Miroslav Čorba

Manažér plánovania - Branislav Hašto

Manažér kvality - Roman Pipík

Manažér podporných prostriedkov - Lukáš Lipka

## Krátkodobé úlohy

Pre 1. a 2. šprint mali členovia tímu nasledovné úlohy:

***Matej Lipták*** - dokumentácia (časť z inžinierskeho diela, časť z riadenia projektu), programovanie klientskej časti aplikácie

***Miroslav Čorba*** - dokumentácia (časť z inžinierskeho diela, časť z riadenia projektu, jej sumarizovanie do jedného celku), programovanie serverovej časti aplikácie, hlavne rozhranie

***Lukáš Lipka*** - dokumentácia (časť z riadenia projektu), programovanie klientskej časti aplikácie

***Branislav Hašto*** - dokumentácia (časť z inžinierskeho diela), programovanie serverovej časti aplikácie

***Roman Pipík*** - dokumentácia (časť z riadenia projektu), programovanie serverovej časti aplikácie

# Metodiky potrebné pri vývoji

## *Všeobecné pravidlá*

Ak niekto pri implementácii narazí na nejaký problém a nevie identifikovať chybu do 30 minút, musí niekoho vyhľadať, aby mu pomohol. Ak sa mu nepodarilo do tohto času nájsť chybu, je pravdepodobné, že s ňou strávi ešte dosť času. Naopak, niekto druhý tú chybu môže zbadať hneď. Nikto sa preto nesmie hanbiť zavolať niekoho na pomoc, bude to pre tím oveľa prospešnejšie - problém bude skôr vyriešený a do funkcionality bude zapojený aj niekto ďalší (to pomôže profesijnému rastu členov, ich zlepšovaniu v odhadovaní a iné).

### Interná komunikácia

Najdôležitejšie rozhodnutia budú vykonané na oficiálnych stretnutiach, ktoré sa odohrávajú v softvérovom štúdiu na fakulte. Tu bude priestor na prediskutovanie najhlavnejších problémov na abstraktnej vrstve.

Ostatné problémy môžu byť prediskutované na osobných stretnutiach podľa dohody, alebo na to môže použiť Instant Messaging, prípadne mail. Ak je to veľmi súrne a veľmi dôležité, môže použiť aj telefónny kontakt. Ak je to informácia, ktorá sa týka všetkých, pošle to na mailing list tímu, ktorý má adresu `fiit-tp-2010[at]googlegroups[dot]com`. Ak sa to netýka všetkých, ale iba určitej podmnožiny členov, tak na to použije priamo ich email. V takom prípade v predmete správy uvedie, že ide o tímový projekt značkami "TP", "Team Project", alebo "Tímový projekt".

## *Pravidlá pre vytváranie programov*

### Názvoslovie

Nižšie uvedené pravidlá budú platiť hlavne pre názvy tried, metód, atribútov, konštánt, ale aj pre názvy súborov, balíkov a iných entít, ktoré bude potrebné pomenovať.

Názvy budú písané v anglickom jazyku; nebudú sa používať skrátené formy, ale celé názvy. Zložené názvy budú písané štýlom CamelCase, teda slová budú spojené do jedného s tým, že začiatočné písmeno ďalšieho slova bude veľké, ostatné písmená budú malé. Ak bude názov obsahovať aj skratku (napríklad pre "get HTTP Request"), prvé písmeno skratky bude veľké, ostatné malé (v našom prípade "getHttpRequest").

Slová v názvoch balíkov sa budú podľa potreby oddeľovať bodkami. Napríklad:

```
mpay.client.reusable
```

### Komentovanie kódu

Kód je potrebné dôsledne komentovať. Ku každej triede by mal byť popis - čo je jej úlohou. To isté platí aj pre metódy, ku nim je však potrebné pridať aj popis parametrov, prípadne návratovej hodnoty. Pre serverovú časť, ktorá bude naimplementovaná v Jave sa použijú pravidlá pre javadoc, pre klientskú časť sa použijú pravidlá pre doxygen.

### Členenie kódu

Každá entita musí vykonávať iba tú funkcionalitu, na ktorú je určená. Je lepšie vytvoriť viacero tried tak, aby každá vykonávala iba to, čo o nej napovedá jej názov. To isté platí aj pre metódy. Okrem toho, žiadna metóda by nemala byť príliš dlhá. Ideálna dĺžka metódy je do 10 riadkov, ak je dlhšia, treba zvážiť jej rozčlenenie na viac častí.

Triedy a súbory je potrebné rozumne rozdeliť do balíčkov. Treba rozlišovať medzi časťami systému, ktoré súvisia s dátovou časťou, rozhraním, logickou časťou, časťami systému, ktoré sa môžu znovu použiť aj v iných aplikáciách a podobne.

## *Java a serverová časť*

### Štruktúra projektu

Projekt sa skladá z troch vrstiev, podľa čoho sú definované aj balíky: *sk.fiit.mpayserver.core* - jadro servera, riadi spracovanie údajov na serveri, načíriadiace ukladanie, načítavanie dát, ... *sk.fiit.mpayserver.rest* - REST API, pre komunikáciu klienta (web, mobil) so serverom *sk.fiit.mpayserver.gui* - web rozhranie servera, určené pre registráciu užívateľov, administráciu a podobn všetky triedy by mali patriť do jedného z týchto balíkov. Štruktúra podbalíkov záleží od implementovanej funkčnosti!

Adresáre v rámci projektu sa riadia štruktúrou ktorú používa GWT, čiže máme tri základné adresáre: *src, test, war*.

### Javadoc

Každý element programu s prístupom public alebo protected má byť okomentovaný pomocou prostriedku Javadoc. Pre triedy to predstavuje značku *@author* Pre metódy treba uviesť značky *@param*, *@return* a *@throws* (záleží od metódy) Pre polia tried treba uviesť aspoň význam Použitie ostatných značiek je dobrovoľné, je vhodné použiť značku *@see* pri odkazovaní sa na inú časť programu

### Anotácie

V tejto časti si ozrejmíme používanie anotácií z balíka java.lang . Je potrebné používať základné anotáciu *@Override* pre akékoľvek prekrytie metódy (aj metódy rozhrania). Ak je určitá časť programu zastaralá označí sa anotáciou *@Deprecated*, bez toho aby bola odstránená. K odstráneniu takejto metódy môže dôjsť až po 7 dňoch od označenia, aby sa ostatný členovia tímu dokázali prispôsobiť zmenám. Anotácia *@Deprecated* by mala následne obsahovať popis prečo je použítá a ako nahradiť zastralý program za nový. Anotácu *@SupressWarning?* nieje možné používať, nakoľko jej použitie môže spôsobiť problémy.

## *Pravidlá pre Agilo*

### Pridávanie úloh

Každý nový príbeh (User Story) musí byť namapovaný na príslušnú požiadavku na aplikáciu (Requirement). Keď niektorá úloha (Task) implementuje časť príbehu, musí byť namapovaná

na tento príbeh. Niektoré príbehy, ktoré nebudú implementovať žiadny príbeh nebudú namapované na nič.

### Prideľovanie úloh

Pri vytváraní šprintu je dôležité, aby boli úlohy čo najskôr pridelené jednotlivým členom tímu, aby každý vedel čo má na starosti a aby sa nezabudlo prideliť nejakú úlohu. Úlohou každého člena tímu je, aby si pre každú svoju úlohu urobil odhad koľko hodín mu bude trvať vypracovanie danej úlohy. Toto je dôležité urobiť čo najskôr po pridelení úlohy, najneskôr v daný deň.

### Zadávanie hodín

Je veľmi dôležité, aby sa hodiny zadávali hneď po nejakej zmene. Tak ako sa pri prideľovaní úloh zadajú hodiny hneď, keď sa pridelia, tak sa musia zadať aj hodiny určujúce zostávajúci čas a čas strávený prácou na úlohe. Tým, že sa zadajú okamžite sa predíde viacerým problémom. Človek si lepšie pamätá koľko hodín strávil s danou úlohou, je ešte v obraze, a preto vie lepšie urobiť odhad zostávajúceho času na dokončenie úlohy. Okrem toho tým dá vedieť ostatným členom tímu, že na danej úlohe pracoval, prípadne ich môže touto cestou informovať o stave úlohy (ak ich úlohy závisia od tejto úlohy a potrebujú počkať na jej dokončenie). Potom si môžu aj lepšie zorganizovať čas.

Tieto hodiny sa budú zadávať takým spôsobom, že si používateľ v systéme Agilo dá vyhľadať aktuálne úlohy (Active Tickets), tam si vyberie úlohu, na ktorej pracoval, zmení pohľad na upravovanie (Edit) a naraz zmení aj zostávajúci čas, aj čas strávený prácou na úlohe. Okrem týchto dvoch polí napíše aj komentár, ktorým vysvetlí v akom štádiu je úloha, čo sa tam zmenilo a iné veci, ktoré môžu informovať ostatných členov tímu o stave úlohy.

Členovia tímu by mali zadávať hodiny s rozumnou granularitou. Najmenší časový úsek, ktorý sa zadá sa odporúča na 0.5 hodiny. Ak má nejaká úloha trvať viac ako 8 hodín, stojí za zváženie rozdeliť ju na viac častí.

# Manažment verzií, konfigurácií a zmien

## *Práca s SVN*

### Načítanie z SVN

Príkazy na načítanie z SVN repozitárov:

```
svn co svn://147.175.159.182/team08/svn/mpay/trunk mpay
svn co svn://147.175.159.182/team08/svn/mpay-server/trunk mpay-server
svn co svn://147.175.159.182/team08/svn/docs/trunk docs
```

### Ukladanie do SVN

Pri ukladaní nových súborov do repozitára je potrebné dodržiavať nasledovnú štruktúru. Súbory z klientskej časti aplikácie majú vyhradený repozitár `mpay`. Serverová časť aplikácie bude používať repozitár `mpay-server`. Pre ostatné súbory, ktoré budú predstavovať väčšinou textové dokumenty je určený repozitár `docs`.

V rámci repozitára SVN sú pre každý modul dostupné tri adresárové štruktúry (resp. vývojové vetvy):

- trunk - predstavuje hlavnú vývojovú vetvu (tzv. "bleeding-edge development"). Jedná sa o vetvu, do ktorej sa pridáva nová funkcionalita a následne snaha o jej stabilizáciu.
- branches - predstavuje experimentálne vývojové vetvy, ktoré markantným spôsobom menia obsah jednotlivých modulov. Následne v prípade ich úspešnosti budú zlúčené s hlavnou vetvou *trunk*.
- tags - služia na označenie jednotlivých *milestones*, resp. *releases*. (Priklad: mpay-server-0.0.1)

Pri pridávaní nových súborov do repozitára musí prispievateľ zaradiť daný súbor do štruktúry tak, aby sa zachovala prehľadnosť jednotlivých repozitárov. Napríklad pri pridávaní novej zápisnice sa táto uloží v repozitári `docs` do priečinka `zapisnice` a podobne.

Prikaz pre uloženie:

```
svn commit -m "Správa o importe" <adresár>
```

# Prílohy

# Príloha A - Ponuka tímu

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

# Ponuka

**Matej Lipták**

**Lukáš Lipka**

**Miroslav Čorba**

**Branislav Hašto**

**Roman Pipík**

Študijný program: Softvérové Inžinierstvo
Ročník: 1.
Predmet: Tímový Projekt
Ak. rok: 2010/2011
Tímový mail: fiit-tp-2010@googlegroups.com

# 1. Členovia tímu

## 1.1. Matej Lipták

Najčastejšie programujem v jazyku Java SE, avšak veľa školských projektov som robil aj v C/C++. Pokročilé znalosti mám aj z jazyka Lua, bash a modelovania v UML. Základy mám z množstva ďalších jazykov – Lisp, Prolog, Ruby, MySQL. Mám rád linux, momentálne pracujem na pozícii PL/SQL developer. V tomto semestri mám zapísané predmety Pokročilé databázové technológie, Objektovo orientovaná analýza a návrh softvéru.

## 1.2. Lukáš Lipka

Programovaniu sa aktívne venujem už približne 7 rokov. Za túto dobu som mal možnosť pracovať vo viacerých komunitách a v rámci vnútropodnikových tímových oddelení. Na výbornej úrovni ovládam programovacie jazyky C/C++, C# a Java. Mal som možnosť pracovať na platformách DOS, Windows a Linux, a vytvárať aplikácie prenositeľné medzi týmito platformami. S veľkou mierou som sa podieľal na vedení, koordinovaní a vývoji v open-source komunitách. Jedná sa konkrétne o nasledujúce projekty:

• oZone (http://ozonegui.sf.net/) – GUI rozhranie pre DOS/Windows/Linux, získalo ocenenie v LinuxOnPower
• Beagle (http://beagle-project.org/) – fulltextové indexovanie a vyhľadávanie pre Linux desktop
• Dashboard (http://code.google.com/p/dashboard/) – zobrazovať kontextovo relevantné údaje pri práci používateľa

## 1.3. Miroslav Čorba

Zaujímam sa najmä o objektovo orientovanú paradigmu programovania, ale pracoval som už aj s inými paradigmami, najmä s procedurálnou. Z programovacích jazykov mi je najbližšia Java, pracoval som už ale aj v C, C++ a taktiež mám základy v jazyku PHP či Adobe Flex. Ešte viac ako programovanie ma ale zaujalo modelovanie pomocou UML 2.0. V tomto semestri mám zapísané predmety Pokročilé databázové technológie a Objektovo orientovaná analýza a návrh softvéru.

## 1.4. Branislav Hašto

Zaujímam sa hlavne o web a webové technológie. V tejto oblasti mám niekoľkoročné skúsenosti aj z komerčnej praxe. Veľmi dobre ovládam jazyky Java, PHP, HTML a CSS. Pri programovaní si zakladám na čistote a kvalite kódu a dodržiavaní „best practices“. Mimo webovej oblasti mám z práce na bakalárskom projekte skúsenosti so spracovaním textu, sémantikou a ontológiami. Dlhšiu dobu sa zaujímam o agilné metódy vývoja softvéru, ale zatiaľ som nemal možnosť vyskúšať si ich v praxi.

## 1.5. Roman Pipík

Mojou primárnou oblasťou je jazyk Java (SE,EE) a XML. Mám základné znalosti jazyka SQL, ktorý som využíval s jazykom Java (a serverom IBM DB2). Ako študent bakalárskeho štúdia v odbore PSS som získal aj hardvérovo orientované skúsenosti a som v tejto oblasti rozhľadený od assembleru (8051, 8086) až po PLC . Pri vývoji sa snažím o kvalitnú dokumentáciu (Javadoc, UML ), kvalitný a efektívny program. Využívam hlavne nástroje od firmy IBM. Posledné mesiace som sa venoval vývoju aplikácie pre tvorbu a objednávanie foto-produktov. V súčasnosti sa zaujímam o umelú inteligenciu, tvorbu používateľských rozhraní a Aspektovo orientované programovanie.

# 2. Objektové úložisko dát

## 2.1. Motivácia

S pokrokom doby a nástupom nových technológií sa činnosť práce človeka s počítačom menila. Menia sa používateľské rozhrania, zjednoduchšujú a zefektívňujú sa pracovné postupy. Jedna časť však ale stále pretrváva viac-menej v tej istej nezmenenej podobe – súborový systém. Ten však s postupom času a neustále geometricky narastajúcim objemom dát, ktoré chce používateľ uložiť, začína zaostávať a spomaľuje jeho efektivitu.

Jednotlivé súbory – od fotiek, cez dokumenty až po naše obľúbené pesničky - sú roztrúsené po celom disku. Nie je ich možné žiadnym jednoduchým spôsob usporiadať a klasifikovať a používateľ strávi väčšinu času hľadaním jednotlivých súborov. Popritom sa strácajú dôležité informácie o súboroch – odkiaľ z internetu bol daný dokument stiahnutý; ktoré osoby z adresára kontaktov sa nachádzajú na danej fotke; miesto kde bol daný videoklip natočený; používateľ, ktorý nám cez instant-messaging zaslal danú aplikáciu a podobne. Všetky tieto „metadáta“ sa dajú použiť na zefektívnenie, spríjemnenie práce s počítačom a umožnia vytvárať nové druhy aplikácií, ktoré budú zohľadňovať tieto vzťahy a zdielať tieto informácie medzi sebou.

Potenciál je ukrytý v jednotnej relačnej databáze, ktorú by ako úložisko využívali všetky aplikácie dostupné na systéme. Umožňovala by ukladanie nie len obyčajných súborov, ale aj úplne nových dátových typov ako sú tímové projekty, divadelné predstavenia, cestovné trasy a iných, podľa potreby danej aplikácie. Vývojári aplikácií by sa nemuseli sústrediť na vytváranie vlastného úložiska dát, ale mali by prístup k jednotnému systému, ktorý by spĺňal všetky ich požiadavky. Umožňoval by vyhľadávať, spravovať, klasifikovať, tagovať jednotlivé objekty a prezentovať ich v presne takej forme, ako o nich používateľ uvažuje.

Možnosti takéhoto ukladania dát sú nekonečné – čo sa s nimi stane ďalej, spočíva v rukách používateľa.

## 2.2. Koncepcia riešenia

Riešenie je postavené na ukladaní objektov a ich potrebných atribútov v databáze. Súbory môžu byť ďalej ukladané na bežnom súborovom systéme. V databáze sa budú nachádzať len potrebné metadáta. Riešenie bude poskytovať API, ktorá bude figurovať ako vrstva medzi bežným súborovým systémom a aplikáciou.

Objekty budú zadefinované pomocou schém. Z týchto sa budú priamo generovať „first class“ objekty, ktoré budú môcť byť ďalej použité v aplikáciách. Rozšírenie a dodefinovanie vlastných objektov bude taktiež možné prostredníctvom schém, ktoré môžu rozširovať už existujúce objekty. Takto napr. z typu Document môže vzniknúť nový špecializovaný typ ContractDocument, ktorý bude obsahovať rozšírené atribúty týkajúce sa tohto typu dokumentu – napr. zmluvné strany a pod.
Medzi jednotlivými objektmi je možné zadefinovať rôzne vzťahy – napr. vzťah medzi dokumentom a emailom v ktorom bol priložený, alebo medzi kontaktom a fotkou, na ktorej sa nachádza. Tieto vzťahy budú taktiež uložené v databáze. Na základe týchto vzťahov bude možné taktiež implementovať funkcionalitu tagovania.

Pristupovať k objektom sa bude programaticky prostredníctvom vyhľadávania a následného filtrovania – na základe tagov, atribút a pod.

Keďže sa v databáze budú nachádzať aj citlivé informácie, bude potrebné klásť dôraz na bezpečnosť uložených údajov a zabrániť ich zneužitiu.

Riešenie by malo byť implementované platformovo nezávisle a prenositeľné medzi rôznymi platformami – Windows, Linux a prípadne Mac. Z tohto dôvodu uvažujeme použitie nasledovných technológií.

Implementačné jazyky:

- C# (.NET pod Windows, Mono pod Linux)
- Java

Databáza na ukladanie údajov:

- Lucene
- SQL databáza

# 3. Crowdsourcing verejných dát

## 3.1. Motivácia

Motiváciou pre prácu na tomto projekte je pre nás:

1. Sme mladí ľudia so záujmom o verejný život na Slovensku a tento projekt vidíme ako možnosť aktívne sa zapojiť do zlepšenia situácie. Keďže tendre vo verejnej správe sa často týkajú informačných technológií, je našim priamym záujmom, aby korupcia v tejto oblasti bola čo najnižšia.
2. Výborná možnosť naučiť sa programovací jazyk Ruby a framework Ruby on Rails, v ktorých vidíme veľký potenciál a budúcnosť.
3. Keďže na Slovensku sa agilné metódy vývoja softvéru používajú stále pomerne málo, nie je veľa možností vyskúšať si ich v praxi. Vzhľadom na svetové trendy si myslíme, že získané znalosti budú konkurenčnou výhodou v budúcnosti.
   Zároveň ako jednotlivci ani nemáme možnosť vyskúšať si všetky agilné metódy (TDD vo dvojici, párové programovanie), rovnako ani metodiku SCRUM.
4. Možnosť lepšie sa oboznámiť so správaním davu. Využitie davu, kolaborácia, obsah tvorený používateľmi, to všetko sú internetové trendy. Projekt vnímame aj ako možnosť rozšíriť si obzory a preskúmať aj netechnické záležitosti (využiteľné napr. v oblasti marketingu a pod.)

Môžeme teda povedať, že projekt považujeme za možnosť vytvoriť užitočný produkt použiteľný v praxi a popritom získať množstvo cenných skúsenosti a zvýšiť svoju cenu na trhu práce.

## 3.2. Koncepcia riešenia

Neplánujeme vytvoriť veľké a prekomplikované riešenie, ale skôr nástroj s malým množstvom užitočných, dobre vyladených a dobre použiteľných vlastností. Aby sme dosiahli tento cieľ, plánujeme:

- Časté konzultácie so zákazníkom (vedúcim témy). Od konzultácií si sľubujeme, že nám pomôžu vytvoriť kvalitnejší produkt podľa požiadaviek zákazníka.
- Časté uvoľňovanie nových verzií produktu – aby vôbec bolo o čom konzultovať. Na začiatku plánujeme uvoľniť prototyp so základnou najdôležitejšou funkcionalitou a potom vlastnosti prototypu neustále zdokonaľovať a postupne pridávať nové vlastnosti (pričom by sme chceli dôkladne zvážiť pridanie každej novej vlastnosti – radšej menej kvalitných vlastností). Úvodný súbor vlastností by sme museli prediskutovať s vedúcim práce a aj podrobne v rámci tímu, ale mohli by sem patriť napr. komentovanie, tagovanie, označovanie príbuzných dokumentov a pod.
- Chceme sa pokúsiť čo najskôr zapojiť dav – snažiť sa získať ľudí, ktorí by boli od prvých fáz zapojení do projektu a získavali by sme od nich nielen spätnú väzbu, ale hneď od začiatku aj dáta, ktoré budú základom celej aplikácie. Týmto dátam by sme potom mohli prispôsobiť aj obsah, zobrazovanie, používateľské rozhranie atď. Zdrojom podobných ľudí by mohli byť odporúčania expertov (aliancia Fair-play) alebo internetové diskusné fóra.
- Kód by sme v čo najväčšej miere chceli pokryť testami. Plánujeme tím rozdeliť na dve časti: prvá bude vytvárať testy a druhá funkcionalitu podľa týchto testov. Zloženie skupín by sme chceli pravidelne obmieňať.

# 4. Platforma pre realizovanie transakcií prostredníctvom mobilných zariadení

## 4.1. Motivácia

V dnešnej dobe plnej technológií je už aj nakupovanie či platba za služby veľmi jednoduché. Stačí mať pri sebe platobnú kartu a transakcia bude ihneď vybavená. Čo tak ale aj takúto činnosť ešte zjednodušiť? Čo tak nepoužiť platobnú kartu ale zariadenie, ktoré máme pravidelne pri sebe? Keďže mobilné telefóny sa dnes stávajú samozrejmosťou pre každého človeka, vzniká množstvo aplikácií, ktorými je možné vykonať mnohé služby. Možnosť vytvoriť aplikáciu pre mobilné telefóny, umožňujúcu simulovať mobilné bankovníctvo, je pre nás veľkou motiváciou. Dôvodov je viacero. Mnoho ľudí vrátane nás by totiž rado používalo pre vybavenie čo najväčšieho počtu svojich povinností jedno zariadenie. Keďže mobilné zariadenie máme vo väčšine prípadov so sebou, je ideálnym multitaskingovým prístrojom. Takáto aplikácia by znamenala veľký prevrat nielen pri využívaní mobilných telefónov ale znamenalo by to aj uľahčenie každodenného života pre väčšinu ľudí. Keďže sa nepýtame, čo môže informatika urobiť pre nás, ale čo môžeme urobiť my pre informatiku, pokúsili by sme sa urobiť pre nás malý, ale pre ľudstvo veľký krok v podobe implementovania aplikácie, simulujúcej mobilné bankovníctvo.

V súčasnosti sa mobilné zariadenia využívajú stále viac a majú veľkú perspektívu do budúcnosti. Preto ďalším dôvodom prečo chceme túto tému, je možnosť implementovať aplikáciu pre mobilné zariadenia. S takouto implementáciou zatiaľ nemáme žiadne skúsenosti a práve to nás viac motivuje vyskúšať si tvorbu softvéru, ktorý bude určený pre iné zariadenie, ako je počítač.

## 4.2. Koncepcia riešenia

Základom riešenia bude nadviazanie komunikácie medzi dvomi mobilnými zariadeniami pomocou grafického obrazca. Naše riešenie bude primárne určené pre mobilné telefóny. V dnešnej dobe má už drvivá väčšina mobilov zabudovaný fotoaparát, čo z neho robí potenciálny snímač grafického obrazca.

Grafický obrazec bude v našom prípade 2D čiarový kód. Existuje viacero typov 2D kódu (QR a DM), my si vyberieme DM (“Data Matrix”), ktorý predstavuje kombináciu dvoch typov zakódovanie informácie – priamej a nepriamej.

Proces snímania kódu sa skladá z troch krokov:

1. Používateľ odfotí kód
2. Obrázok sa zanalyzuje a vyhľadá sa čiarový kód – existuje viacero nástrojov vyhodnocovanie a dekódovanie 2D čiarových kódov. Telefóny značky Nokia majú v nových telefónoch predinštalovanú čítačku čiarových kódov, okrem toho existuje aj open source projekt Zxing, ktorý tiež dokáže skenovať 2D kód.
3. Používateľovi sa zobrazí zakódovaná informácia[1]

1 http://www.itnews.sk/temy/pcr-free-clanky/2009-12-28/132174-pcr-mobil-ako-snimac-ciaroveho-kodu

Na generovanie kódu môžeme využiť už existujúci softvér, ktorý dokáže takýto kód vygenerovať.

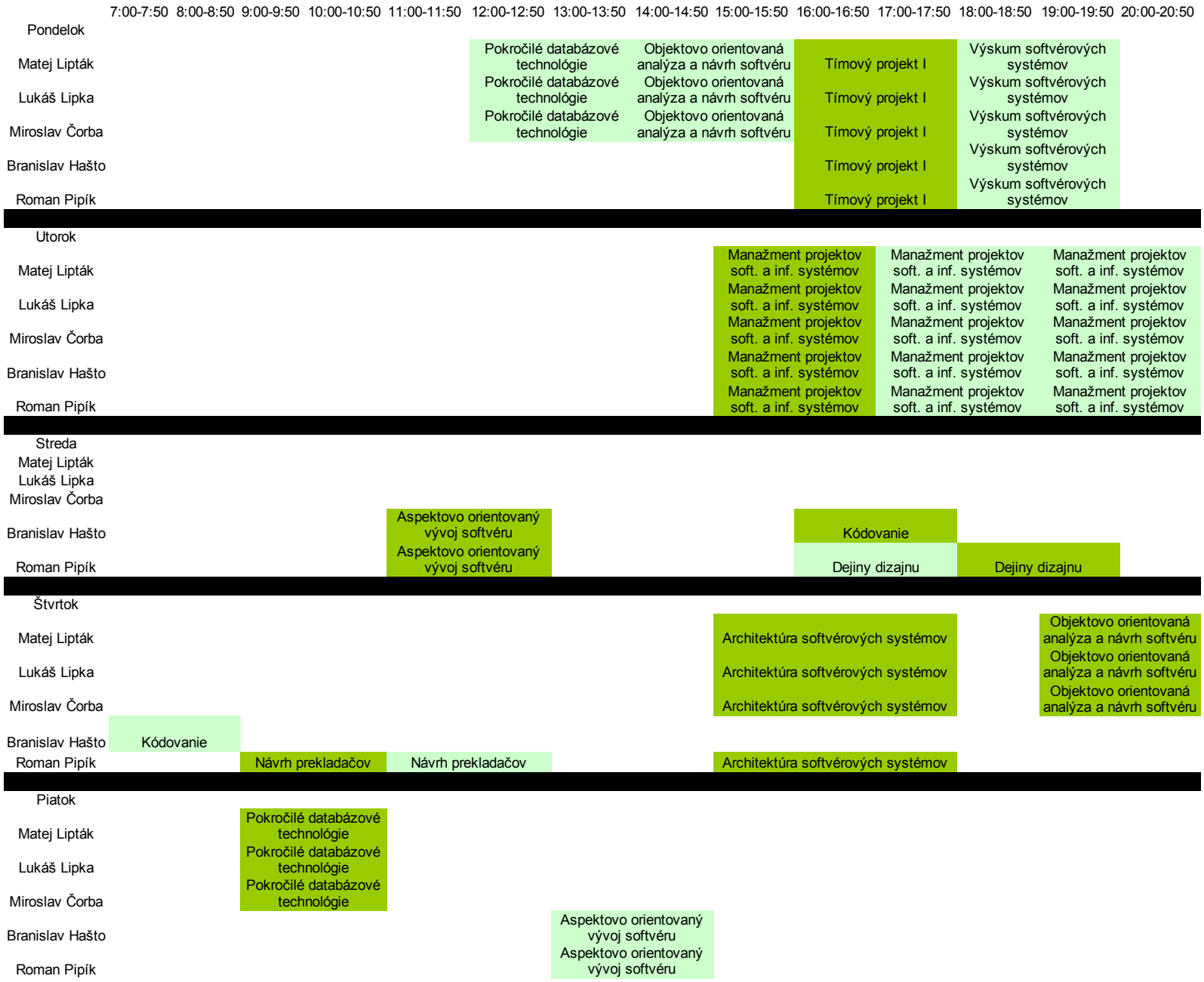
Po nadviazaní spojenia bude ďalšia komunikácia prebiehať bezdrátovým pripojením pomocou protokolu TCP/IP cez server. Aplikácie potrebné k rôznym transakciám budú spúšťané pomocou "cloud computingu", konkrétne Google App Engine. Táto "cloud" služba podporuje viacero programovacích jazykov, medzi nimi aj jazyk Java, ktorý použijeme na implementáciu.

Keďže pôjde o citlivé transakcie týkajúce sa bankových služieb, dôraz budeme klásť aj na bezpečnosť vytváranej aplikácie.

# Príloha A - Poradie tém

1. Objektové úložisko dát
2. Crowdsourcing verejných dát
3. Platforma pre realizovanie transakcií prostredníctvom mobilných zariadení
4. Simulated Car Racing Competition 2011
5. Vyhľadávanie a sprístupnenie citácií
6. Tréner mentálnych schopností
7. 3D grafická podpora vyhľadávania znalostí v dokumentoch
8. Evolučný simulátor umelého života založený na heuristických pravidlách
9. Model používateľa pre jeho identifikáciu
10. Dizajn s použitím obohatenej reality
11. Prispôsobiteľný Widget
12. Portál pre časopis
13. Správa študentských projektov na fakulte
14. Tvorba rozvrhov
15. Interaktívna vizualizácia grafových štruktúr v 3D priestore
16. Virtuálna FIIT
17. RoboCup  tretí rozmer
18. Imagine Cup 2011: Game Design
19. Adaptívny proxy server

# Príloha B – Rozvrh tímu


7:00-7:50
8:00-8:50
9:00-9:50
10:00-10:50
11:00-11:50
12:00-12:50
13:00-13:50
14:00-14:50
15:00-15:50
16:00-16:50
17:00-17:50
18:00-18:50
19:00-19:50
20:00-20:50
Pondelok
Matej Lipták
Lukáš Lipka
Miroslav Čorba
Branislav Hašto
Roman Pipík
Pokročilé databázové technológie
Objektovo orientovaná analýza a návrh softvéru
Tímový projekt I
Výskum softvérových systémov
Utorok
Manažment projektov soft. a inf. systémov
Streda
Aspektovo orientovaný vývoj softvéru
Kódovanie
Dejiny dizajnu
Štvrtok
Architektúra softvérových systémov
Objektovo orientovaná analýza a návrh softvéru
Kódovanie
Návrh prekladačov
Piatok

# Príloha B - Záznamy zo stretnutí

# *Zápis z 1. stretnutia tímu č. 8*

**Dátum:**

28.9.2010

**Miestnosť:**

Softvérové štúdio

**Prítomní:**

*Vedúci tímu:*

Ing. Michal Čerňanský, PhD

*Členovia tímu:*

Bc. Miroslav Čorba
Bc. Branislav Hašto
Bc. Lukáš Lipka
Bc. Matej Lipták
Bc. Roman Pipík

*Zapisovateľ:*

Bc. Miroslav Čorba

**Téma stretnutia:**

úvod, organizácia projektu, konzultácia témy, SCRUM

**Vyhodnotenie úloh z predchádzajúceho stretnutia:**

-

**Opis stretnutia:**

1. Vedúci tímu otvoril stretnutie
2. Vedúci tímu predstavil metodiku SCRUM
3. Tím sa dohodol na používaných technológiách
4. Vedúci tímu predstavil organizáciu projektu
5. Vedúci tímu predstavil svoju predstavu o produkte
6. Vedúci tímu predostrel architektúru systému
7. Členovia tímu určili úlohy do ďalšieho stretnutia

**Úlohy do ďalšieho stretnutia:**

| Číslo | Úloha | Zodpovedný | Dátum ukončenia |
|---|---|---|---|
| 1. | Vytvorenie plagátu a názvu tímu | Miroslav Čorba | 4.10.2010 |
| 2. | Vytvorenie základnej štruktúry web prezentácie tímu | Branislav Hašto | 24.10.2010 |
| 3. | Vytvorenie zápisu zo stretnutia | Miroslav Čorba | 5.10.2010 |
| 4. | Zabezpečenie serverových služieb (SVN, TRAC) | Matej Lipták | 12.10.2010 |

**Zapisovateľ na ďalšom stretnutí:**
Bc. Branislav Hašto

**Poznámky:**
-

Vypracoval: _____ Overil: _____
Miroslav Čorba Matej Lipták

V Bratislave, 4.10.2010

# *Zápis z 2. stretnutia tímu č. 8*

**Dátum:**

5.10.2010

**Miestnosť:**

Softvérové štúdio

**Prítomní:**

*Vedúci tímu:*

Ing. Michal Čerňanský, PhD

*Členovia tímu:*

Bc. Miroslav Čorba
Bc. Branislav Hašto
Bc. Lukáš Lipka
Bc. Matej Lipták
Bc. Roman Pipík

*Zapisovateľ:*

Bc. Branislav Hašto

**Téma stretnutia:**

smerovanie projektu, TP Cup, podporné nástroje, organizácia tímu, priebeh vývoja

**Vyhodnotenie úloh z predchádzajúceho stretnutia:**

8. Plagát a názov tímu boli vytvorené.
3. Zápis zo stretnutia bol vytvorený.

Ostatné úlohy ešte nie sú dokončené, ale ich termín realizácie je až po tomto stretnutí.

**Opis stretnutia:**

1. Vedúci tímu zdôraznil potrebu rýchleho rozbehania podporných nástrojov pre vývoj. Dohodli sme sa na štruktúre SVN repozitára.
2. Prebrali sme stav internetovej prezentáciu tímu a určili termín dokončenia základnej kostry prezentácie do ďalšieho stretnutia.
3. Preberali sme smerovanie aplikácie – či sa budeme zameriavať skôr na simuláciu bankových prevodov, alebo na platformu pre realizovanie transakcií.
4. Preberali sme možnú účasť tímu na súťaži TP Cup. Predbežne sme súhlasili s účasťou. V dôsledku toho sa objavila potreba pridať do aplikácie nejakú pútavú vlastnosť, ktorá by mohla v súťaži zaujať.
5. Dohodli sme sa na rozdelení vývojárov pre klientskú a serverovú časť. Zároveň sa toto rozdelenie môže ešte zmeniť, keď sa definitívne dohodneme na smerovaní aplikácie (bod 3).
6. Vedúci tímu určil, že treba spraviť prieskum trhu a zistiť, aké podobné aplikácie už existujú.
7. Vedúci tímu predviedol vývojové prostredie XCode a opísal základné prvky používateľskeho rozhrania pre iPhone na príkladoch.
8. Vedúci tímu odporučil dokumenty na preštudovanie (Human Interface Guidelines, Objective C, Cocoa Touch, Vies programming guide).
9. Diskutovali sme, ako budeme produkt testovať – po prvom týždni šprintu sa dohodneme, či použijeme aj TDD.
10. Vedúci tímu predbežne určil náplň práce v prvom a druhom týždni šprintu:

- Prvý týždeň bude určený na oboznámenie sa s vývojárskymi platformami.
- V druhom týždni by sme mali implementovať jednoduchú HTTP komunikáciu. Tiež by sme mali skúsiť nájsť vhodné knižnice t.j. knižnice na generovanie a rozpoznávanie 2D kódov pre iPhone; knižnice na implementáciu PKI a certifikátov na serveri.

11. Vedúci tímu určil programátorské úlohy pre každého člena tímu – jednoduché Hello World aplikácie – každý na platforme, ktorú ma pridelenú.
12. Vedúci tímu navrhol REST architektúru, HTTPS ako komunikačný protokol a JSON ako prenosový formát.
13. Diskutovali sme o rozdelení uloh v tíme – je potrebné určiť role členov. Zároveň sa treba dohodnúť, ako sa prispôsobiť špecifickej situácii tímu – 2 do veľkej miery nezávislé vývojárske podtímy.
14. Vedúci tímu zdôraznil potrebu projektového denníka.

**Úlohy do ďalšieho stretnutia:**

| Číslo | Úloha | Zodpovedný(í) | Termín dokončenia |
|---|---|---|---|
| 1. | Spojazdnenie SVN + návod pre tím | Branislav Hašto | 12.10.2010 |
| 2. | Spojazdnenie Trac + Agilo | Matej Lipták | 12.10.2010 |
| 3. | Vytvorenie kostry webu | Branislav Hašto | 12.10.2010 |
| 4. | Vytvorenie šablóny pre zápisy zo stretnutí | Miroslav Čorba | 8.10.2010 |
| 5. | Vytvorenie zápisu zo stretnutia | Branislav Hašto | 12.10.2010 |
| 6. | Návrh na položky zo zápisu vhodné do backlogu | Branislav Hašto | 12.10.2010 |
| 7. | Prieskum trhu, hľadanie podobných produktov | Roman Pipík | 12.10.2010 |
| 8. | Prezrieť dokumenty k tvorbe používateľských rozhraní pre iPhone (viď bod 8 vyššie) | Lukáš Lipka, Matej Lipták | 12.10.2010 |
| 9. | Prezrieť dokumentáciu ku Google App Engine | Miroslav Čorba, Branislav Hašto, Roman Pipík | 12.10.2010 |
| 10. | Hello world aplikácia pre iPhone | Lukáš Lipka, Matej Lipták | 12.10.2010 |
| 11. | Hello world aplikácia pre Google App Engine | Miroslav Čorba, Branislav Hašto, Roman Pipík | 12.10.2010 |
| 12. | Dohodnúť sa na rolách a rozdelení úloh v tíme | Všetci | 12.10.2010 |
| 13. | Dohodnúť sa na smerovaní aplikácie | Všetci | 12.10.2010 |
| 14. | Zamyslieť sa nad architektúrou aplikácie | Všetci | 12.10.2010 |
| 15. | Zamyslieť sa nad vlastnosťami vhodnými pre TP Cup | Všetci | 12.10.2010 |

**Zapisovateľ na ďalšom stretnutí:**
Bc. Lukáš Lipka

Vypracoval: _____ Overil: _____
Branislav Hašto Miroslav Čorba

V Bratislave, 7.10.2010

# *Zápis z 3. stretnutia tímu č. 8*

**Dátum:**

12.10.2010

**Miestnosť:**

Softvérové štúdio

**Prítomní:**

*Vedúci tímu:*

Ing. Michal Čerňanský, PhD

*Členovia tímu:*

Bc. Miroslav Čorba
Bc. Branislav Hašto
Bc. Lukáš Lipka
Bc. Matej Lipták
Bc. Roman Pipík

*Zapisovateľ:*

Bc. Lukáš Lipka

**Téma stretnutia:**

Zhodnotenie týždňa, plánovanie úloh

**Vyhodnotenie úloh z predchádzajúceho stretnutia:**

- Trac/Agilo/SVN + návod – čiastočne splnené, chýbajú prístupové účty, presun na iný stroj
- Webová stánka – čiastočne hotová, problémy s kódovaním textu
- Podobné produkty – Cimbal (v Apple App Store)
- Hello World pre iPhone – čiastočne hotové
- Hello Google App Engine – čiastočne hotové

**Opis stretnutia:**

1. Cimbal – prehľad existujúcej aplikácie
2. Zhodnotenie úloh z predchádzajúceho týždňa, zošity
3. Rozdelenie úloh na budúci týždeň
4. Diskusia o možných nápadoch

**Úlohy do ďalšieho stretnutia:**

| Číslo | Úloha | Zodpovedný | Dátum ukončenia |
|---|---|---|---|
| 1. | Funkčné a prístupné Trac/Agilo/SVN + návod na pripojenie | Lipták | čoskoro |
| 2. | Webová stránka | Hašto | |
| 3. | Prototyp mobilnej aplikácie | Lipka & Lipták | 19.10.2010 |
| 4. | Prototyp GWE | Pipík & Čorba | 19.10.2010 |
| 5. | Návrh interface pre komunikáciu medzi klientom a serverom | Lipka | 19.10.2010 |
| 6. | Komunikácia, knižnice, certifikáty | erárne | 19.10.2010 |
| 7. | Zamyslieť sa nad TP cupom + smerovanie projektu | erárne | 19.10.2010 |
| 8. | Architektúra (použité technológie) | erárne | koniec 2. šprintu |

**Zapisovateľ na ďalšom stretnutí:**

Bc. Matej Lipták

**Poznámky:**

- korektné vymyslenie rozhraní a mechanizmov pre platenie
- napísať sumu, namiesto potvrdzovania
- nespoliehať sa na kameru na zariadení
- možnosť zobraziť štatistiky z transakcií (napr. webová aplikácia)
- urýchliť platenie cez internet banking

Vypracoval: _____ Overil: _____

Bc. Lukáš Lipka Bc. Branislav Hašto

V Bratislave, 12.10.2010

# *Zápis zo 4. stretnutia tímu č. 8*

**Dátum:**

19.10.2010

**Miestnosť:**

Softvérové štúdio

**Prítomní:**

*Vedúci tímu:*

Ing. Michal Čerňanský, PhD

*Členovia tímu:*

Bc. Miroslav Čorba
Bc. Branislav Hašto
Bc. Lukáš Lipka
Bc. Matej Lipták
Bc. Roman Pipík

*Zapisovateľ:*

Bc. Matej Lipták

**Téma stretnutia:**

Zhodnotenie predchádzajúceho šprintu, určenie úloh, naplánovanie ďalšieho šprintu

**Vyhodnotenie úloh z predchádzajúceho stretnutia:**

Funkčné a prístupné Trac/Agilo/SVN + návod na pripojenie - splnené
Webová stánka – skoro hotová, treba pridať obrázok

Prototyp mobilnej aplikácie - splnené
Prototyp GWE - splnené
Návrh interface pre komunikáciu medzi klientom a serverom – splnené
Zamyslieť sa nad TP cupom + smerovanie projektu – stále v procese zamýšľania
Architektúra (použité technológie) – nevyriešené

**Opis stretnutia:**

- Vedúci tímu skontroloval projektové denníky.
- Vyhodnotili sme výsledok špritnu.
- Povedali sme si naše skúsenosti zo šprintu.
- Vedúci určil nejaké user story: používateľ dokáže zobraziť 2D kód – keď na iPhone zadáme dáta, ktoré chceme odoslať na server, tak okrem ID sa zobrazí 2D kód [bodové ohodnotenie: 13]
- Vedúci určil ďalšiu user story: dokáže prečítať 2D kód – nenačíta sa ID, ale ID sa získa z načítaného 2D kódu. To by sa mohlo rozdeliť na dve časti – 1) obrázok sa načíta zo súboru 2) kód sa zosníma kamerou. [Dokopy bodové ohodnotenie: 40]
- Dohodli sme sa, že používateľ bude mať na transakčnom systéme konto, aby mohol byť autentifikovaný pri úlohách, ktoré vyžadujú bezpečnosť. Okrem toho bude mať možnosť zostať neprihlásený pri úlohách, ktoré bezpečnosť nevyžadujú. Toto bude ako requirement – manažment používateľa. Časť by bola anonymné transakcie a časť transakcie, kde bude prihlásený. Môže to byť ako iPhone aplikácia, standalone, alebo web aplikácia.

- Dohodli sme sa, že zatiaľ bude iba jednoduchá registrácia (meno, heslo). Registráciu cez nejaké certifikáty doriešime neskôr.
- Vedúci určil významnú úlohu: určiť rozhranie a scenáre rozhrania.
- Vedúci určil ďalší requirement: aby si používateľ mohol pozrieť transakcie. Aby sa mohol pozerať čo kedy urobil, aby potom mohol veci reklamovať, atď.
- Braňo povedal svoj nápad ohľadom bezpečnosti, že všetky dáta by mohli byť zašifrované na serveri. Vedúci zhodnotil, že by to bola užitočná súčasť systému, ktorá by ho robila zložitejším.
- Ďalšia možnosť na spôsob platby bola taká, že v obchode by bol napríklad pri pokladni Bar kód a rovno pri tom by sme mohli platiť.
- Vedúci určil user stories z manažovania používateľov:
- možnosť registrovania používateľa [bodové ohodnotenie: 13]
- Vedúci určil user stories z requirementu prezerania transakčného systému:
- možnosť prihásenia do systému [bodové ohodnotenie: 5]
- možnosť vylistovania transakcií
- Braňo navrhol ďalšiu user story – z iPhonu pošleme viacero informácií [bodové ohodnotenie: 8]
- Obodovali sme jednotlivé user stories. (pridané do hranatých zátvoriek za nich)
- Dohodli sme user stories na najbližší šprint:
- používateľ dokáže zobraziť 2D kód (client)
- možnosť registrovania používateľa (server)
- posielanie viacero informácií (client + server)
- Vedúci pripomenul dôležitosť testovania a ich zapojenia do user stories

**Úlohy do ďalšieho stretnutia:**

| Číslo | Úloha | Zodpovedný | Dátum ukončenia |
|---|---|---|---|
| 1. | Štábna kultúra | všetci | |
| 2. | Webová stránka | Hašto | 22.10.2010 |
| 3. | Zobrazenie 2D kódu | Lipka, Lipták | 2.11.2010 |
| 4. | Nájsť knižnicu, ktorá zobrazuje, aj číta 2D kód | Lipka, Lipták | 2.11.2010 |
| 5. | Posielanie viacero informácií (klientská časť) | Lipka, Lipták | 2.11.2010 |
| 6. | Rozšíriť súčasné REST API o možnosť registrovania používateľov (JSON objekty) | Hašto | 2.11.2010 |
| 8. | Ukladanie používateľov | Pipík | 2.11.2010 |
| 9 | Rozhranie, stránka | Čorba | 2.11.2010 |
| 10 | Architektúra (použité technológie) | všetci | koniec 2. šprintu |

**Zapisovateľ na ďalšom stretnutí:**
Bc. Roman Pipík

Vypracoval: _____ Overil: _____
Bc. Matej Lipták Bc. Lukáš Lipka

V Bratislave, 19.10.2010

# *Zápis z 5. stretnutia tímu č. 8*

**Dátum:**
26.10.2010
**Miestnosť:**
Softvérové štúdio
**Prítomní:**
*Vedúci tímu:*
Ing. Michal Čerňanský, PhD
*Členovia tímu:*
Bc. Miroslav Čorba
Bc. Branislav Hašto
Bc. Lukáš Lipka
Bc. Matej Lipták
Bc. Roman Pipík
*Zapisovateľ:*
Bc. Roman Pipík

**Téma stretnutia:**
Zhodnotenie priebehu šprintu, zhodnotenie existujúcich úloh

**Vyhodnotenie úloh z predchádzajúceho stretnutia:**
- Naplnenie programu Trac/Agilo úlohami – splnené
- Zamyslieť sa nad „TPcup“ + smerovanie projektu – stále v procese riešenia
- Webová stránka – splnené
- Dokument s použitými technológiami – splnené

**Opis stretnutia:**
1. Vedúci tímu kontroloval priebeh úloh. Celkovo bola splnená iba úloha „Použité technológie“, ostatné úlohy boli čiastočne splnené, alebo blokované.
2. Bolo zistené že úsilie v druhej polovici šprintu bude potrebné zvýšiť.
3. Vedúci tímu zhodnotil úlohy v šprinte. V šprinte chýbajú úlohy testovania, ktoré bolo potrebné odčleniť od samotnej funkčnej úlohy. Ďalej bolo potrebné úlohy viac rozčleniť, aby bolo možné lepšie kontrolovať priebeh šprintu.
4. Vedúci riešil vzdialený prístup na počítač Apple v softvérovom štúdiu.

**Zapisovateľ na ďalšom stretnutí:**
Bc. Miroslav Čorba

Vypracoval: _____ Overil: _____
Bc. Roman Pipík Bc. Matej Lipták

# Príloha C - Preberacie protokoly