# Technická dokumentácia pre modul UserActivity.Client

Verzia 19.04.2016

*Tabuľka 1. Autori*

| Autor | Rola |
|---|---|
| Peter Halaš | Autor |
| Maroš Grošaft | Autor |

*Tabuľka 2. História zmien*

| Verzia | Dátum | Autor | Popis |
|---|---|---|---|
| 1.0 | 25.11.2015 | Peter Halaš | Vytvorenie dokumentu |
| 1.1 | 04.04.2016 | Peter Halaš | Doplnenie dokumentu |
| 1.2 | 12.04.2016 | Peter Halaš | Doplnenie dokumentu |
| 1.3 | 19.04.2016 | Maroš Grošaft | Doplnenie analýzy a častí 3.4 – 3.7 a 4.4 – 4.7 |
| 1.4 | 13.05.2016 | Maroš Grošaft | Úprava častí 3.4, 3.6 a 4.4, 4.6 |

## Obsah

# 1 Úvod

V dokumente sú opísané zmeny modulu UserActivity.Client, v rámci tímového projektu DevActs.

# 2 Analýza

Modul UserActivity.Client predstavuje aplikáciu , ktorá je spustená u programátora a zaznamenáva jeho aktivitu. Udalosti si ukladá do databázy a následne odosiela do modulu UserActivity.Web.

Momentálne sa aplikácia pri odosielaní údajov neautorizuje voči UserActivity.Web. Je potrebné doplniť hlavičku *Authorization header*.

Pri neúspešnom odoslaní Eventov na server UACA, nie je používateľ nijakým spôsobom upozornení o neúspešnom odoslaní dát. Naviac klient sa pokúša odoslať všetky Eventy, ktoré mu prišli v dávke (toto po knzultácii má zostať).

Ďalším problémom, na ktorý sme narazili je, že pri zmene používateľského mena sa nezmení hodnota atribútu Eventu „user“. Samo o sebe to nie je problém, avšak UACA server pri prijatí správy porovnáva meno v atribúte Eventu „user“ s menom v autorizačnej hlavičky, ktorá sa vytvára na základe aktuálneho mena nastaveného používateľom.

Pre odosielanie Eventov zachytených aplikáciou je potrebná adresa, na ktorú sa majú Eventy posielať. Táto adresa je momentálne uložená pri inštalácii do registrov. Adresu by však bolo lepšie načítavať priamo zo servera.

Po štarte aplikácie, ako aj počas jej behu je potrebné overovanie dostupnosti novej verzie.

Nakoľko vizuálna stránka aplikácie nezodpovedá jej kvalite, je potrebné prerobenie grafického rozhrania.

# 3 Návrh riešenia

V nasledujúcej časti bude opísané, ako boli navrhnuté jednotlivé úpravy, ktoré boli opísané v časti analýza.

## 3.1 Doplnenie autorizácie do Put requestu

Na pridanie už spomínanej *Authorization header* je potrebné získať používateľove meno a heslo a následne ho zakódovať do Base64.

Ak volaná Rest služba vráti odpoveď, že sa nepodarilo autorizovať používateľa, používateľ musí znovu zadať svoje meno a heslo.

## 3.2 Doplnenie messageBoxu

Na upozornenie používateľa o neúspešnom odoslaní je vhodné použiť messageBox, ktorý bude informovať používateľa. Tento messageBox by mal byť zobrazený len určitú dobu.

## 3.3 Doplnenie premenovania hodnoty atribútu „user“

Existujú dva prípady, kedy je potrebné vykonať premenovanie všetkých udalostí. Prvým je zmena používateľovho mena a druhým je spustenie aplikácie. Keďže používateľské meno sa zadáva už pri inštalácii.

## 3.4 Získavanie adresy pre odosielanie Eventov

Pre odosielanie Eventov je potrebná adresa, na ktorú sa majú odosielať. Nakoľko adresa sa získava z registrov a my potrebujeme aby sa adresa získavala zo servera, potrebujeme nadviazať spojenie so serverom a následne pomocou volania služby získať adresu, na ktorú sa majú Eventy odosielať. Adresa sa bude získavať vždy pri zmene adresy na server v okne nastavení a pred odosielaním eventov. Adresa bude vždy zapísaná do registrov.

## 3.5 Overovanie dostupnosti novej verzii

Pre overovanie novej verzie je potrebné zistiť zo serveru najnovšiu dostupnú verziu aplikácie User Activity. Zistenú verziu porovnáme s aktuálne nainštalovanou verziou, ktorej hodnota je uložená v registroch.

## 3.6 Stiahnutie novej verzie, inštalácia a vymazanie inštalačného súboru

Ak je dostupná nová verzia aplikácie, je potrebné stiahnutie a inštalácia. Zo serveru zistíme adresu z ktorej môžeme stiahnuť najnovší inštalačný balíček. Používateľovi sa zobrazí okno, ktoré mu ponúkne stiahnutie a inštaláciu novej verzie. Ak má používateľ záujem o inštaláciu, tá sa po jeho potvrdení automaticky spustí. Po stiahnutí balíčku sa do registrov zaznamená, kde sa balíček uložil. Pri najbližšom štarte aplikácie sa stiahnutý balíček a jeho rozbalená verzia vymažú.

## 3.7 Zmena grafického rozhrania aplikácie

Pre zmenu grafického rozhrania je potrebné kompletné prerobenie rozloženia komponentov jednotlivých okien, ako aj dizajnové úpravy grafického rozhrania. Je potrebné vytvorenie nových tlačidiel, a celkového zobrazovania okien.

# 4 Implementácia

Implementácia navrhnutej funkcionality bola realizovaná podľa návrhu v nasledujúcej kapitole bude detailnejšie opísané niektoré dôležité implementačné detaily a rozhodnutia, ktoré k nim viedli.

## 4.1 Implementácia autorizácie Put requestu

Implementácia prebehla úspešne bez komplikácií. Metóda na poslanie udalosti vyzerá nasledovne:

public void commitEvent(CachedEvent cachedEvent, String userNameAndPassword)

Teda vyžaduje aby sa posielalo spojené meno a heslo používateľa vo formáte **meno + „:“ + heslo.**

Samotné pridanie hlavičky je implementované takto:

Response response = fullTarget.request().header(authorizationHeaderName, authorizationHeaderValue).put(Entity.json(cachedEvent.getData()));

Pred odoslaním requestu sa vloží *Basic Authorization header* spolu so zakódovaným menom a heslom v požadovanom tvare.

## 4.2 Implementácia messageBoxu

Rozhodol som sa doplniť do triedy *MessageBox*, ktorá poskytuje statické metódy na zobrazenie rôznych info, warning a error správ pridať metódu *showDisapearringBox*. Táto metóda na vstupe vyžaduje správu, ktorú má zobraziť a názov dialógu. Metóda používa statickú premennú DELAY, ktorá je nastavená na 10 s.

## 4.3 Implementácia premenovania atribútu „user“ v eventoch

Implementácia bola veľmi jednoduchá, keďže trieda *EventCache* už má implementovanú metódu *updateUserNameInAllEvents*, ktorá vykonáva presne to čo je potrebné. V prvom prípade sa pridalo volanie tejto metódy do triedy *LoginActionListener*, ktorá sa nachádza v súbore *LoginDialog* pod triedou s rovnakým názvom. Konkrétne po úspešnom overení používateľovho mena a hesla voči Administračnému portálu. V druhom prípade sa táto metóda volá v triede *App*, v metóde *initCore*, čo je metóda kde sa vytvára aj prvotné spojenie s databázou.

## 4.4 Implementácia získavania adresy pre odosielanie Eventov

Adresu, na ktorú sa majú odosielať eventy získavame v dvoch prípadoch:

1. Keď používateľ zmení adresu na server v okne nastavení.
2. Pred odosielaním eventov.

V oboch prípadoch získame adresu pomocou metódy *setSvcUrl()* v triede *Settings.java,* ktorá získa a zapíše adresu, na ktorú sa majú odosielať Eventy do registrov. Táto metóda ďalej využíva tiež novovytvorenú metódu *getValueFromService(String modulName, String keyName)* v triede *Settings.java,* pomocou ktorej najprv získame adresu na server z registrov pomocou metódy *getServerUrl()* triedy *Settings.java* a následne voláme službu na serveri, ktorá nám na základe modulName a keyName vráti odpoveď. Na získavanie odpovede sme doplnili triedu *SimpleHttpCaller.java* o metódu *getJSON()*. Z odpovede vo formáte JSon získavame adresu, na ktorú bude aplikácia posielať Eventy.

## 4.5 Implementácia overovania dostupnosti novej verzie

Pri štarte aplikácie sa automaticky kontroluje dostupnosť novej verzie. Za pomoci metódy *initInstallGui()* v triede *App.java* sa overí nová verzia. Na overenie sa používa metóda *newVersion()* z triedy *App.java*. Metóda *newVersion()* volá metódu *checkNewVersion()* z triedy *Settings.java,* v ktorej sa pomocou metódy *getValueFromService(String modulName, String keyName)* získa najnovšia dostupná verzia uložená na serveri a adresa pre stiahnutie inštalačného balíčka. Zistená verzia sa pomocou metódy *compareVersion(String version1, String version2)* porovná s aktuálne nainštalovanou verziou. Ak je dostupná nová verzia, adresa na stiahnutie inštalačného balíčka sa vráti postupne ako návratová hodnota metóde *newVersion()*. Nová verzia sa takisto kontroluje aj každých 12 hodín pomocou triedy *UpdaterThread.java*, ktorej inštancia beží počas celej doby programu v samostatnom vlákne.

Všetky spomínané metódy (okrem *getValueFromService(String modulName, String keyName)*) boli vytvorené pre účel overovania dostupnosti novej verzie.

## 4.6 Implementácia stiahnutia novej verzie, inštalácia a vymazanie inštalačného balíčka

Ak je dostupná nová verzia aplikácie, automaticky sa vytvorí a zobrazí inštalačné okno (trieda *InstallWindow.java*). V okne sa zobrazia dve možnosti. Stiahnutie a inštalácia novej verzie – pomocou *metódy createDownloadWProgressBarThread()* sa vytvorí nové vlákno a zaháji sa sťahovanie inštalačného balíčka. Po stiahnutí sa balíček vo formáte .zip rozbalí pomocou metódy *unZipFile(String zipFile, String outputFolder)* triedy *VersionControl.java*. Následne sa pomocou

metódy *installNewVersion(String msiInstallFile)* triedy *VersionControl.java* zaháji inštalácia novej verzie User Activity a ostatných súčastí inštalačného balíčka (pluginy User Activity). Druhou možnosťou na inštalačnom okne je stiahnutie a inštaláciu odmietnuť – inštalačné okno sa zavrie.

Všetky spomínané triedy a metódy boli vytvorené pre účel stiahnutia a inštalácie novej verzie aplikácie a jej súčastí.

Po stiahnutí inštalačného balíčka sa do registrov pomocou metódy *unZipFile(String zipFile, String outputFolder)* triedy *VersionControl.java*, ktorá využíva metódu *setDownloadedFilePath(String downloadedFilePath)* triedy *Settings.java* zapíše cesta, kde je balíček uložený a rozbalený. Pri najbližšom štarte aplikácie sa overí pomocou metódy *deleteDownloadedFiles()*, či je v registroch uložená cesta stiahnutého balíčku. Ak je v registroch zapísané niečo odlišné ako prázdny reťazec (je tam zapísaná cesta), vymažú sa stiahnutý a rozbalený balíček pre inštaláciu a do registrov sa pomocou metódy *setDownloadedFilePath(String downloadedFilePath)* zapíše prázdny reťazec, aby sa nasledovnom štarte aplikácie nepokúšala aplikácia vymazať súbory opätovne.

## 4.7 Implementácia zmeny grafického rozhrania

Pre zmenu grafického rozhrania boli upravované už existujúce triedy. Bola vytvorená trieda *UIConstants.java*, v ktorej sú uložené konštanty zdieľaných grafických prvkov. Ďalej boli doimplementované triedy *InstallWindow.java*, *TranslucentButton.java*, ktorá vytvára základné polo transparentné tlačidlá pre aplikáciu. Pre základnú obrazovku boli vytvorené špeciálne tlačidlá pomocou triedy *MainWindow.java* a metódy *addButtonWImage(...)*. Všetky okná aplikácie majú doimplementované pozadie, na základe čoho sme ostatné komponenty aplikácie museli ošetriť knižničnou metódou *setOpaque(false)* – táto metóda zabezpečuje aby bolo cez komponent viditeľné pozadie komponentu na ktorom leží.

# 5 Testovanie

Kvôli existujúcej implementácii volania Rest služby nebolo možné napísať jednotkové testy.