EKONOMICKÁ UNIVERZITA V BRATISLAVE

Fakulta hospodárskej informatiky

Evidenčné číslo: 103004/D/2016/2268386125

Analýza, návrh a implementácia mobilnej aplikácie s backendovým rozhraním zameranej na gamifikáciu

Diplomová práca

2016 **Bc. Ľubomír Šima**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

Fakulta hospodárskej informatiky

Analýza, návrh a implementácia mobilnej aplikácie s backendovým rozhraním zameranej na gamifikáciu

Diplomová práca

Študijný program:	Informačný manažment
Študijný odbor:	Kvantitatívne metódy v ekonómii
Školiace pracovisko:	Katedra aplikovanej informatiky FHI
Vedúci záverečnej práce:	Ing. Ján Pittner, PhD.

Miesto a rok predloženia práce

Bratislava 2016

Meno a priezvisko

Bc. Ľubomír Šima

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: 18. apríla 2016

.....

(podpis študenta)

ABSTRAKT

Bc. Šima, Ľubomír: *Analýza, návrh a implementácia mobilnej aplikácie s backendovým rozhraním zameranej na gamifikáciu* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky – Vedúci záverečnej práce : Ing., Ján Pittner, PhD. – Bratislava: FHI EU, 2016, 59 s.

Cieľom diplomovej práce je vytvorenie systému pozostávajúceho z aplikácie pre operačný systém Android a jej webového rozhrania, ktorý bude využívať prvky gamifikácie. Práca je rozdelená do 3 kapitol. Obsahuje 48 obrázkov a 2 prílohy. V prvej kapitole sú vysvetlené princípy gamifikácie, oblasti jej využitia a najčastejšie používané prvky. Prvá kapitola tiež pojednáva o komponentoch pomocou ktorých bolo možné vytvoriť aplikáciu a jej webové rozhranie, počnúc základnými črtami operačného systému Android, cez ilustráciu životného cyklu aktivít v tomto OS a taktiež popis komponentov potrebných na vytvorenie webového rozhrania a popis jeho životného cyklu.

V ďalšej časti je v krátkosti opísaný dôvod vzniku systému a možnosť jeho využitia v praxi. Sú tu tiež popísané komponenty potrebné na úspešnú implementáciu systému.

Záverečná kapitola sa zaoberá výsledkami práce, popisom návrhu a implementácie jednotlivých komponentov systému. Sú v nej popísané funkcionality na jednotlivých úrovniach systému, fungovanie serverovej a klientskej časti systému. Táto kapitola taktiež obsahuje informácie o využívaných prvkov gamifikácie.

Výsledkom riešenia tejto problematiky je systém pozostávajúci z mobilnej aplikácie pre operačný systém Android, slúžiaci na evidenciu dochádzky študentov hravou formou a k nej prislúchajúce webové rozhranie, ktoré rozširuje možnosti aplikácie a poskytuje priestor pre prístup k systému bez potreby vlastniť zariadenie s OS Android.

Kľúčové slová: Android, mobilná aplikácia, gamifikácia, GPS, overenie polohy, webové rozhranie, Java webstránka, hravá evidencia dochádzky

ABSTRACT

Bc. Šima, Ľubomír: *Analysis, design and implementation of mobile application with backend interface utilising gamification* – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics – Supervisor : Ing. Ján Pittner, PhD. – Bratislava: FHI EU, 2016, 59 p.

The aim of this thesis is to create system consisting of Android OS application and its web interface utilizing gamification elements. The thesis is divided into 3 chapters. It contains 48 figures and 2 annexes. The first chapter explains gamification principles, areas of its utilisation and its most frequently used elements. It also describes elements that were used to create the application and its web interface; starting from basic features of Android OS, through presentation of this OS activity life cycle as well as description of components required for web interface implementation and description of its life cycle.

The next part briefly describes the reason to the system creation and its practical usage. It also describes components needed for successful system implementation.

The closing chapter deals with the thesis results, description of design and implementation of individual system components. It describes features of each system level as well as server/client-side system operation. This chapter also contains information on commonly used gamification elements.

This problem solving resulted in the system consisting of mobile application for Android OS used as a playful way of recording the student attendance, as well as corresponding web interface which extends the possibilities of its usage. It also allows access to the system without owning Android OS device.

Key words: Android, mobile application, gamification, GPS, localization, web interface, Java web page, playful attendance records

Obsah

Zoznam skratiek a značiek

Skratka	Anglický názov	Slovenský názov
HTML	HyperText Transfer Protocol	Hypertextový značkový jazyk
DVM	Dalvik Virtual Machine	Dalvikov virtuálny stroj
JVM	Java Virtual Machine	Java virtuálny stroj
SDK	Software Development Kit	Súbor nástrojov, pre vývoj softvéru
GPS	General Positioning System	Globálny lokalizačný systém
JSON	JavaScript Object Notation	Javascriptový objektový zápis
ORM	Object-relational mapping	Vedný odbor a progr. technika na konverziu dát medzi nekompatibilnými systémami pomocou techník OOP
API	Application programming interface	Rozhranie pre programovanie aplikácií
JPA	Java Persistence API	Rozhranie popisujúce manažment dát v Java aplikáciách
POJO	Plain old Java object	Skratka pre bežný objekt v jazyku Java
HQL	The Hibernate Query Language	Objektovo orientovaný jazyk podobný jazyku SQL
XML	eXtensible Markup Language	Rozšíriteľný značkovací jazyk, zovšeobecnenie jazyka HTML
Md5	Message digest	Funkcia na šifrovanie

Úvod

Trh s mobilnými a inteligentnými zariadeniami sa medzičasom stal jedným z najrýchlejšie sa rozvíjajúcich odvetví. Mobilné a inteligentné zariadenia a v nich implementovaný špeciálne vytvorený operačný systém vytvorili priestor pre vývojárov na vytváranie natívnych aplikácií pre tieto zariadenia, ktoré do značnej miery rozširujú ich funkcionalitu. S narastajúcim počtom a škálou zariadení sa zvyšuje aj množstvo aplikácií pre tieto zariadenia. Napriek faktu, že sa aplikácie pre mobilné a inteligentné zariadenia vyvíjajú pomocou rovnakých programovacích jazykov ako bežné stolové aplikácie, ich štruktúra a vývoj je pomerne špecifický. Je potrebné brať do úvahy variabilitu rozmerov obrazoviek mobilných aplikácií, ich hardvérový výkon, verziu operačného systému a tiež aj platformu, pre ktorú bude aplikácia vyvíjaná. Pri vývoji aplikácií pre inteligentné zariadenia ako napr. náramky, žiarovky či autá, treba brať do úvahy hardvérové vybavenie a rozsah možností ktoré toto vybavenie ponúka a tiež životný cyklus zariadenia.

Hlavným dôvodom voľby prvkov systému – aplikácie pre Android a webového rozhrania bol podiel zastúpenia a používania týchto prvkov používateľmi. V dnešnej dobe je takmer každá služba, či produkt zastrešená web stránkou, ktorá obsahuje potrebné informácie, alebo ponúka istú funkcionalitu pre využívanie produktov a služieb. Ako žiadané sa pre poskytovanie niektorých služieb môže javiť sprístupnenie poskytovaného rozhrania pre ďalšie platformy – napríklad pre mobilné zariadenia. Je možné predpokladať, že systém pozostávajúci z Android aplikácie a webového rozhrania bude príťažlivejší pre širší okruh používateľov ako systémy, ktoré sú dostupné len prostredníctvom jednej platformy.

1. Súčasný stav riešenej problematiky doma a v zahraničí

1.1 Gamifikácia

1.1.1 Čo to je gamifikácia

Veľmi všeobecné a zjednodušené vysvetlenie pojmu gamifikácia je implementovanie herných techník a prvkov do neherného prostredia[3]. Gamifikácia má za účel hravou formou povzbudiť v používateľovi chuť používať konkrétnu aplikáciu, či riešiť zadanú úlohu.[1]

Človek je tvor hravý a preto je vhodné poskytnúť mu v každodennom živote možnosť popri nudných a rutinných povinnostiach sa aj trocha zabaviť, či zasúťažiť si práve pomocou vykonávania neherných činností hravou formou. Takto je možné taktiež podporiť kreatívne myslenie a zápal pre aplikáciu. Pridaním herných prvkov môžeme docieliť nielen väčšej obľúbenosti u už existujúcich používateľov, ale taktiež je možné získať mnoho ďalších. Spojením s príjemným používateľským prostredím sa stane gamifikovaná aplikácia neodolateľným lákadlom pre každého používateľa, predsa je zaujímavé vyhrať napr. aj pri bežnom otváraní e-mailu.[1]

1.1.2 Gamifikácia ako novodobý trend

Pojem gamifikácia ako prvý použil Charles Coonradt pre názov svojej knihy o herných technikách s názvom Grandfather of Gamification[3]. Trend vkladať herné prvky do neherného prostredia sa začal tešiť obľube v roku 2010, keď ho vedúci pracovníci zaviedli do širšej firemnej praxe. Medzi najčastejšie okruhy využitia môžeme zaradiť: motivovanie používateľa aby intenzívnejšie využíval aplikáciu, motivácia pre podávanie lepších fyzických výkonov, vrátenie investícií, vyššiu kvalitu vstupných dát, dochvíľnosť či výukové procesy[3] [1].

Podľa prieskumu z januára roku 2014 má implementovanie herných prvkov vo všetkých skúmaných sférach pozitívny účinok[4] [1].

1.1.3 Využívané prvky

Koncept gamifikácie využívamnoho prvkov k motivácii používateľa, medzi hlavné prvky patria:

- Ocenenia/odznaky: Ocenenia sú virtuálnou odmenou za uskutočnenie nejakej činnosti, alebo za dosiahnutie určitého cieľa. Môžu byť zadelené do viacerých kategórií podľa dosiahnutého cieľa. Zverejnenie dosiahnutého ocenenia môže byť nepriamou formou pochválenia sa ostatným používateľom.
- Úrovne: Používateľom prideľujeme úrovne podľa dosiahnutého skóre v určitej kategórii, alebo za všetko spolu. Aj keď je úroveň len akési virtuálna reprezentácia skóre, čím vyššiu úroveň používateľ dosiahne, tým lepši pocit získa.
- Rebríčky: rebríčky zachytávajú poradie používateľov v hodnotení podľa rôznych kritérií.
- Ukazovatele postupov: Zobrazujú zvyčajne percento dosiahnutia určitého cieľa, môže sa jednať napr. o rozsah vyplnenia informácii v profile.
- Oznámenia aktivít: Ostatní používatelia môžu v reálnom čase vidieť činnosť iného používateľa, prípadne získané ocenenia.
- Obrázky: Používateľ si môže pre svoj účet priradiť osobný obrázok s cieľom personalizovať si aplikáciu.
- Odozva v reálnom čase: Informuje používateľa o získaní nejakého ocenenia, prípadne dosiahnutia novej úrovne ihneď po jeho získaní.
- Virtuálna mena: Slúži ako spôsob ocenenia používateľa za vykonanie istej činnosti. Najmä pri rutinných činnostiach je vhodné, aby používateľ za ich vykonanie niečo dostal.
- Oceňovanie: Podobne ako pri virtuálnej mene, používateľ je nejakým spôsobom ocenený za svoju aktivitu, či už je to virtuálna medaila, alebo symbolická vlajka.
- Výzvy a úlohy: Implementovaním výziev je používateľ nabádaný k vykonaniu akejsi série aktivít, či zdolaniu zadanej úlohy.

1.1.4 Príklady využitia v praxi

V auguste roku 2009 dala Švajčiarska spoločnosť Gbanga do obehu náučnú aplikáciu, ktorá po virtuálnom kantóne Zürich rozmiestnila mnoho ohrozených zvierat a úlohou používateľov bolo tieto zvieratá zbierať. Zozbierané zvieratá mohli byť odovzdané buď priamo v ZOO, alebo na zberných miestach, resp. plagátoch patriacich ku aplikácii kde za každé odovzdané zviera dostal používateľ body a virtuálne osivo, ktorým mohol zalesniť ohrozené územia.[7] [1]

Vo februári roku 2011 vznikla on-line hra a sociálne sieť s názvom Fitocracy, ktorá za pomoci gamifikácie nabádala užívateľov k tomu, aby zlepšovali svoju fyzickú kondíciu. V hre je možné si vybrať disciplínu, v ktorej môže používateľ získavať body a zlepšovať sa. Do hry boli začlenené úlohy, úrovne a odznaky a tieto úspechy bolo možné zdieľať na sociálnej sieti s ostatnými užívateľmi.[8] [1]

Z domáceho prostredia uvedieme do popredia internetovú banku mBank, ktorá 10. februára 2014 spustila nové internetové bankovníctvo, kde je už na prvý pohľad vidno, že o gamifikáciu núdza nebude. Za používanie získava používateľ automaticky odznaky a ocenenia podľa toho, do akej miery a koľko služieb využíva. Odznaky je možné získať napr. za počet bezkontaktných platieb uskutočnených od zavedenia nového IB, tiež za pravidelné prihlasovanie do systému, alebo za počet uskutočnených prevodov. [1]

1.2 Operačný systém Android

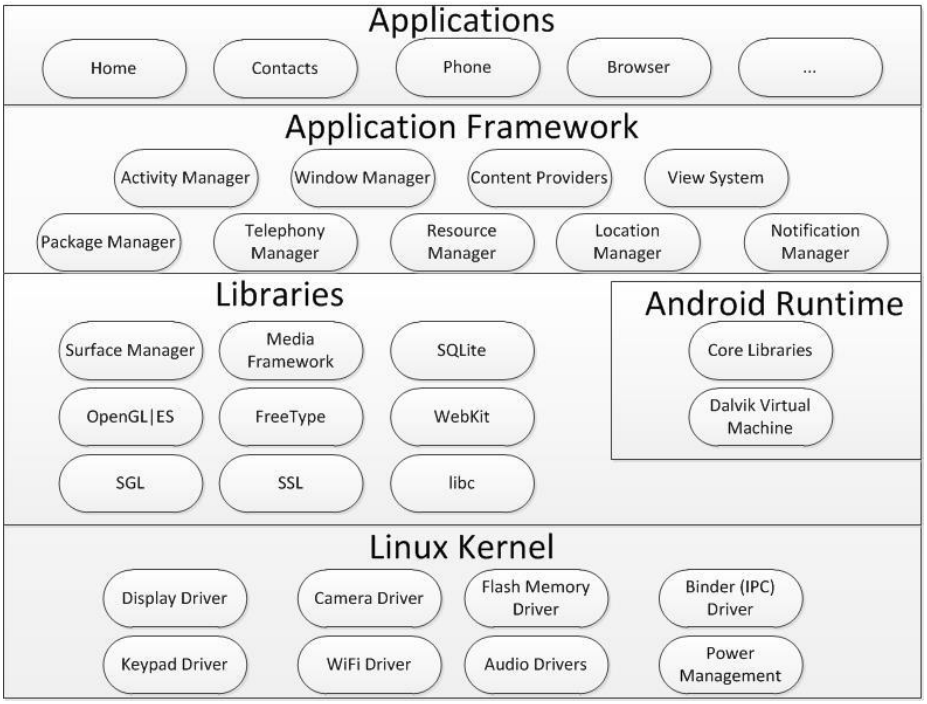
1.2.1 Popis systému

Android je operačný systém postavený na linuxovom jadre, vytvorený najmä pre dotykové mobilné zariadenia avšak je možné ho nájsť v rôznych inteligentných zariadeniach ako sú televízory, autá alebo hodinky. Operačný systém Android bol pôvodne vyvíjaný ako systém pre digitálne kamery, neskôr sa zakladatelia spoločnosti Android Inc. Andy Rubin, Rich Miner a Nick Sears rozhodli ho preorientovať na využitie v mobilných zariadeniach. V roku 2005 spoločnosť odkúpil Google s cieľom vstúpiť na trh s mobilnými zariadeniami. Na vývoji a zdokonaľovaní tejto platformy sa v značnej miere podieľa spoločnosť Google a aliancia Open Handset Alliiance, konzorcium spoločností z rôznych odvetví vo sfére informačných technológií[6]. Prvým

dotykovým zariadením využívajúcim operačný systém Android bol mobilný telefón z rady Dream od značky HTC vydaným v októbri roku 2008[6]. Pri vývoji operačného systému boli brané do úvahy hardvérové obmedzenia, ktorými mobilné zariadenia disponujú. Medzi najvýraznejšie obmedzenia patrí výdrž batérie, obmedzené množstvo operačnej pamäte a taktiež obmedzená výpočtová kapacity procesora. Jadro systému Android bolo navrhnuté pre prácu so širokou škálou zariadení a preto je Android distribuovateľný na akékoľvek mobilné zariadenia, bez ohľadu na hardvérové vybavenie, veľkosť či rozlíšenie obrazovky. Užívateľské prostredie operačného systému Android je navrhnuté tak, aby s ním používateľ dokázal komunikovať prostredníctvom akejkoľvek akcie vykonateľnej na konkrétnom zariadení, či už je to dotyk krátky, alebo dlhý, posúvanie či otáčanie, skrátka všetko, čo dokáže dotyková obrazovka zachytiť. Používateľ má možnosť ovládať zariadenie taktiež pomocou polohovania zariadenia, či prudkými pohybmi, ktoré sníma vstavaný gyroskop. Systém Android ponúka operačný systém s používateľským prostredím pre koncových používateľov a tiež aj kompletné riešenie nasadenia operačného systému (špecifikácia ovládačov a pod.) pre mobilných operátorov a výrobcov zariadení a napokon tiež efektívne nástroje pre vývoj aplikácií - Software Development Kit[5] [1].

Obrázok 1: Logo OS Android

1.2.3 Architektúra systému



Obrázok 2: Architektúra OS Android

Operačný systém Android je rozdelený do piatich častí, každá má svoj účel a nemusí byť priamo oddelená od ostatných vrstiev.

Jadro operačného systému (Linux Kernel)

Základ operačného systému tvorí linuxové jadro, ktoré je rôzne pre rozdielne verzie systému. Jadro použité v Android 6.0 je vo verzii 3.18.10. V zariadeniach s verziou Android nižšou ako 4.0 (Ice Cream Sandwich) bolo použité jadro verzie 2.6. Použité jadro bolo modifikované tak, aby čo možno najefektívnejšie využívalo zdroje mobilného zariadenia. Upravuje správu pamäte a spotrebu energie. Jadro sa stará o komunikáciu s hardvérom zariadenia a poskytuje služby vyššej vrstvy, ktoré vytvárajú abstrakciu nad samotným hardvérom zariadenia. [1]

Vrstva knižníc (Libraries)

V tejto vrstve sa nachádzajú natívne knižnice napísané v jazyku C a C++. Oproti jadru systému je táto vrstva viac prispôsobiteľná, pretože knižnice je možné ľubovoľne meniť podľa konkrétnych potrieb vývojárov. Tieto knižnice boli rovnako ako jadro vydané pod Open-source licenciami. Medzi základné knižnice patrí: [1]

- WebKit – knižnica umožňujúca zobrazovanie HTML obsahu.
- SQLite – knižnica poskytujúca databázové služby ostatným aplikáciám.
- FreeType: knižnica zaisťujúca vykresľovanie písmen.
- MediaFramework: táto knižnica zabezpečuje prácu s multimediálnym obsahom. Umožňuje obsah prehrávať a taktiež aj zaznamenávať.
- SGL: táto knižnica slúži na vykresľovanie 2D grafiky.
- OpenGL: táto knižnica je založená na OpenGL ES API, pomocou ktorej je možné v zariadení využívať 3D akceleráciu a vykresľovanie obrazu.
- Surface Manager: poskytuje prístup k displeju a zabezpečuje zobrazovanie obrazu.

Android runtime

Táto vrstva sa v hierarchii architektúry nachádza na úrovni vrstvy knižníc, plní však mierne odlišnú funkciu. Prvou časťou sú systémové knižnice odvodené z knižníc určených pre systém BSD napísané v jazyku C. Druhú časť tvorí DVM(Dalvik Virtual Machine) čo je vlastne JVM(Java Virtual Machine) upravený tak, aby zabezpečoval beh aplikácií v jazyku Java s pomocou Android SDK(Software Development Kit). DVM je oproti klasickému JVM odlišný v tom, že oproti zásobníkovej architektúre používa registre a veľkosť jeho pamäte je menšia. Pomocou nástroja dx sú .class súbory skompilované do bytekódu a uložené v .dex súbore namiesto známych .jar súborov. Dôvodom tejto zmeny je zvýšenie efektivity využívania obmedzených systémových zdrojov mobilného zariadenia. [1]

Application Framework

Táto vrstva poskytuje služby priamo bežiacim aplikáciám, jedná sa o prístup k službám poskytovaných operačným systémom. Zaisťuje taktiež beh aplikácií. Vrstva je rozdelená do modulov v napísaných v jazyku Java. Aplikácie komunikujú s modulmi pomocou správ. Medzi významné časti patrí: [1]

- Activity Manager: riadi životný cyklus aplikácií.
- Views: komponenty, ktoré slúžia pri vytváraní používateľského rozhrania, môžu to byť textové polia, obrázky, tlačidlá, zoznamy.
- Notification Manager: umožňuje aplikáciám zobrazovať upozornenia v stavovom riadku.

- Content Providers: Poskytujú úroveň abstrakcie pre akékoľvek dáta uložené v zariadení, ku ktorým je možné pristupovať z viacerých aplikácií.

Vrstva aplikácií (Applications)

Táto vrstva obsahuje samotné aplikácie, s ktorými prichádza bežný používateľ do styku pri využívaní zariadenia. Až na pár výnimiek sú tieto aplikácie napísané v jazyku Java, no je taktiež možné, aj keď sa to pri bežných aplikáciách neodporúča, programovať aplikácie v jazyku C/C++. Ak by sa malo jednať o aplikáciu, ktorá by nadmerne využívala procesor napr. zložitými výpočtami, vtedy je vhodné využiť na vytvorenie aplikácie práve tento jazyk. Aplikácie je možné vyhľadávať a inštalovať pomocou aplikácie Google play, kde je aspoň aký-taký predpoklad, že daná aplikácia nebude škodlivá a zámerne nepoškodí zariadenie, na ktoré bude nainštalovaná. Aplikácie si je možné nainštalovať aj pomocou inej služby podobnej Google play, býva však zvykom, že na obdobných stránkach bude enormné množstvo zobrazovanej reklamy. Vo všeobecnosti sa neodporúča inštalovať aplikácie bez overenia a navyše od neznámeho vydavateľa. [1]

1.2.4 Komponenty

Základné komponenty

Základnou a nevyhnutnou súčasťou každej aplikácie je AndroidManifest.xml, ktorý obsahuje deklarácie všetkých súčastí aplikácie a spôsob práce s nimi. Pri programovaní aplikácie pre OS Android budeme pracovať so štyrmi základnými objektmi: [1]

- Aktivity (Activity): Aktivity je možno označiť ako základný stavebný kameň užívateľského prostredia. Aktivitu je možno prirovnať ku oknu aplikácie, či stránke v prehliadači. Aktivity majú krátky životný cyklus a môžu byť kedykoľvek vypnuté. [1]
- Služby (Services): Služby sú navrhnuté tak, aby mohli bežať nezávisle na aktivite, môžu sa taktiež správať ako démoni v klasickom operačnom systéme. Po spustení služby používateľ nespozoruje žiaden priebeh ani informáciu o tom, v akom štádiu ktorá služba momentálne je, pretože bežia v pozadí. [1]
- Poskytovatelia obsahu (Content Provider): Poskytujú úroveň abstrakcie pre akékoľvek dáta uložené v zariadení, ku ktorým je možné pristupovať z viacerých

aplikácií. Dáta sú zapuzdrené a práve pomocou poskytovateľa obsahu je možné zachovať si kontrolu nad spôsobom prístupu k týmto dátam. Vo všeobecnosti je vhodné v zariadení narábať s dátami tak, aby k nim mali prístup aj iné aplikácie.

- Systémové správy (Intents): Systémové správy kolujú v zariadení a informujú aplikácie o rôznych zmenách ako napr. vloženie SD karty, prijatie SMS správy alebo spustenie aplikácie. Tieto správy je možné prirovnať ku udalostiam v klasickom operačnom systéme, na správu je možné reagovať, alebo si vytvoriť vlastnú správu napr. aby používateľovi oznámila nastanie vopred zadefinovanej situácie. [1]

Integrované komponenty

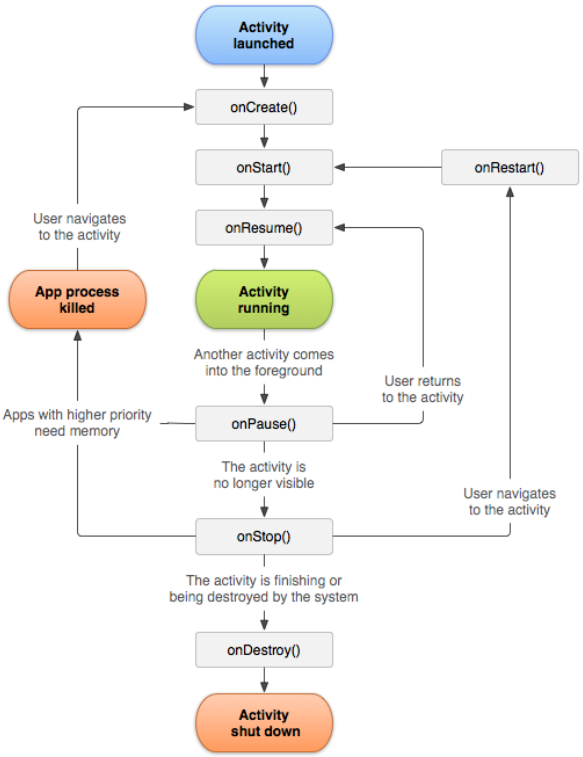
Pri vývoji aplikácií pre Android máme k dispozícii hneď niekoľko komponentov, s ktorými vieme pracovať:

- Úložisko: Aplikácie môžu využívať vstavané úložisko poskytované zariadením na ukladanie obsahu, z ktorého sú vytvorené, prípadne obsahu, ktorý používateľ pri interakcii s aplikáciou niekedy zadal a je žiadané ich ukladať. Je možné taktiež pristupovať a ukladať dáta na SD kartu, pokiaľ je v zariadení prítomná.
- Sieť: Inteligentné zariadenie by nebolo možné plnohodnotne využívať bez internetového pripojenia. Počas vývoja aplikácie je preto možné taktiež pracovať so sieťovými prvkami na akomkoľvek stupni.
- Multimédiá: Máme k dispozícii možnosť pracovať s multimediálnym obsahom.
- GPS: Pracovať môžeme hneď s dvoma typmi zisťovania polohy zariadenia, jedným je zistenie polohy prostredníctvom siete internet alebo modulu GPS a druhým je zistenie polohy pomocou siete GSM.

Telefón: Pri programovaní je možné tiež narábať so samotnými funkciami telefónu ako volanie, alebo prijímanie či odosielanie SMS správ. [1]

1.2.5 Životný cyklus aktivít

Zdrojový kód Android aplikácie v rámci svojej štruktúry neobsahuje funkciu main(). Operačný systém má vopred zadefinované stavy aplikácie, ktoré môžu nastať a ako sa bude aplikácia pri aplikovaní konkrétnej situácie správať.[1]



Obrázok 3: Životný cyklus aktivity

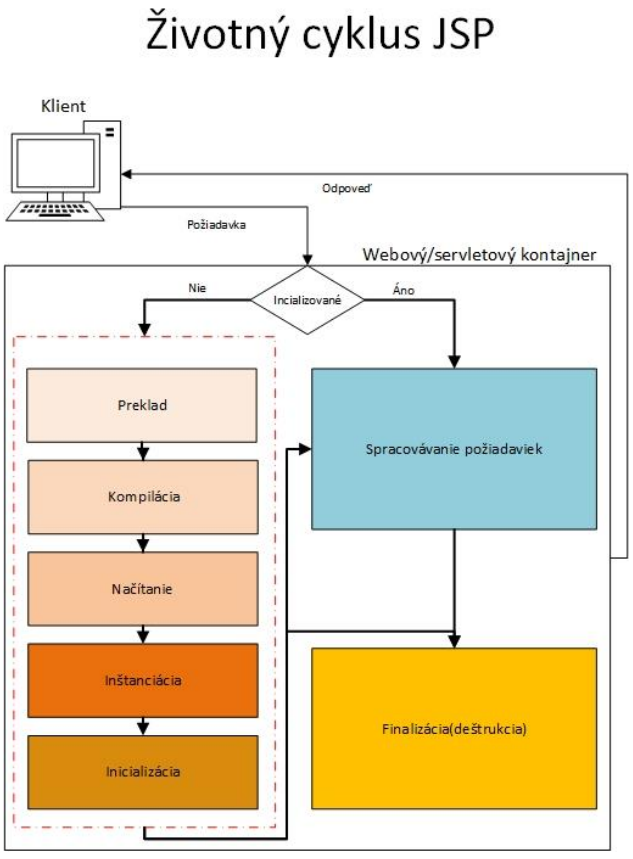
Po spustení Android aplikácie používateľom je vytvorený objekt typu Activity, mimo iné je vytvorený nový proces a volaná metóda onCreate(). V tejto funkcii sú obvykle vytvorené všetky objekty, ktoré sú nevyhnuté pre beh aktivity. Po vykonaní úvodnej metódy je volaná funkcia onStart(), ktorá vytvára prepojenie s komponentmi zobrazovanými v konkrétnom súbore *.xml, ktorý slúži ako obrazovka pre danú aktivitu. Pri behu aplikácie je v prípade potreby možné akúkoľvek aktivitu pozastaviť, kde by bola volaná metóda onPause(). Ak už aktivita nie je viditeľná, ani potrebná dá sa zastaviť zavolaním metódy onStop(). Zo stavu pozastavenia uvedieme aktivitu do popredia volaním metódy onResume() a po úplnom zastavení opätovným volaním metódy onCreate(). Pri programovaní Android aplikácie je nevyhnutné preto s dátami v aplikácii pracovať tak, aby sme v každom stave, ktorý môže nastať mali kontrolu nad všetkými dátami, ktoré sú pre aplikáciu dôležité a aby nedošlo k akejkoľvek nechcenej strate.[1]

1.3 Dynamická webová aplikácia

1.3.1 Životný cyklus webovej stránky

Po odoslaní požiadavky od klienta ju ďalej spracúva webový resp. servletový kontajner(napr. Tomcat). Ako hlavné fázy životného cyklu stránky JSP je možné uviesť:

- Inštanciáciu
- Spracovávanie požiadaviek
- Finalizáciu(deštrukciu)



Obrázok 4: Životný cyklus JSP stránky

Inštanciácia

Po prijatí požiadavky od klienta prv webový kontajner zistí, či jestvuje inštancia požadovanej stránky, ak áno, pracuje s ňou, ak nejestvuje, alebo ak preložená verzia je staršia ako kód jsp stránky, webový kontajner vytvorí inštanciu stánky v týchto fázach:

- Preklad – webový kontajner preloží kód jsp do java kódu. Vytvorí z neho servlet a teda výsledkom je java trieda typu servlet.
- Kompilácia – java trieda je preložená prostredníctvom javac na byte-kód, ktorý je možné spúšťať na JVM.
- Načítanie – preložený byte-kód je načítaný webovým kontajnerom.
- Inštanciácia – vytvorí sa inštancia servletového objektu danej triedy, aby mohla spracovávať požiadavky od klienta.
- Inicializácia – inicializujú sa objekty prislúchajúce danému servletu volaním metódy jspInit(). Po inicializácii servletu je možné spracovávať požiadvky od klientov.

Spracovávanie požiadaviek

V tejto fáze je dostupný inicializovaný servlet a pomocou metódy jspService() je obslúžená požiadavka od klienta. Odpoveď je zobrazená prostredníctvom metódy out().

Finalizácia

Pri vypínaní serveru, alebo pri potrebe získať väčšie množstvo dostupnej pamäte je inštancia jsp stránky odstránená pomocou metódy jspDestroy(). Po finalizácii servletu je nutné pre opätovné použitie objekt znova inicializovať.

Implementácia vlastnej aplikačnej logiky

Metódy jspInit() a jspService() je možné nahradiť resp. rozšíriť vlastnými metódami a aplikačným kódom – napr. pre vytvorenie a inicializáciu pripojenia do databázy, overenie spojenia a pod.

1.3.2 Aplikačný server Tomcat

Obrázok 5: Logo serveru Tomcat

Apache Tomcat, často nazývaný len ako Tomcat, je open-source webový server vyvíjaný spoločnosťou Apache Software Foundation (ASF). Server implementuje niekoľko Java EE komponentov ako je Java Servlet, JavaServer Pages (JSP), Java EL, a WebSocket, a poskytuje prostredie pre beh Java webových stránok. K serveru sa viaže open-source licencia Apache License 2.0 license.

Aktuálne najnovšou produkčnou verziou je verzia 8.0.32 označovaná ako stabilné vydanie, ako testovacia verzia je uvedená verzia 9.0.0.

Komponenty

- Catalina

Catalina je kontajner pre servlety serveru Tomcat, implementujúci špecifikáciu spol. Sun pre servlety a Java stránky - JavaServer Pages (JSP). Elementy tejto oblasti predstavujú zoznam používateľských mien, hesiel a ich rolí podobne ako systém oprávnení pre skupiny v Unixe. Rozličná implementácia elementov povoľuje kontajner integrovať do oblastí, kde už sú vytvárané a spravované systémy prístupov a následne dokáže pre tento systém prístupov integrovať bezpečnostné prvky podľa špecifikácie.

- Coyote

Tento prvok slúži ako spojovací komponent pre tomcat server a podporuje protokol HTTP 1.1, vďaka tomu je možné prostredníctvom kontajnera Catalina resp. priamo servletom a JSP kontajnerom vystupovať voči klientovi ako klasický webový server a poskytovať súbory a http dokumenty. Coyote počúva na špecifickom TCP porte na prichádzajúce spojenia, následne ich presmeruje na Tomcat server aby požiadavku spracoval a odoslal klientovi odpoveď. Mierne inú funkcionalitu ponúka prvok Coyote

JK, ktorý dokáže prichádzajúce požiadavky presmerovať na servery iného typu, ako napr. Apache.

- Jasper

Tento prvok slúži na preklad JSP súborov na servlety v Jave, ktoré môžu byť následne spracované Catalinou. Pri spustení prvok kontroluje, či v JSP súboroch nastala zmena a ak áno, preloží ich nanovo.

Prídavné funkcionality

Od verzie 7 poskytuje server Tomcat mimo iné aj tieto funkcie

- Možnosť zapojiť do servery do Clusteru

Táto možnosť bola pridaná najmä pre správu veľkých aplikácii – napr. na rozloženie záťaže pri veľkom množstve súčasných požiadaviek.

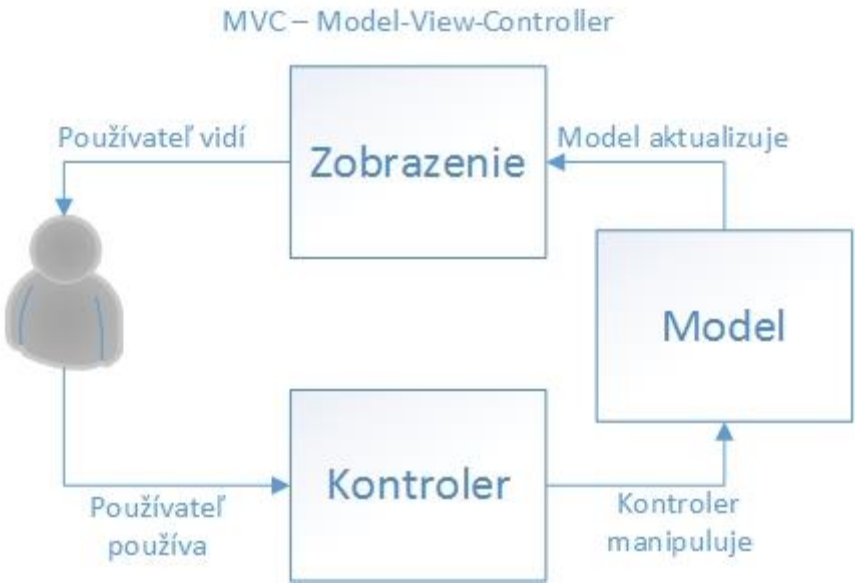
- Vysoká dostupnosť (HA)

Možnosť zrkadliť servery buď pre výskyt výpadku, alebo pri aktualizácii serverových komponentov.

1.3.3 Prezenčná knižnica Stripes

Stripes je Java webová knižnica s cieľom uľahčiť vývoj webstránok založených na servletoch a JSP. Stripes preferuje princípy odľahčených komponentov s nízkym počtom externých závislostí na balíčkoch, zrýchlený a iteratívny spôsob vývoja pre vývojárov a integráciu do už jestvujúcich riešení. Táto knižnica je založená na myšlienke, robiť málo ale robiť to veľmi dobre.

Stripes je prezentačná knižnica určená pre webové aplikácie, je možné ju zaradiť medzi MVC knižnice. Tento doplnok na základe konfigurácie prostredníctvom anotácií priraďuje HTTP požiadavky konkrétnym metódam typu *Resolution*, ktoré slúžia ako kontrolér a ďalej spracúvajú pridelenú požiadavku. Prostredníctvom tohto doplnku je taktiež možné pracovať s lokalizovanými hodnotami textových premenných pomocou ich lokalizácie obvykle v súbore *stripesresources.properties*. Na základe zvoleného jazyka je knižnica schopná používať danú lokalizáciu naprieč celou aplikáciou.



Obrázok 6: Model MVC

1.3.4 Java persistence API

Java Persistence API (JPA) je špecifikácia programovacieho jazyka Java, ktorá umožňuje objektovo relačné mapovanie (ORM). To zjednodušuje prácu s ukladaním objektov do databázy a naopak. Je určený ako pre Java SE, tak aj pre Java EE.

Entity JPA

Entita je objekt, ktorý reprezentuje dáta v databáze. Typická trieda pre entitu reprezentuje tabuľku v relačnej databáze a každá inštancia tejto triedy tak korešponduje s jedným riadkom tabuľky.

Aby mohli byť entity perzistované, musí mať trieda vytváraná pre entitu nasledujúce vlastnosti:

- Musí byť anotovaná anotáciou *javax.persistence.Entity*
- Musí mať *public* alebo *protected* konštruktor bez parametrov. Môže mať však aj ďalšie konštruktory.
- Nesmie byť deklarovaná ako *final*. To platí aj pre jej metódy.
- Ak relácia beanu bude pracovať s inštanciami tejto triedy a bude typu *Remote*, potom táto entitná trieda musí implementovať interface *Serializable*
- Môže dediť z entitnej aj z ne-entitnej triedy.

- Jej atribúty musia byť deklarované ako private, protected alebo package-private a je k nim možné pristupovať len cez metódy danej triedy(gettery a settery). [11]

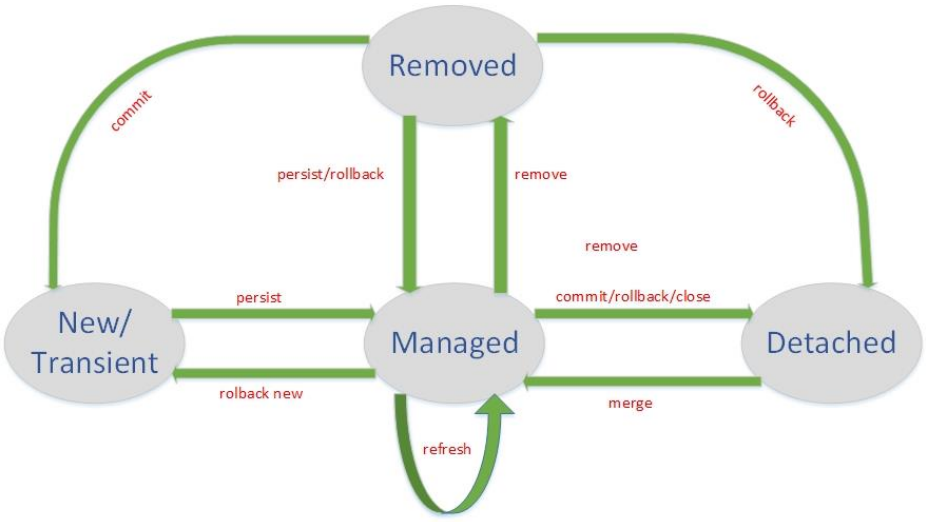
Aby mohla byť entita uložená do databázy, dátový typ jej atribútov sa musí nevyhnutne nachádzať v tabuľke nižšie. Iné dátové typy atribútov nie sú podporované.

primitívne typy		**Enumeračné typy**
java.lang.String	byte[]	iné entity a/alebo kolekcie entít
Wrapper triedy primitívnych typov	Byte[]	Triedy s anotáciou Embeddable
java.math.BigInteger	char[]	
java.math.BigDecimal	Character[]	
java.util.Date		
java.util.Calendar		
java.sql.Time		
java.sql.TimeStamp		

Životný cyklus entity

Každá entita má určitý stav, v ktorom sa v každom okamihu svojej existencie nachádza. O tomto stave rozhoduje inštancia triedy EntityManger. Entita sa pohybuje medzi jednotlivými stavmi vždy po prevedení určitej akcie EntityMangerom. Počiatočným stavom entity je vždy New/Transient, žiadny jej stav nie je konečný. [11]

Všetky stavy entity a hrany medzi nimi vystihuje nasledujúci graf:



Obrázok 7: Životný cyklus entity

Existujú 4 typy vzťahov v entitných triedach:

- One-to-one: inštancia má referenciu na jednu entitu inej triedy.
- One-to-many: inštancia má referenciu na množinou inštancií inej triedy.
- Many-to-one: viac inštancií má referenciu na jednu inštanciu inej triedy.
- Many-to-many: viac inštancií má referencie na inštancie iných tried.

Existujú 2 typy smerov mapovania:

- Unidirectional: inštancia má referenciu na iný objekt, ten ale nemá referenciu naspäť.
- Bidirectional: inštancia má referenciu na iný objekt a ten má tiež referenciu naspäť. Vidia sa navzájom.

Správca entít

Entity sú spravované objektom triedy *EntityManager*. Ak ho chceme získať v *SessionBeane*, musíme deklarovať atribút typu *EntityManager* a anotovať ho pomocou *PersistenceContext*. [11]

Základné metódy *EntityManageru* pre prácu s objektami

Pre *EntityManager* s názvom *EntMan* a entitu s názvom *entita* by operácie vyzerali nasledovne:

- EntMan.persist(entita): uloží objekt entita do databázy (INSERT)
- EntMan.remove(entita): vymaže objekt entita z databázy (DELETE)
- EntMan.merge(entita): entita bola perzistovaná, ale následne bola zmenená. Po operácii merge sa tieto zmeny prejavia v databáze (UPDATE).
- EntMan.find(class,id): vráti objekt v tabuľke, ktorá korešponduje s triedou *class* a má primárny kľúč *id* (SELECT)

Java Persistence Query Language

EntityManager dokáže vytvárať dotazy podobné SQL dotazom. Pre tieto operácie sa využíva *Java Persistence Query Language* -jazyk podobný jazyku SQL.

Ako jeho hlavný prínos je možné uviesť to, že dotazy sú nezávislé na zvolenej technológii databázy (Oracle, Microsoft a iné). Tieto dotazy majú objektové vlastnosti a teda v dotazoch nespomíname konkrétne názvy tabuliek databázy, či ich vlastností, ale priamo názvy tried resp. atribútov v jazyku Java. K porovnávaniu objektov môžeme využívať i ich referencie. [11]

Implementácia JPA

Java Persistence API je len špecifikáciou, nie je to však implementácia, ktorá by bolo možné použiť na konkrétnom príklade. Medzi implementácie JPA je možné zaradiť tieto knižnice:

- Hibernate
- Oracle Toplink
- OpenJPA

1.3.5 Knižnica Hibernate ORM

Hibernate je knižnica napísaná v jazyku Java, umožňujúca tzv. Objektovo-relačné mapovanie (ORM), vytvoril ho Gavin King a v roku 2006 spolu s projektom prešiel do zázemia spoločnosti Red Hat.. Uľahčuje riešenie otázky zachovania dát objektov i po ukončení behu aplikácie. Je jednou z implementácií Java Persistence API (JPA). [12]

Hibernate poskytuje spôsob, pomocou ktorého je možné zachovať stav objektov medzi dvoma spusteniami aplikácie. Stav dát v tejto situácii sa nazýva perzistentný. Tento stav je možné dosiahnuť pomocou ORM, čo znamená, že mapuje objekty jazyka Java na entity v relačnej databáze. Na to sú využívané mapovacie súbory, ktoré obsahujú informácie o tom, akým spôsobom sa majú dáta z objektu transformovať do databázy a naopak a akým spôsobom sa z databázových tabuliek majú vytvoriť objekty. Druhý spôsob ako mapovať objekty je použiť anotácie namiesto mapovacích súborov. [12]

V Hibernate sa teda pracuje s bežnými biznis objektami, pričom môžu byť stĺpce tabuľky spojené priamo s atribútmi objektu, alebo môžu byť prepojené pomocou metódy *get / set* a metódy *hashCode()* a *equals()*. Táto knižnica podporuje klasické *POJO* objekty(Plain Old Java Object). [12]

Po uložení objektov do databázy je k nim možné pristupovať pomocou jazyka *HQL* (Hibernate Query Language), ktorý je odvodený z SQL a je mu veľmi podobný.

Mapovanie

- Mapovacie súbory

Jedná sa súbory vo formáte XML s príponou ".hbm.xml" (teda napr. Den.hbm.xml), keď pre jednu triedu máme jeden súbor, zvyčajne umiestnený v rovnakom balíku ako samotná trieda. V štruktúre mapovacieho súboru sa používajú napr. Tagy <property> alebo <set>. Prvým sa popisuje normálny dátový atribút triedy. Tagom <set> sa tiež opisuje atribút, ale tentoraz má špeciálny význam, pretože udržiava odkazy na súvisiace objekty. [12]

```
<hibernate-mapping>
    <class name="databaseObjects.Den" table="DEN">
        <id name="idDen" type="int">
            <column name="IDDEN" />
            <generator class="assigned" />
        </id>
        <property name="den" type="java.lang.String">
            <column name="DEN" />
        </property>
        <list name="prednaskas" inverse="false" table="PREDNASKA" lazy="true">
            <key>
                <column name="IDDEN" />
            </key>
            <list-index></list-index>
            <one-to-many class="databaseObjects.Prednaska" />
        </list>
    </class>
</hibernate-mapping>
```

Obrázok 8: Mapovací súbor tabuľky deň

- Anotácie

Označenie prvkov anotáciou možno použiť dvoma spôsobmi – buď priamo pred atribútom alebo pred ich metódami.

```
1   @Entity(access = AccessType.FIELD)
2   @Table (name="Den")
3   public class Zajezd {
4
5       @Id (generate = GeneratorType.AUTO)
6       private int idden;
7
8       @Column (length=100, nullable=false, unique=true)
9       private String den;
10
11  }
```

Obrázok 9: Použitie anotácie pred atribútom

```java
@Entity(access = AccessType.PROPERTY)
@Table (name="Den")
public class Den {

    private int idden;
    private String Den;

    @Id (generate = GeneratorType.AUTO)
    public int getIdden() {
        return this.idden;
    }

    @Column (length=100)
    public String getDen() {
        return this.Den;
     }
    }
```

Obrázok 10: Použitie anotácie pred metódami

Konfiguračný súbor pre Hibernate

Pre korektné fungovanie potrebuje Hibernate nastaviť konfiguračný súbor, uviesť treba používaný ovládač databázy, adresu databázy a prístupové údaje a v neposlednom rade odkaz na vytvorené POJO objekty. V štandardnej konfigurácii má tento súbor názov *hibernate.cfg.xml*.

```xml
<hibernate-configuration>
    <session-factory>
<!-- connection na databazu -->
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/workground</property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

  <!-- mapping resources -->
  <mapping resource="databaseObjects/Den.hbm.xml" />
  <mapping resource="databaseObjects/Udaje.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

Obrázok 11: Konfiguračný súbor HIbernate

Hibernate Query Language – HQL

Preferovaný spôsob práce s objektami je prostredníctvom jazyka HQL. Jeho syntax sa veľmi podobá jazyku SQL, no HQL je objektovo orientovaný a teda rozumie prvkom ako sú dedičnosť, polymorfizmus a pod. HQL podporuje bežne používané konštrukcie jazyka SQL ako sú: INNER a OUTER JOIN, klauzula WHERE alebo WITH, agregačné výrazy ako AVG() alebo COUNT(), subdotazy atď. V prípade potreby je možné vykonávať aj natívne SQL príkazy. [12]

1.3.6 Knižnica značiek JSTL

JavaServer Pages Standard Tag Library (JSTL) je kolekcia užitočných JSP značiek, ktorá zapuzdruje základné funkcie spoločné pre mnoho aplikácií JSP.

JSTL podporuje bežné štrukturálne úlohy, ako je iterovanie a podmienené výrazy, značky pre manipuláciu s XML dokumentami, internacionálne značky a SQL značky. Taktiež poskytuje knižnicu pre integráciu existujúcich vlastných značiek pomocou JSTL značiek. Ďalej budeme tieto značky označovať ako tagy.

JSTL značky všeobecne môžu byť klasifikované podľa ich funkcií do skupín knižníc JSTL tagov, ktoré môžu byť použité pri vytváraní JSP stránky:

- Hlavné Tagy
- formátovacie tagy
- SQL tagy
- XML tagy
- JSTL Funkcie

```
<c:forEach items="${actionBean.top5list}" var="item"
    varStatus="status">
    <tr>
        <td>${status.count}</td>
        <td>${item.meno}</td>
        <td>${item.priezvisko}</td>
        <td>${item.total}</td>
    </tr>
</c:forEach>
```

Obrázok 12: Príklad použitia JSTL

Na obrázku je možné vidieť príklad použitia tagov JSTL, konkrétne iteratívne vypísanie hodnôt objektu top5list.

2. Cieľ práce, metodika práce a metódy skúmania

2.1 Cieľ a metodika práce

Cieľom tejto záverečnej práce bolo vytvoriť funkčnú aplikáciu, ktorá by hravou formou nabádala používateľov - študentov aby sa pravidelne zúčastňovali výukového procesu v maximálne možnej miere a zároveň používali aplikáciu, ktorá bude slúžiť na zaznamenávanie, porovnávanie a oceňovanie dochádzky pre každého používateľa zvlášť a taktiež by mohla slúžiť pre vyučujúcich a pracovníkov školy ako prostriedok na sledovanie priebežnej účasti študentov na uskutočnených prednáškach, či cvičeniach a tiež na záverečný súhrnný dochádzkový zoznam a prideľovanie bodov za dochádzku.

Aplikácia by mala pozostávať z mobilnej aplikácie pre telefóny s OS Android s a webového rozhrania dostupného cez akýkoľvek aktuálny internetový prehliadač.

Mobilná aplikácia

Android aplikácia by mala prioritne slúžiť na zaznamenávanie účastí študentov na výukovom procese z dôvodu možnosti overiť si polohu zariadenia prostredníctvom GPS polohy získanej zo siete. Tento spôsobom získavania polohy je pomerne dôveryhodný, síce je možné poskytnúť zariadeniu nepravdivú polohu tak, aby ju považovalo za autentickú, avšak spôsob ktorým to je možné urobiť nie je jednoduchý, ani rozšírený. V aplikácii si používateľ bude môcť zobraziť informácie o svojej dochádzke, získaných úspechoch a oceneniach.

Webové rozhranie

Rozhranie, ktoré bude dostupné cez webový prehliadač v značnej miere rozšíri funkcionalitu aplikácie hneď v niekoľkých smeroch. Po načítaní stránky bude systém rozlišovať niekoľko typov používateľských účtov:

- Neprihlásený používateľ
- Študent
- Vyučujúci
- Správca/admin

Oprávnenia a dostupné funkcie pre jednotlivé typy účtov sú detailnejšie špecifikované v záverečnej kapitole tohto dokumentu.

Ďalším z rozšírení webového rozhrania bude možnosť pre používateľov typu vyučujúci potvrdiť účasti zaznamenané študentami na konkrétne prednášky, či pridelenie bonusových bodov za aktivitu ku týmto účastiam a tiež zobrazenie sumárneho zoznamu počtu účastí študentov a ich získaných bonusových bodov podľa vyučovaných predmetov.

V tomto rozhraní budú tiež implementované rozličné grafické zobrazenia rebríčkov a úspechov, napr. najúspešnejšia fakulta, ročník, či študent a tiež súhrnné štatistiky v podobe počtu používateľov v systéme podľa fakúlt, či podľa typu účtu.

Pre typ účtu admin bude hlavnou funkcionalitou pridávanie a úprava údajov v databáze a potvrdzovanie novovytvorených používateľských účtov.

2.2 Vývojové prostredie a použité doplnky

Vývojovým prostredím pre webové rozhranie bude program Eclipse - edícia programu s názvom Eclipse Mars. Toto prostredie bolo vybrané na základe kompatibility s viacerými OS a možnosti spúšťať program bez nutnosti inštalácie, čo taktiež zabezpečuje jeho jednoduchú prenositeľnosť na iné systémy. Pre vývoj Aplikácie na Android bude použitý program Android Studio. Tento program bol vybraný z dôvodu pomerne rozsiahlej dokumentácie programu a systému a tiež širokej komunite vývojárov využívajúcej tento program. Ako prostredie pre beh databázy bude slúžiť MySQL server. Tento server bol vybraný z dôvodu jeho jednoduchej inštalácie a prenositeľnosti na iné prostredia a taktiež preto, že pre používanie tohto servera nie je nutné zakúpiť si licenciu. Pre implementáciu a testovanie webovej aplikácie bude použitý webový kontajner Tomcat server. Tento server bol vybraný z dôvodu rozsiahlej komunity vývojárov využívajúcich tento server a jednoduchosti implementácie.

Mobilná aplikácia bude komunikovať s MySQL databázou prostredníctvom internetového pripojenia mobilného zariadenia. Ako formát pre výmenu údajov medzi mobilným zariadením a databázou bude použitý štandard JSON, kvôli nízkemu objemu dát potrebných na výmenu.

Pre komunikáciu webového rozhrania s databázou a taktiež mapovanie databázy do rozhrania bude použitá knižnica Hibernate. Pre zabezpečenie prezenčnej vrstvy bude použitá knižnica Stripes, z dôvodu dostatku dokumentácie, priamočiarej počiatočnej implementácii a možnosti rozšírenia bez nutnosti doplňujúcej konfigurácie. Pre zobrazenie dynamického obsahu na samotných jsp stránkach bude použitá knižnica JSTL, keďže sa jedna o knižnicu štandardných zobrazovacích prvkov využívaných na zobrazovanie dynamického obsahu.

2.3 Požiadavky pre implementáciu a beh systému

Systém bude pozostávať z klientskej a serverovej časti. Používateľskú časť bude tvoriť aplikácia pre Android a webové rozhranie, kde obe budú pozostávať z Java tried slúžiacich na vykonávanie funkcionalít a aplikačnej logiky a tiež xml a jsp súborov slúžiacich na zobrazovanie obsahu. Pre plnohodnotné využívanie aplikácie pre Android je nevyhnutné zabezpečiť internetové pripojenie mobilného zariadenia. Serverová časť bude pozostávať z aplikačného servera podporujúceho všetky požadované funkcionality dostupného na verejnej adrese.

Pre nasadenie systému bude potrebné zabezpečiť externý aplikačný server s podporou vykonávania skriptov v jazyku php, možnosťou nasadenia MySQL databázy a taktiež možnosťou inštalácie a použitia webového serveru Tomcat.

3. Výsledky práce a diskusia

3.1 Popis systém ako celku

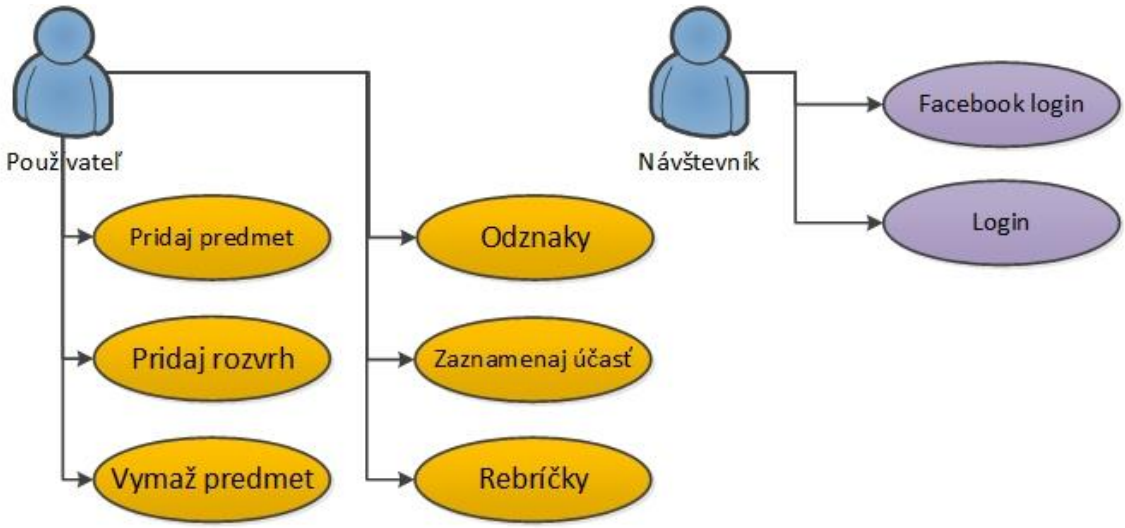
Tento systém bol navrhnutý preto, aby hravou formou, pomocou využitia prvkov gamifikácie povzbudzoval užívateľov, teda študentov k navštevovaniu prednášok a cvičení. Systém pozostáva z Android aplikácie a webového rozhrania vytvoreného pomocou programovacieho jazyka Java pracujúcich s rovnakou databázou.

Android aplikácia by mala primárne slúžiť na zaznamenávanie účasti študentov a webové rozhranie by malo slúžiť ako centrálny prístupový bod s rozšírenými možnosťami schvaľovania účastí zaznamenaných študentami. Obe rozhrania obsahujú aj mnoho prídavných funkcionalít a možností, ktoré by ich mali činiť pre používateľov atraktívnejšími a taktiež praktickejšími.

3.2 Popis Android aplikácie

Po spustení Android aplikácie sa používateľovi zobrazí prihlasovacia obrazovka, kde má možnosť prihlásiť sa prostredníctvom Facebook účtu, ale aj pomocou lokálneho účtu. Pri neúspešnom prihlásení sa používateľovi aplikácie znova načíta prihlasovacia obrazovka, po úspešnom prihlásení používateľa privíta úvodná obrazovka, kde má na výber z niekoľkých možností. Pre úpravu zoznamu navštevovaných predmetov, pomocou tlačidla nastav rozvrh sa môže prepnúť na ďalšiu obrazovku, kde je možné vybrať si, či chce ku svojmu účtu pridať predmet, rozvrh podľa zaradenia do študijnej skupiny, alebo vymazať predmet.

Po zvolení možnosti zaznamenaj na hlavnej obrazovke sa pre používateľa automaticky načítajú dáta zo servera podľa zoznamu prednášok priradených k danému účtu. Prvkom overenia faktu, či sa používateľ ozaj nachádza v škole, prípadne aspoň v jej blízkosti je podmienkou pre pridanie účasti implementované overenie polohy Android zariadenia podľa informácií zo siete WI-FI, prípadne mobilného pripojenia. Táto možnosť bola zvolená s ohľadom na výdrž batérie mobilného zariadenia. Po zvolení tlačidla úspechy na hlavnej obrazovke sa používateľovi zobrazí rebríček najlepších používateľov a tiež všetky odznaky, ktoré používateľ získal.

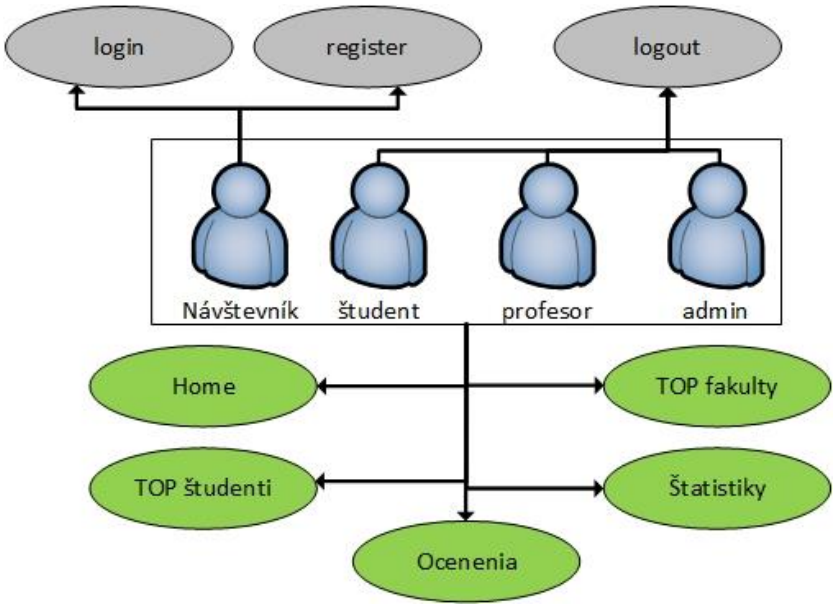


Obrázok 13: Diagram prípadov použitia Android aplikácie

Android aplikácia rozlišuje dva typy používateľov a to neprihláseného používateľa – teda návštevníka a prihláseného používateľa – teda študenta. Graf (obr.13) znázorňuje diagram prípadov použitia.

3.3 Popis webového rozhrania

Po spustení webového rozhrania používateľa privíta úvodná obrazovka, kde má možnosť sa odkázať na stránku pre prihlásenie, či registráciu. Na tejto stránke má tiež možnosť odkázať sa na všeobecné informácie o systéme a ich grafy– napr. rebríček najlepších študentov podľa celkovej účastí, či detailnejšie podľa fakúlt a ročníka. Prezrieť si je tiež možné rebríček najlepších fakúlt podľa účasti študentov a graf. Dostupné sú tiež informácie o oceneniach, ktoré je možné získať a tiež aj o štatistikách systému napr. ako počty používateľov v systéme, či podľa fakúlt. Diagram prípadov použitia je znázornený na obrázku č.14.

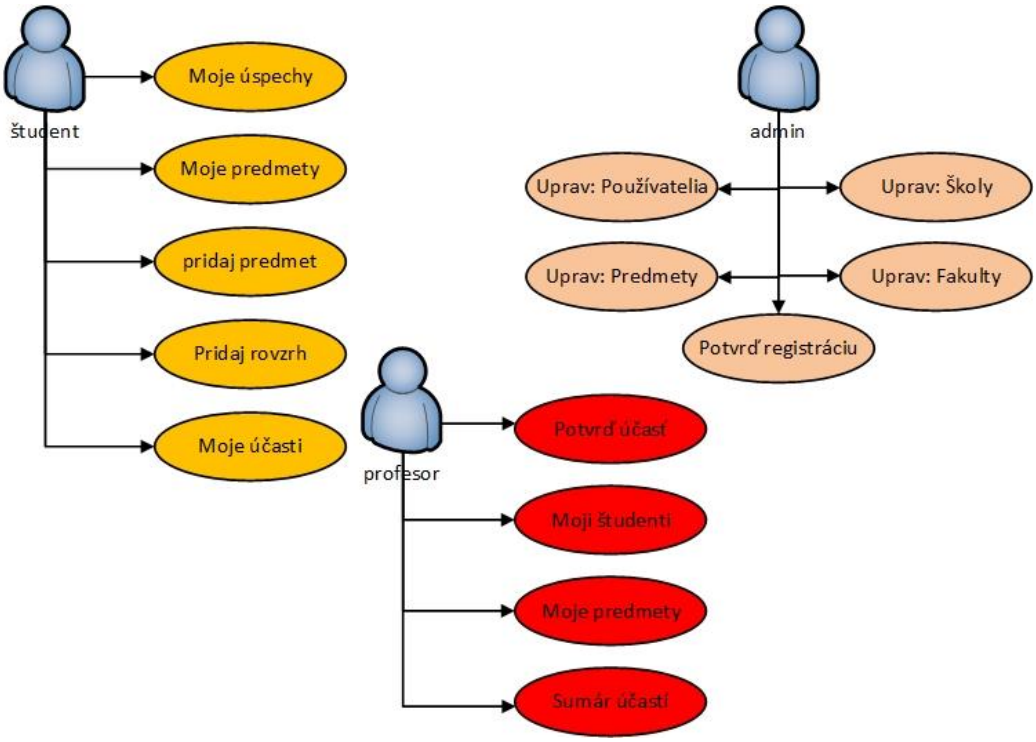


Obrázok 14: Všeobecný diagram prípadov použitia webového rozhrania

Po prihlásení systém rozpozná typ používateľského účtu a na základe neho načíta prislúchajúcu obrazovku pre daný typ účtu. Pre typ používateľa študent sú k dispozícii funkcie pre zobrazenie dosiahnutých úspechov, pridaných predmetov a zaznamenaných účastí a tiež aj možnosť pridať si do študijného plánu predmet, či sadu predmetov v podobe rozvrhu pre celý krúžok. Tento typ používateľského účtu si taktiež môže zobraziť všetky obrazovky dostupné aj neprihlásenému používateľovi.

Pre typ používateľa profesor je možné zobraziť si zoznam vyučovaných predmetov, zoznam študentov, ktorí navštevujú predmety daného vyučujúceho a na základe zaznamenaných účastí študentov na prednáškach tieto účasti potvrdiť a ku každému študentovi pre konkrétnu účasť potvrdiť alebo nepotvrdiť zaznamenanú účasť a tiež ku každej takto zaznamenanej účasti priradiť od 0 do 5 bonusových bodov. Profesor si môže tiež pre vyučované predmety zobraziť sumár účastí študentov a im pridelených bonusových bodov podľa jednotlivých predmetov, ktoré vyučuje.

Pre typ používateľa admin resp. správca databázy je možné po prihlásení upravovať hlavné doménové údaje databázy a to používateľov, predmety, školy a fakulty a tiež možnosť pre novovytvorené účty zaregistrované pomocou webového rozhrania tieto účty schváliť a každý účet priradiť ku konkrétnej fakulte a nastaviť typ používateľského účtu. Diagram prípadov použitia pre jednotlivé typy používateľských účtov je znázornený na obrázku č. 15.



Obrázok 15: Diagram prípadov použitia používateľov webového rozhrania

3.4 Implementácia systému prihlasovania

Prihlasovanie do Android aplikácie

Do Android aplikácie sa je možné prihlásiť pomocou účtu sociálnej siete Facebook, alebo pomocou lokálneho účtu vytvoreného prostredníctvom registrácie cez webové rozhranie. Pri prvom otvorení Android aplikácie sa používateľovi načíta úvodná obrazovka, kde je možné si zvoliť spôsob prihlásenia - verejne známe modré Facebook prihlasovacie tlačidlo alebo kombináciou lokálneho používateľského mena a hesla. Po stlačení Facebook prihlasovacieho tlačidla aplikácia buď v zariadení vyhľadá už uložené účty, alebo vyzve používateľa na zadanie prihlasovacích údajov a následne ho prihlási, pri prihlasovaní cez lokálny používateľský účet bude používateľ vyzvaný na zadania mena a hesla.

V prípade zvolenia možnosti prihlásenia cez Facebook účet si aplikácia po prihlásení vyžiada povolenia na získanie informácii z profilu a informácií o zozname priateľov. Po úspešnom prihlásení a získaní Facebook ID používateľa, mena a priezviska aplikácia volaním do databázy overí podľa Facebook ID, či sa už používateľ v zozname nachádza, ak áno, vráti jeho ID používané v databáze, ak nie, podľa získaných údajov vytvorí nového používateľa a následne taktiež vráti jeho

novovytvorené ID, ktoré sa bude používať v rámci databázy. Pre pridanie istej miery zabezpečenia proti automatizovaným nástrojom na internete bol do aplikácie pridaný systém overovania používateľov prostredníctvom verifikácie novovytvorených používateľských účtov administrátorom a jeho následné zaradenie do skupiny podľa školy a fakulty. Pri implementácii prihlasovania boli použité zdrojové kódy vývojárskej sady Facebook SDK[9] spolu s Apache licenciou „the License“ verzie 2.0[10].

Prihlasovanie do webového rozhrania

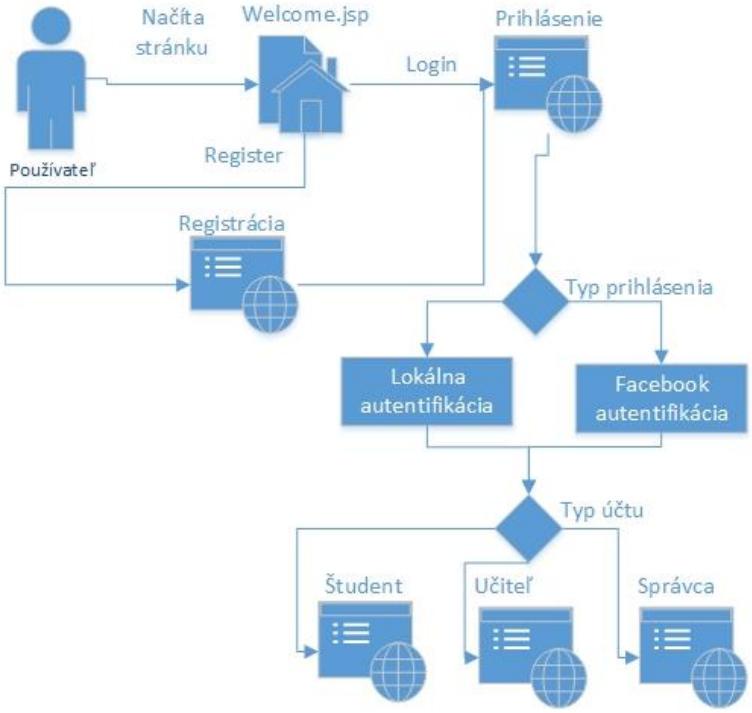
Rozhranie dostupné prostredníctvom prehliadača poskytuje rovnaké možnosti prihlásenia ako Android aplikácia a to prihlásenie prostredníctvom facebook účtu, ale aj možnosť prihlásiť sa pomocou lokálneho účtu. Možnosť registrácie lokálneho účtu bola implementovaná jedine vo webovom rozhraní. Po zadaní prihlasovacieho mena a hesla pre lokálne prihlásenie je následne heslo zašifrované algoritmom md5 a spolu s používateľským menom je overované voči databáze, či daný používateľ jestvuje a môže sa prihlásiť. Systém pre zadané vstupy preveruje, či sa jedná o korektné hodnoty premenných a tiež, či majú správny rozsah, aby sa predišlo neoprávneným zásahom do databázy. Minimálna dĺžka prihlasovacieho mena sú 3 znaky a minimálna dĺžka hesla je 6 znakov. Systém po prihlásení rozlišuje niekoľko typov používateľských účtov, a to:

Tabuľka 1: Typy používateľských účtov a ich oprávnení

Typ používateľského účtu	Popis oprávnení	Používatelia
Neprihlásený používateľ	Tento používateľ bude mať možnosť otvoriť si úvodnú stránku rozhrania, rebríčky najlepších študentov, či fakúlt a sekciu so všeobecnými informáciami	Verejnosť
Študent	Študentský účet slúžiaci na pridávanie účastí a prehliadanie úspechov. Budú môcť prehliadať zoznamy predmetov, rozvrhy podľa krúžkov, svoje účasti a získané úspechy.	Študenti
Vyučujúci	Majú práva na prehliadanie zoznamu predmetov, rozvrhov a možnosť potvrdzovať účasti nimi vyučovaných predmetov pre študentov. Ku každej	Vyučujúci, cvičiaci, doktorandi

	študentskej účasti na cvičení bude možnosť pridať číselné ohodnotenie – bonusové body. Môžu si zobraziť sumár účasti študentov nimi vyučovaných predmetov.	
Správca/admin	Tento typ účtu slúži na správu celej databázy a teda bude mať oprávnenia na akékoľvek úpravy produkčných dát na databázovom serveri, taktiež slúži na schvaľovanie nových účtov podľa fakulty a typu účtu.	Vyučujúci, zamestnanci katedry

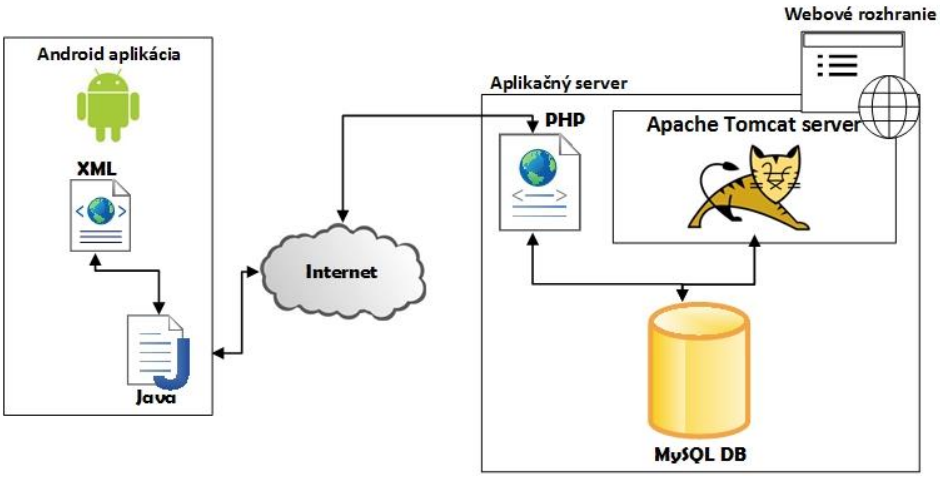
Pre každý typ používateľského účtu je po autentifikácii dostupné prostredie prislúchajúce možnostiam daného typu používateľského účtu ako aj hlavná časť dostupná všetkým typom účtov bez ohľadu na stav prihlásenia(obr. 16).



Obrázok 16: Prihlasovanie do webového rozhrania na základe typu používateľského účtu

3.5 Návrh komponentov systému

Pre korektné fungovanie systému je potrebné aby na jednom aplikačnom serveri mohli popri sebe fungovať aplikačný php skript slúžiaci na komunikáciu Android aplikácie s databázou a aplikačný server Tomcat slúžiaci ako servletový kontajner pre aplikáciu webového rozhrania. Prepojenie komponentov znázorňuje schéma na obrázku(obr. 17). Android aplikácia komunikuje s databázou vďaka internetovému pripojeniu. Ako prostriedok na výmenu dát slúži aplikačný php skript umiestnený na serveri spolu s databázou. Keďže aplikačný server pre webové rozhranie je umiestnený na totožnom serveri ako aj systémová databáza, spojenie na databázu sa deje natívne prostredníctvom knižnice Hibernate ORM na porte 3306.



Obrázok 17: Architektúra fungovania systému

3.5.1 Návrh Java tried

Android Aplikácia pozostáva celkovo z 16 tried rozdelených do 2 balíčkov, 8 tried slúži priamo na vytváranie aktivít a zachytávanie interakcie používateľa, ďalšie 4 na komunikáciu zo serverom a posledné 4 slúžia na vykreslenie zobrazovaných údajov. Každá obrazovka je tvorená samostatnou aktivitou, ktorá ju po inicializácii vykoná a následne vykreslí. Časť *Rebríčky* a *odznaky* netvorí samostatná aktivita, ale sú zobrazované v rámci aktivity *úspechy*.

Webové rozhranie je tvorené z 31 tried, rozčlenených do 6 balíčkov podľa spôsobu použitia triedy. Perzistenciu databázy a prístup k dátam zabezpečujú celkovo 3 balíčky *databaseObjects*, *dataIntern* a *DB* pozostávajúce spolu z 22 tried, kde 14 tried mapuje tabuľky databázy, 4 slúžia ako lokálne dátové triedy a 4 zabezpečujú vytvorenie

spojenia s databázou, prístup do databázy a vykonávanie priamych SQL príkazov. Na uchovávanie objektu používateľa naprieč aplikáciou slúži balíček *customActionBeanContext* pozostávajúci z dvoch tried. Trieda v balíčku *tools* zabezpečuje šifrovanie textu a na napĺňanie obsahu jednotlivých podstránok systému slúži balíček *Workers* pozostávajúci z 6 tried, kde každá reprezentuje prvok pre manipuláciu s dátami a ich poskytnutie podstránkam.

3.5.2 Znovapoužitie prvkov na webstránke

Pri implementácii aplikácie bol použitý koncept znovapoužitia už vykreslených prvkov zastrešený prezenčnou knižnicou Stripes – *stripes:layout*. Pri inicializácii webového rozhrania sa použije stránka *body.jsp*, ktorá vytvorí štruktúru pre vkladanie prvkov - hlavičky, tela a päty stránky a tiež pomocou JavaScript funkcie vytvorí spojenie s Facebook autentifikačným rozhraním. Pri otvorení stránky sa teda prvky ako hlavička a päta a štruktúra vykreslia iba raz a následne sa dynamicky mení iba telo stránky s dátami. Prvok hlavička zobrazí tlačidlo na prihlásenie, resp. odhlásenie na základe údaju o tom, či sa používateľ už prihlásil, alebo nie. Všetky stránky sa teda načítajú len raz a na základe typu dostupných dát zobrazia alebo nezobrazia obsah. Na obrázku (obr. 18) je znázornená štruktúra hlavnej stránky, ktorá na základe prvkov *stripes:layout-component* zobrazí komponenty - *header* a *contents*. Pri zmene nejakého komponentu ako napr. *contents* – teda tela stránky sa opätovne vykreslí len tento prvok. Na obrázku (Obr.19) je znázornená štruktúra stránky v jazyku XML, ktorá na základe typu objektu získaného cez rozhrania *ActionBean* zobrazí konkrétne dáta.

```
<stripes:layout-definition>

    <html>
  <head>
    <link rel="shortcut icon" href="/images/favicon.ico" type="image/x-icon">
    <title>DiploFirst - ${title}</title>
    <stripes:layout-component name="html-head" />
  </head>
  <body id="mainBody" class="mainbody">
    <script>
     <div id="contentPanel">
        <stripes:layout-component name="header">
            <jsp:include page="/jsp/base/header.jsp" />
        </stripes:layout-component>
        <div id="pageContent">
            <stripes:layout-component name="contents" />
        </div>
        <div id="footer">
            <br>
            <stripes:label for="Createdby"/>
            <stripes:label for="owner" />
            @FHI 2016 | Built on Eclipse |
            <div class="fb-like" data-share="true" data-width="450"
                data-show-faces="true"></div>
        </div>
    </div>
</body>
    </html>
</stripes:layout-definition>
```

Obrázok 18: Hlavná stránka

Obrázok 19: Štruktúra stránky študentov

Ďalším z prvkov, ktoré je možné opätovne využívať sú k parametrom mapované texty uvedené v súbore *stripesresources.properties*. V prípade potreby zobraziť ten istý popis, či reťazec viackrát sa naň stačí odkázať názvom parametru. Značnou výhodou tohto prístupu je korektné zobrazovanie špecifických znakov a diakritiky naprieč zobrazovanými prostrediami. Na obrázku (obr. 20) je zachytená časť mapovacieho súboru textov.

Obrázok 20: Ukážka mapovacieho súboru stripesresources.properties

3.5.3 Spúšťanie servletov a spracovávanie prijatých dát

Po odoslaní požiadavky na zobrazenie istej časti stránky požiadavku spracuje priradená trieda a na základe požadovanej akcie sa spustí rovnomenná trieda typu *Resolution*. Ak v požiadavke nebola špecifikovaná žiadna akcia, spustí sa metóda

označená anotáciou *@DefaultHandler. Servletové triedy je možné v tomto projekte volať pomocou názvu triedy a prípony .action* – napr. pre spustenie študentského rozhrania je volaný odkaz „*/studentBase.action*". Metódy typu Resolution vracajú ako návratovú hodnotu odkaz na ďalší servlet, prípadne jsp stránku. Na obrázku (obr. 21) je znázornený príklad servletovej triedy s ukážkou východzej metódy a návratovou hodnotou.

```
@DefaultHandler
public Resolution StudentBase() {
public Resolution myStatus() {
public Resolution mySubject() {
public Resolution addSubject() {
public Resolution addSchedule() {
return new ForwardResolution("/jsp/roles/studentBase.jsp");
public Resolution myPresence() {
public void setRozvrhy(ArrayList<Rozvrh> rozvrhy) {
public void saveSubjects(String[] input) {
public void removeSubjects() {
public void saveSchedule() {
}
```

Obrázok 21: Spracúvanie servletových požiadaviek

Ako servletové triedy, tak aj jsp stránky implementujú rozhranie ActionBean a teda pre prístup k dátovým premenným sa stačí v rámci jsp stránky odkázať na názov premennej prostredníctvom konštrukcie *${actionBean.nazovPremennej}*, kde k jednoduchým dátovým typom vieme pristupovať prostredníctvom ich zapisovacích a čítacích metód(gettery a settery) a štruktúrované dátové typy máme možnosť vypísať iteratívne, resp. k nim pristupovať v rámci jsp stránky aj objektovo. Príklad použitia premennej dostupnej prostredníctvom rozhrania actionBean je uvedený na obrázku(obr. 19).

3.5.4 Objektový prístup webstránky do databázy

Spojenie s databázou zastrešuje aplikačná knižnica Hibernate. Pri potrebe získať dáta u databázy sa inicializuje objekt-singleton typu CustomSessionFactory, ktorý na základe konfigurácie a prihlasovacích údajov vytvorí spojenie pre vytváranie relácií. Pre vytváranie dotazov na databázu slúži statická trieda *defaultDataString*, ktorej vstupným parametrom je reťazec znakov reprezentujúci natívny SQL dotaz a generická preťažená trieda *defaultDataObject*, ktorá vytvára a spravuje relácie na základe vstupných dát a obmedzení typu *Criterion* vracia zoznam žiadaných objektov typu ArrayList.

```
40    public static <T> ArrayList<T> getAllData(Class<T> clazz, Criterion inpCrit) {
41        SessionFactory factory = CustomSessionFactory.Generate();
42        Session session = factory.openSession();
43        Transaction t = session.beginTransaction();
44        Criteria criteria = session.createCriteria(clazz);
45        criteria.add(inpCrit);
46        ArrayList<T> back = (ArrayList) criteria.list();
47        t.commit();
48        session.close();
49        return back;
50    }
```

Obrázok 22: Získavanie dát z DB, jedna z variant metódy GetAllData()

Pre získanie konkrétnych dát z databázy je potrebné špecifikovať typ objektu a kritériá na základe ktorých bude vytvorené spojenie a vrátené dáta. Obrázok (Obrázok 23) znázorňuje príklad získania dát typu *Uzivatel* a *Fakulta* na základe obmedzení, kde typ účtu nesmie byť *null* a názov fakulty nesmie byť „dummy“.

```
public Resolution newUserManage() {
    System.out.println("new user manage");
    this.schvalenie = ((ArrayList<Uzivatel>) defaultDataObject.getAllData(Uzivatel.class,
            Restrictions.like("accounttype", "")));
    this.schvalenieFakulty = ((ArrayList<Fakulta>) defaultDataObject.getAllData(Fakulta.class,
            Restrictions.ne("fakulta", "dummy")));
    return new ForwardResolution("/jsp/roles/adminBase.jsp");
}
```

Obrázok 23: Metóda newUserManage(), príklad získania dát z databázy

3.5.5 Overenie GPS polohy Android zariadenia

V Android aplikácii bolo implementované overenie GPS polohy zariadenia zo siete pre možnosť pridať záznam účasti na prednáške, či cvičení. Na obrázku(obr. 24) je znázornená funkcia *isAtSchool*, ktorá na základe dostupných súradníc zariadenia a uložených súradníc školy porovná polohu mobilného zariadenia a vyhodnotí, či sa zariadenie v škole nachádza, alebo nie. Hodnoty GPS pre polohu školy boli získané pomocou nástroja Google Earth, presnosť súradníc je rádovo asi 100 metrov. Súradnice polohy Android zariadenia funkcia získava pomocou volania funkcie *getLocation*.

```
152    public boolean isAtSchool() {
153        Location loc = new Location(getLocation());
154        double lat = loc.getLatitude();
155        double lon = loc.getLongitude();
156        if (((lat >= 48.125) && (lat <= 48.13))
157                && ((lon >= 17.13) && (lon <= 17.1353))) {
158            return true;
159
160        } else
161            return false;
162    }
```

Obrázok 24: Funkcia isAtSchool

Pre zistenie súradníc zariadenia je nutné pridať do súboru *AndroidManifest.xml* povolenia pre aplikáciu *ACCESS_FINE_LOCATION* a *ACCESS_COARSE_LOCATION*.

3.6 Implementácia prvkov gamifikácie

V Android aplikácii je prítomných viacero gamifikačných prvkov. Po prihlásení do aplikácie a po načítaní dát používateľa sa zobrazia údaje o jeho dosiahnutej úrovni, celkovom postupe a počte účastí nutných na dosiahnutie ďalšej úrovne. Po načítaní počtu absolvovaných účastí používateľa a počte jeho návštev je aktivovaný kód, ktorý popisuje obrázok(obr. 25). Časť obrazovky, ktorá zobrazuje údaje o postupe nie je dostupná ihneď po prihlásení, zobrazí sa však akonáhle sú načítané dáta zo servera v poriadku a je možné ich spracovať.

Úrovne používateľov sú prerátavané na základe percentuálnej účasti na uložených prednáškach. Je možné dosiahnuť maximálnu úroveň 10 počnúc úrovňou 1, kde každá ďalšia úroveň znamená úspešné absolvovanie 10 % pridaných prednášok. Zobrazené sú tiež informácie o tom, koľko prednášok, či cvičení je nutné absolvovať pre dosiahnutie ďalšej úrovne. Zobrazované údaje vizualizuje ukazovateľ postupu.

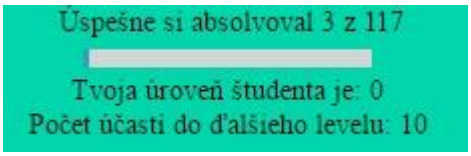
```
127 if (total != 0) {
128     LL.setVisibility(View.VISIBLE);
129     PB.setMax(total);
130     PB.setProgress(current);
131     float num =(((float) current / (float) total) * 100);
132     DecimalFormat df = new DecimalFormat("###.##");
133     progress_actual.setText("Váš postup: "+df.format(num) + " %");
134     student_lvl.setText("" + ((current / total) * 10)+" / 10");
135     next_lvl.setText(""
136             + (((((current / total) * 10) + 1) * 10) - current));
137
138 }
```

Obrázok 25: Zobrazenie postupu používateľa

Zoznam dosiahnutých úspechov používateľa a rebríček najúspešnejších piatich študentov je možné zobraziť po kliknutí na tlačidlo úspechy. Obsah prvej záložky tvorí zoznam dosiahnutých úspechov používateľa - ilustrácia úspechu, názov a popis. Všetky tieto dáta aplikácia získa z databázy podľa identifikátora používateľa. Druhú kartu tvorí zoznam prvých piatich študentov s najvyšším počtom účastí. Používatelia sú usporiadaní podľa výšky dosiahnutých bodov zostupne.

Gamifikačné prvky webového rozhrania

Vo webovom rozhraní je taktiež možné nájsť lištu s aktuálnymi ukazovateľmi postupu pre aktuálne prihláseného používateľa, znázornená je na obrázku (obr. 26).



Obrázok 26: Úroveň používateľa

Vo webovom rozhraní je taktiež možné nájsť rebríčky najlepších používateľov, či už podľa fakulty, podľa ročníka alebo podľa sumárnej účasti celkovo. Zobraziť si je možné tiež aj zoznam najlepších fakúlt podľa účastí. Všetky tieto číselné hodnoty sú doplnené o grafy vykresľujúce dané hodnoty. A j bez prihlásenia je možné si zobraziť zoznam úspechov, ktoré je možné získať. Prihlásenému používateľovi sa po kliknutí na časť „moje úspechy“ zobrazí obrázkový zoznam jeho dosiahnutých úspechov aj s popisom. Náhľady obrazoviek so spomínanými ukazovateľmi je možné nájsť v časti prílohy na konci tohto dokumentu .

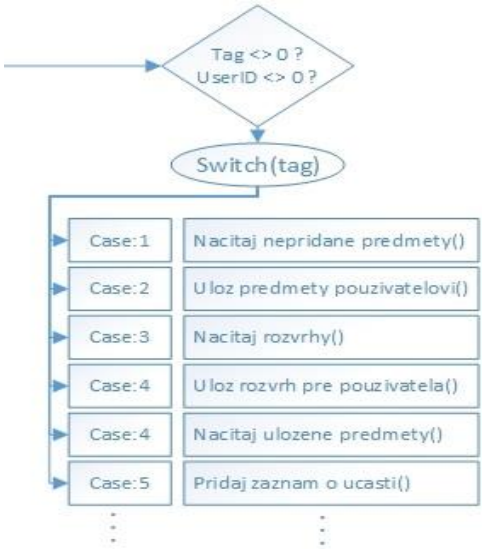
3.7 Aplikačný php skript

Prostriedkom komunikácie Android aplikácie s databázou je php skript umiestnený na aplikačnom serveri. Tento skript na základe typu požiadavky aplikácie vráti požadované dáta vo formáte JSON.

Pre všetky získavania dát z databázy je pre HTTP volania použitá metóda GET. Na základe hodnoty premennej *tag* a *UserID* sa zo zoznamu vyberie vetva funkcie *switch(),* ktorá sa vykoná a následne vráti údaje.

Pri ukladaní dát z Android zariadenia na server je použitá metóda POST, z dôvodu zabránenia odchytenia zasielaných údajov.

Pre robustnosť aplikačného php skriptu je uvedený len jeho model. Systém fungovania aplikačného skriptu znázorňuje diagram na obrázku (obr. 27).



Obrázok 27: Diagram fungovania php skriptu

3.8 Spojenie webového rozhrania s existujúcou databázou

Pre korektné fungovanie systému ako celku bolo nevyhnuté aby aplikácia aj webové rozhranie pristupovali k rovnakej a jednotnej databáze. Prístup Android aplikácie do databázy je implementovaný prostredníctvom priameho prístupu *mysqli()* cez .php skript umiestnený na aplikačnom serveri. Mapovanie tabuliek ako objektov bolo zabezpečené prostredníctvom knižnice Hibernate JPA, ktorá pre jestvujúcu štruktúru databázy vytvorila mapovacie súbory pre jazyk Java. Pre zachovanie totožnej aplikačnej logiky bolo nutné z prostredia webového rozhrania do databázy pristupovať ako objektovo, prostredníctvom JPA, tak aj priamo prostredníctvom jazyka SQL.

Pre integritu prihlasovacích údajov pre prihlásenie prostredníctvom sociálnej siete Facebook bolo nutné v programátorskom rozhraní pre aplikácie sociálnej siete Facebook zaradiť webovú aplikáciu do rovnakej skupiny ako Android aplikáciu, aby bolo pre každý účet využívajúci tento systém pridelené rovnaké identifikačné číslo potrebné pre overenie používateľa voči databáze pri prihlasovaní. Pri nedodržaní tejto podmienky boli v databáze vytvárané rôzne záznamy identifikačných prvkov pre rovnaké účty.

3.9 Návrh databázy

Databáza je plne normalizovaná a rozsahy atribútov boli navrhnuté s prihliadnutím na reálne veľkosti uchovávaných dát. Cieľom bolo vytvoriť spoločnú databázu pre Android aplikáciu a webové rozhranie tak, aby ju bolo možné využiť pri produkčnom nasadení v praxi.

Centrálnym prvkom je tabuľka *udaje*, ktorá uchováva dáta o účasti študentov na konkrétnych prednáškach, v konkrétny deň. Primárnym kľúčom tabuľky je atribút *ID_udaje*, je tu tiež trojica cudzích kľúčov *ID_prednaska*, *ID_tyzden* a *ID_uzivatel*, ktoré sú prepojené s tabuľkami *prednaska*, *tyzden* a *uzivatel*, pomocou vzťahu 1:N pre každý atribút. V tabuľke sa nachádza tiež aj údaj o tom, kedy bol záznam do tabuľky vložený v podobe atribútu *vlozene_dna* a dodatočné atribúty *potvrdene* a *bonus_body*, ktoré boli do tabuľky pridané s ohľadom na praktickú využiteľnosť, atribút *potvrdene* značí to, či bola účasť študenta potvrdená vyučujúcim a bonus_body udáva počet bonusových bodov dosiahnutých na danej hodine.

Tabuľka *prednaska* slúži na evidenciu zoznamu prednášok, pre zamedzenie redundancie dát je tvorená z primárneho kľúča *ID_prednaska* a cudzích kľúčov *ID_typ*, *ID_den*, *ID_nazov_prednasky* a *ID_vyucujuci*.

Tabuľka s názvom *uzivatel* obsahuje zoznam používateľov spolu s údajmi o ich študijnom zaradení, teda škole, fakulte, krúžku a ročníku. V tejto tabuľke sa taktiež nachádzajú údaje o prihlasovacom mene a hesle pre lokálne prihlásenie a tiež atribút *idfacebook*, ktorý slúži na zaradenie používateľa prihláseného prostredníctvom facebooku do systému Prihlasovacie heslo pre lokálne prihlásenie je v databáze uložené ako 32 miestny reťazec vzniknutý pomocou zašifrovania pôvodného hesla funkciou md5.

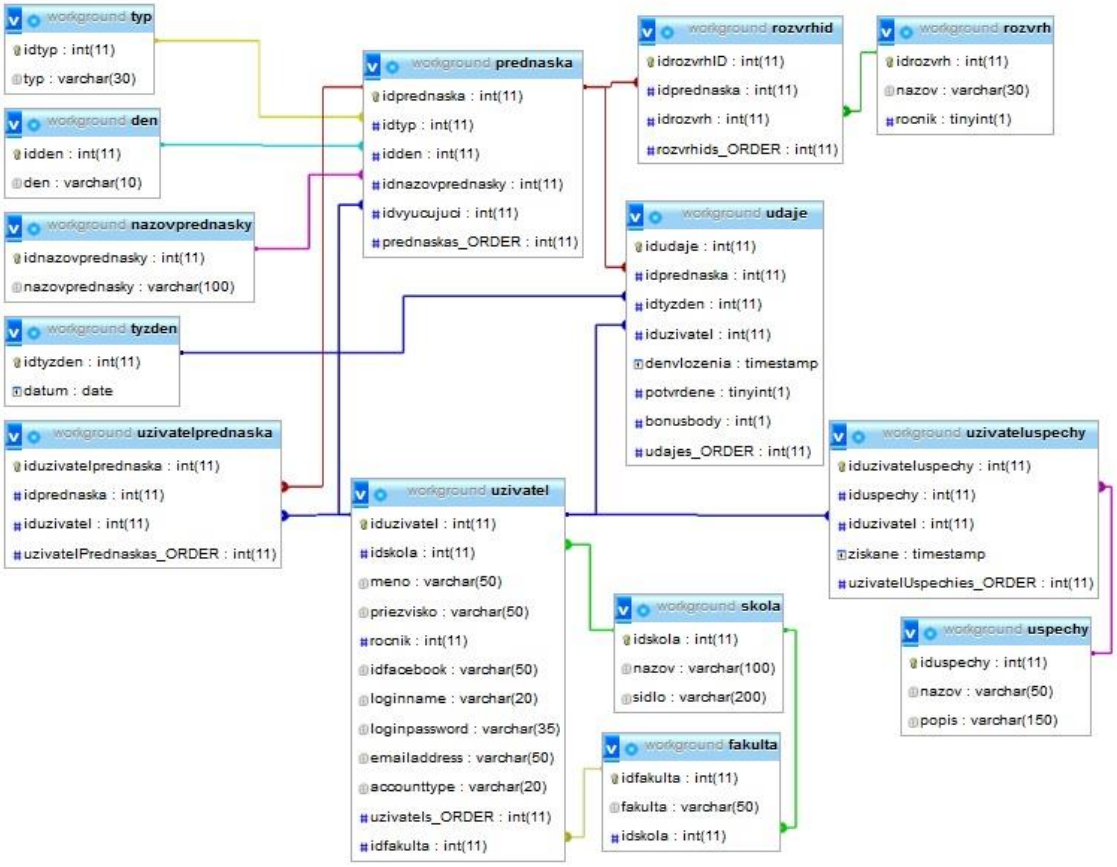
Tabuľka *uzivatel_prednaska* obsahuje dva cudzie kľúče *ID_uzivatel* a *ID_prednaska* a uchováva údaje o zozname prednášok pre všetkých používateľov.

Tabuľka *fakulta* obsahuje primárny kľúč a rovnomenný atribút a tiež cudzí kľúč prepojený na tabuľku *skola*. Táto tabuľka eviduje údaje o tom, aké fakulty sú v systéme a pre ktorú školu sú evidované.

Pri implementácii prvkov gamifikácie bola do databázy pridaná tabuľka *uspechy*, kde sú uložené údaje o úspechoch používateľov a tabuľka uzivatel_uspechy,

ktorá uchováva dosiahnuté úspechy pre všetkých používateľov. Pre zjednodušovanie ukladania predmetov pre používateľov, bola pridaná tabuľka *rozvrh* a *rozvrhID*, kde je možné uložiť viacero prednášok ako jeden výber so spoločným názvom rozvrh a ďalej s ním takto pracovať.

Tabuľky *typ*, *nazov_prednasky*, *den*, *skola* a *tyzden* obsahujú rovnomenné atribúty a primárne kľúče pre uchovávanie konkrétnych hodnôt. Niektoré tabuľky obsahujú atribút s názvom tabuľky a suffixom *ORDER*, tieto atribúty slúžia ako pomocné pri objektovom mapovaní databázy webovou aplikáciou, neuchovávajú však žiadne dáta. Schéma databázy, vzťahy medzi tabuľkami a dátové typy jednotlivých atribútov popisuje obrázok(obr. 22).



Obrázok 28: Návrh databázy

Záver

Výsledkom tejto záverečnej práce je navrhnutý a implementovaný systém pozostávajúci z natívnej mobilnej aplikácie pre operačný systém Android a webového rozhrania implementovaného v programovacom jazyku Java. Tento systému umožňuje hravou formou evidovať účasť študentov na predpísanom výukovom procese. Aplikácia pre Android a webové rozhranie operujú nad spoločnou databázou a poskytujú rozhranie pre rôzne typy používateľov. Ako prínos tohto systému by som označil možnosť motivovať študentov k vyššej účasti na predpísanom študijnom pláne a odbúranie manuálnej evidencie účasti pre vyučujúcich.

Pre vypracovanie tejto práce bolo potrebné zoznámiť sa s princípmi vývoja pre mobilné zariadenia, základnými praktikami a charakteristikami vývoja pre mobilné platformy, osvojiť si princípy návrhu aplikácií a znalosti nevyhnutné na vývoj aplikácií pre operačný systém Android. V neposlednom rade bolo potrebné spoznať spôsob vývoja webových stránok prostredníctvom programovacieho jazyka Java a mapovanie objektov z databázy pomocou programovacej techniky ORM. Nadobudnuté znalosti a schopnosti mi pomohli vytvoriť a implementovať alternatívnu možnosť hravej evidencie dochádzky.

Zoznam použitej literatúry

[1] Šima, Ľ.: Vývoj mobilnej aplikácie využívajúcej prvky gamifikácie: Bakalárska práca. Bratislava. EU FHI 2014. 39 s.

[2] Life Cycle of JSP [online]. 8 október 2013. [cit. 28.3.2016]. Dostupné na internete: <http://www.deepakgaikwad.net/index.php/2009/04/15/jsp-life-cycle.html>

[3] WIKIPEDIA : Gamification [online]. 10. marec 2016. [cit. 7. 03. 2016]. Dostupné na internete: <http://en.wikipedia.org/wiki/Gamification>.

[4] HAMARI, J., KOIVISTO, J., & SARSA, H. (2014). "Does Gamification Work? – A Literature Review of Empirical Studies on Gamification". [Online]. 6.január 2014. [cit. 7. 03. 2016]. Dostupné na internete: <http://www.hiit.fi/u/hamari/2014-hamari_et_al-does_gamification_work.pdf>.

[5] WIKIPEDIA: Android (operačný systém) [online]. 2. marec 2016. [cit. 17. 03. 2016]. Dostupné na internete: <http://sk.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%BD_syst%C3%A9m%29>.

[6] WIKIPEDIA: Android(Operating system) [online]. 13. marec 2016. [cit. 17. 03. 2016]. Dostupné na internete: <http://en.wikipedia.org/wiki/Android_(operating_system)>.

[7] LÜSSI, M: Elefanten und Tiger per Handy retten, 20 Minuten AG, 2009.[online]. 19. júl 2009. [cit. 12. 03. 2016]. Dostupné na internete: <http://www.20min.ch/news/zuerich/story/Elefanten-und-Tiger-per-Handy-retten-30272905>.

[8] WIKIPEDIA: Fitocracy. [online]. 5. august 2015. [cit. 15. 03. 2016]. Dostupné na internete: <http://en.wikipedia.org/wiki/Fitocracy>.

[9] FACEBOOK DEVELOPERS: Facebook SDK [online]. 8. november 2007. [cit. 19. 03. 2016]. Dostupné na internete: <https://developers.facebook.com/docs/android/downloads/>.

[10] THE APACHE SOFTWARE FOUNDATION : Apache License, Version 2.0 [online]. 1. január 2004. [cit. 19. 03. 2016]. Dostupné na internete: <http://www.apache.org/licenses/LICENSE-2.0>.

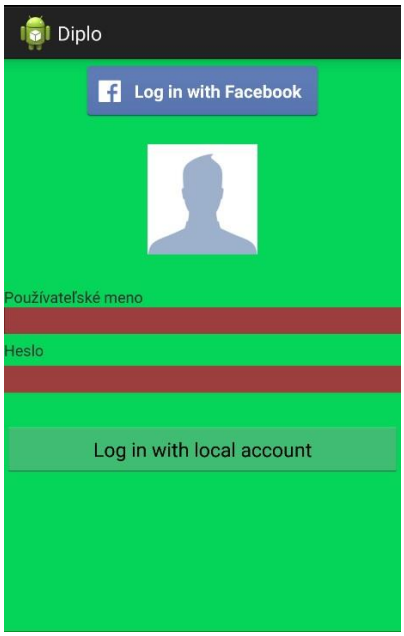
[11] WIKIPEDIA: Java Persistence API [online]. 26. február 2014. [cit. 13.3.2016]. Dostupné na internete: <https://cs.wikipedia.org/wiki/Java_Persistence_API>

[12] WIKIPEDIA: Hibernate [online]. 15. apríl 2015. [cit. 13.3.2016]. Dostupné na internete: <https://cs.wikipedia.org/wiki/Hibernate>

[13] WIKIPEDIA: Apache Tomcat [online]. 23. február 2016. [cit. 13.3.2016]. Dostupné na internete: <https://en.wikipedia.org/wiki/Apache_Tomcat>

Prílohy

Príloha 1 Ukážky obrazoviek Android aplikácie

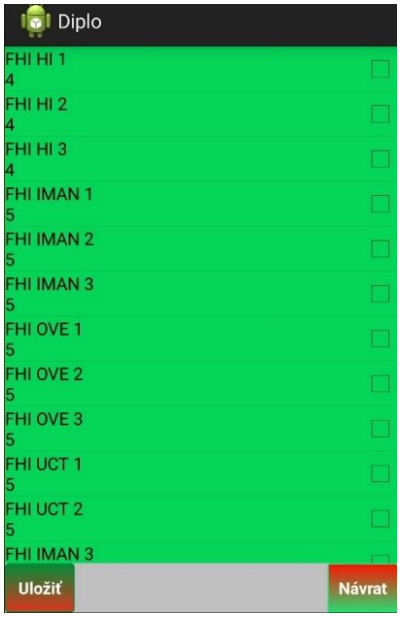


Obrázok 29: Úvodná obrazovka prihlásenia

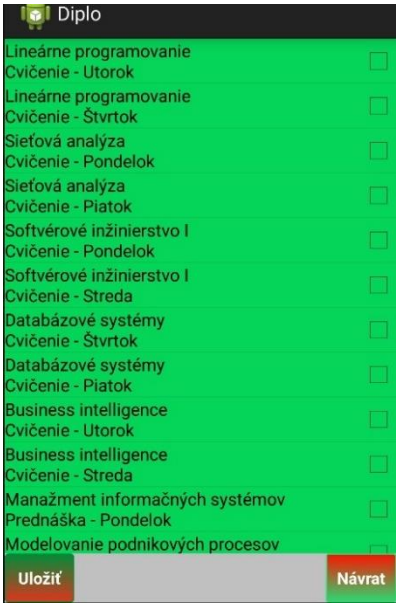
Obrázok 30: Hlavná obrazovka po prihlásení

Obrázok 31: Obrazovka pre výber z menu nastaviť rozvrh

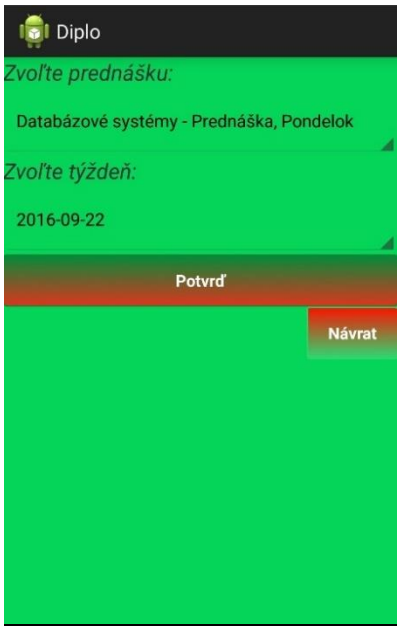
Obrázok 32: Obrazovka zobrazujúca rebríček najlepších piatich študentov podľa dochádzky



Obrázok 33: Obrazovka zobrazujúca možný rozvrh pre pridanie



Obrázok 34: Obrazovka zobrazujúca predmety, ktoré má používateľ uložené



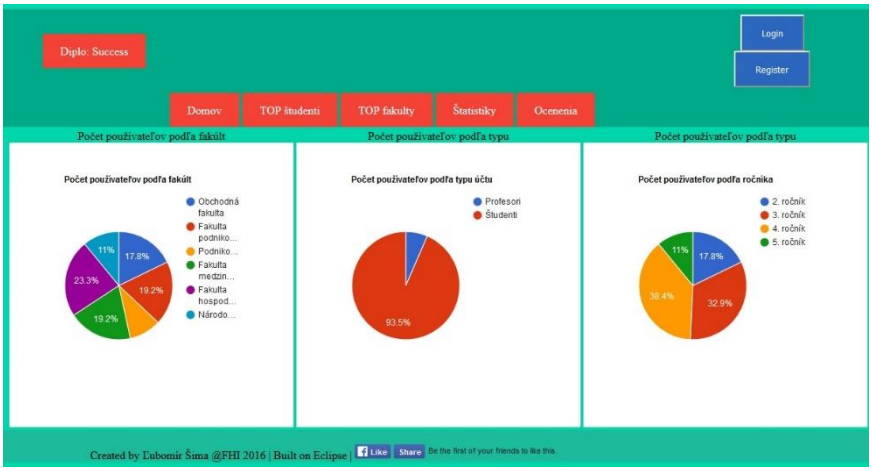
Obrázok 35: Obrazovka pre pridanie účasti na výukovom procese

Obrázok 36: Obrazovka zobrazujúca dosiahnuté úspechy používateľa

Príloha 2 Ukážky stránok webového rozhrania

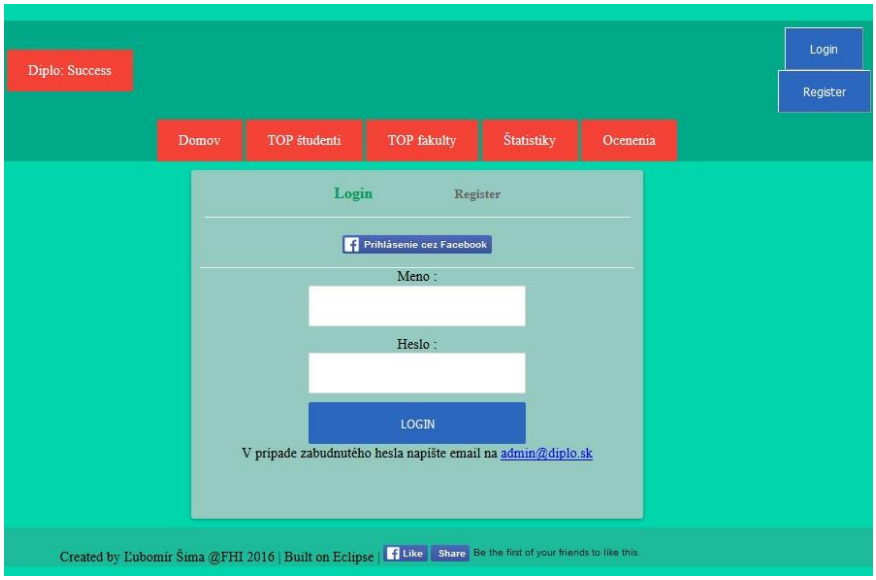
Obrázok 37: Úvodná strana webového rozhrania

Obrázok 38: Rebríček fakúlt

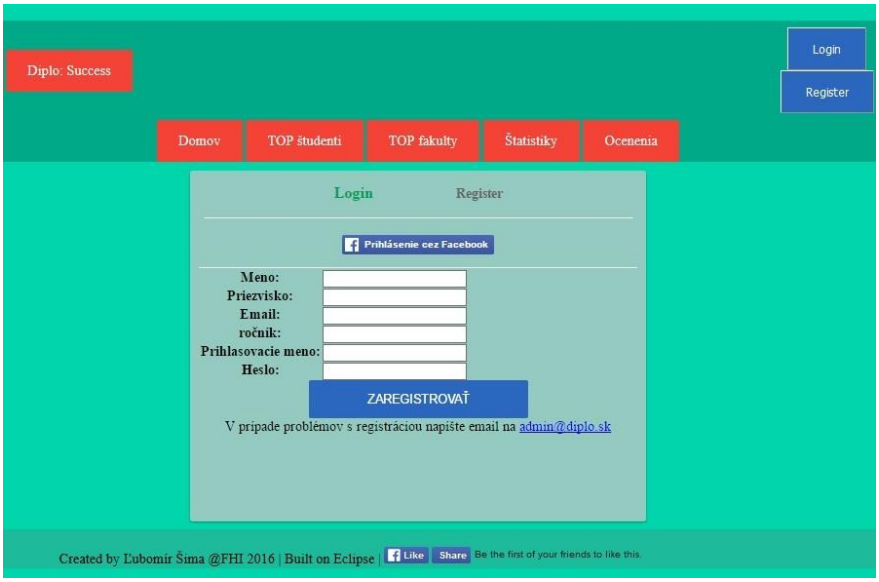


Obrázok 39: Súhrnné info

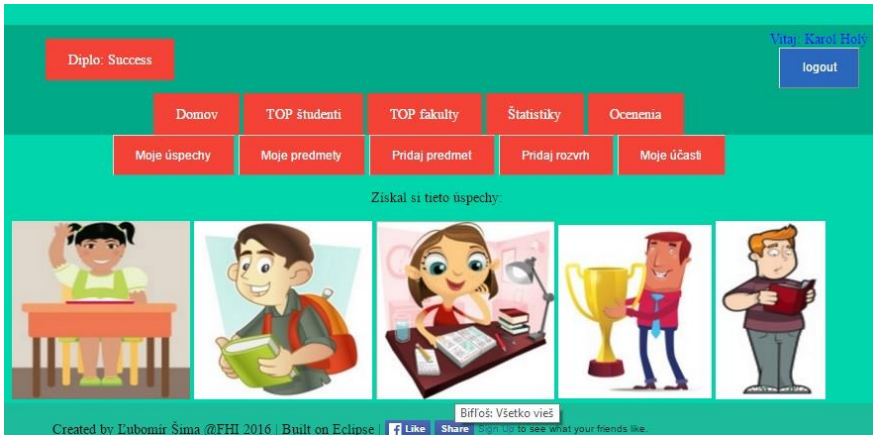
Obrázok 40: Úspechy, ktoré je možné získať



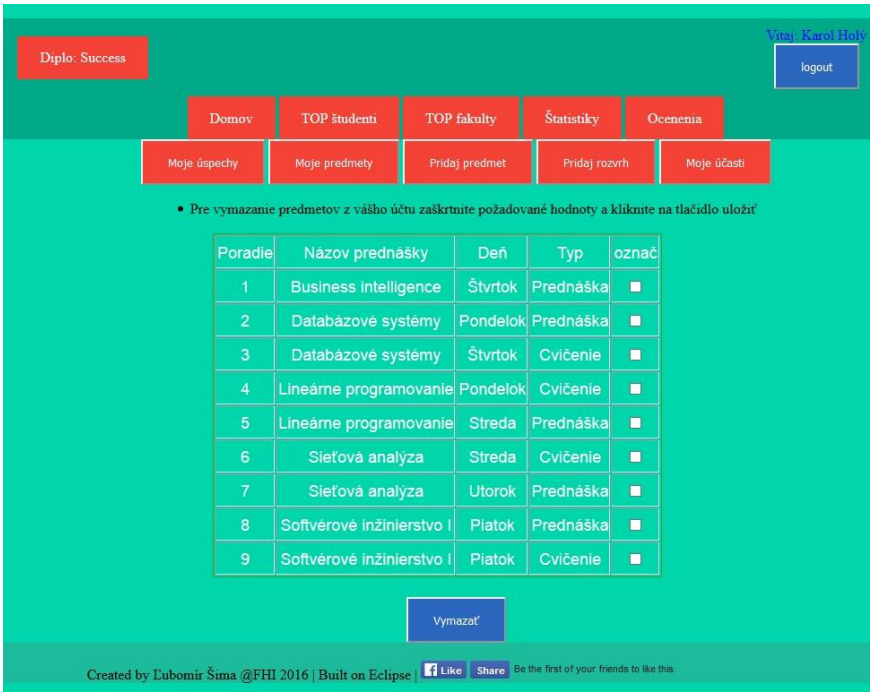
Obrázok 41: Prihlasovacia obrazovka



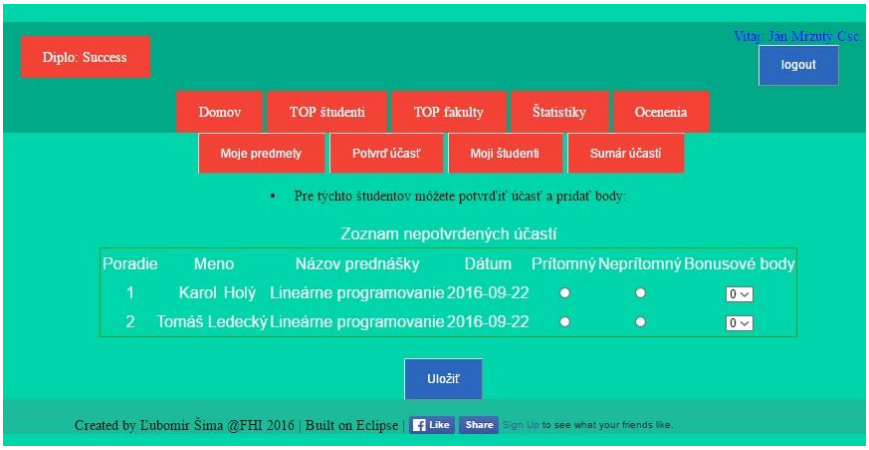
Obrázok 42: Časť: registrácia



Obrázok 43: Úspechy študenta

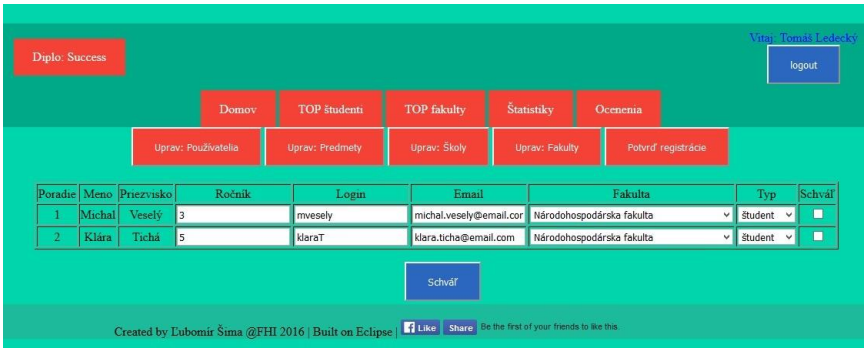


Obrázok 44: Predmety študenta



Obrázok 45: Potvrdenie účasti profesora študentom

Obrázok 46: Sumárny zoznam vyučovaných študentov



Obrázok 47: Schvaľovanie účtov administrátorom

Obrázok 48: Rebríčky TOP študentov