

Sledovanie pohľadu pri používaní aplikácií

Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Róbert Móro

Členovia tímu: Bc. Dominika Červeňová, Bc. Jakub Daráž, Bc. Lukáš Gregorovič,
Bc. Martin Janík, Bc. Róbert Kocian, Bc. Michal Mészáros, Bc. Kristína Mišíková

Akademický rok: 2013/2014

Obsah

1 Úvod.....	1-1
2 Globálne ciele projektu na zimný semester.....	2-1
3 Plán na letný semester	3-1
3.1 Dlhodobý plán	3-1
3.1.1 Anotovať prúd dát	3-1
3.1.2 Správa zariadení	3-1
3.1.3 Poskytnutie API.....	3-1
3.1.4 Export získaných dát	3-1
3.1.5 Zaznamenávanie videa	3-1
3.1.6 Vizualizácia získaných dát	3-2
3.2 Krátkodobý podrobný plán.....	3-2
3.2.1 Prvý šprint - 1. a 2. týždeň	3-2
3.2.2 Druhý šprint - 3. a 4. týždeň	3-2
3.2.3 Tretí šprint - 5. a 6. týždeň	3-2
4 Opis prototypu.....	4-1
4.1 Architektúra	4-1
4.2 Funkcionálne požiadavky	4-2
4.2.1 Desktopová aplikácia	4-2
4.2.2 Webová aplikácia	4-2
4.2.3 Rozšírenie pre webový prehliadač.....	4-3
4.3 Dátový model	4-3
4.3.1 Opis entít.....	4-5
5 ACDC - 1. šprint	5-1
5.1 Identifikácia elementu na stránke.....	5-1
5.1.1 User story	5-1
5.1.2 Úloha.....	5-1
5.1.3 Analýza	5-1
5.1.4 Implementácia.....	5-1
5.1.5 Testovanie	5-2
5.2 Vytvorenie Add-onu pre browser.....	5-2
5.2.1 Úloha.....	5-2
5.2.2 Implementácia.....	5-3
5.3 Kalibrácia zariadenia	5-3
5.3.1 User story	5-3

Obsah

5.3.2 Úloha.....	5-3
5.3.3 Analýza	5-3
5.3.4 Implementácia.....	5-3
5.3.5 Testovanie	5-3
5.4 Autentifikácia a autorizácia	5-4
5.4.1 User story	5-4
5.4.2 Úloha.....	5-4
5.4.3 Implementácia.....	5-4
5.4.4 Testovanie	5-4
5.5 Analýza typov anotácií	5-4
5.5.1 User story	5-4
5.5.2 Úloha.....	5-4
5.5.3 Analýza	5-4
5.6 Komunikácia pluginu s desktopovou aplikáciou.....	5-5
5.6.1 User story	5-5
5.6.2 Úloha.....	5-5
5.6.3 Analýza	5-5
5.7 Ďalšie úlohy na ktorých sa pracovalo počas šprintu.....	5-6
5.7.1 Komunikácia so zariadením.....	5-6
5.7.2 Analýza prehliadačov Firefox a Chrome	5-6
5.7.3 Analýza a porovnanie socketov a RESTu	5-7
6 Beatles - 2. šprint	6-1
6.1 Identifikácia elementov na základe súradnice	6-1
6.1.1 User story	6-1
6.1.2 Úloha.....	6-1
6.1.3 Analýza	6-1
6.1.4 Implementácia.....	6-1
6.1.5 Testovanie	6-1
6.2 Používateľ sa chce prihlásiť (autentifikovať)	6-2
6.2.1 User story	6-2
6.2.2 Úloha.....	6-2
6.2.3 Analýza	6-2
6.2.4 Implementácia.....	6-2
6.2.5 Testovanie	6-3
6.3 Prepočítavanie súradníc vzhľadom na aktívne okno	6-3
6.3.1 User story	6-3
6.3.2 Úloha.....	6-3

Obsah

6.3.3	Analýza	6-4
6.3.4	Implementácia.....	6-4
6.3.5	Testovanie	6-4
6.4	Experimentátor chce vidieť možné oblasti dynamicky pri hoveri.....	6-4
6.4.1	User story	6-4
6.4.2	Úloha.....	6-4
6.4.3	Implementácia zvýraznenia	6-5
6.4.4	Implementácie tooltipu	6-5
6.4.5	Testovanie	6-6
6.5	Simulácia Eye-trackera z pohybu myši	6-7
6.5.1	User story	6-7
6.5.2	Úloha.....	6-7
6.5.3	Analýza	6-7
6.5.4	Návrh	6-7
6.5.5	Implementácia.....	6-7
6.5.6	Testovanie	6-8
6.6	Prihlásenie ako REST.....	6-8
6.6.1	Úloha.....	6-8
6.6.2	Analýza	6-8
6.6.3	Implementácia.....	6-8
6.6.4	Testovanie	6-8
6.7	Vytvorenie nového používateľa	6-9
6.7.1	User story	6-9
6.7.2	Analýza	6-9
6.7.3	Implementácia.....	6-9
6.7.4	Testovanie	6-9
6.8	Zadefinovanie rolí.....	6-9
6.8.1	User story	6-9
6.8.2	Analýza	6-9
6.8.3	Implementácia.....	6-9
6.8.4	Testovanie	6-10
6.9	Zadefinovanie formy protokolu pre odosielanie dát na server	6-10
6.9.1	Úloha	6-10
6.9.2	Implementácia.....	6-10
6.10	Odosielanie dát na server	6-11
6.10.1	User story.....	6-11
6.10.2	Úloha	6-11

Obsah

6.10.3 Implementácia	6-12
6.10.4 Testovanie	6-12
6.11 Vytvorenie projektu	6-13
6.11.1 User story	6-13
6.11.2 Úloha	6-13
6.11.3 Analýza.....	6-13
6.11.4 Implementácia	6-13
6.11.5 Testovanie	6-13
6.12 Vytvorenie sedenia v rámci projektu	6-14
6.12.1 User story	6-14
6.12.2 Úloha	6-14
6.12.3 Analýza.....	6-14
6.12.4 Implementácia	6-14
6.12.5 Testovanie	6-15
6.13 Ďalšie úlohy na ktorých sa pracovalo počas šprintu	6-15
6.13.1 Analýza možných riešení zvýraznenia elementov na stránke	6-15
6.13.2 Analýza knižnice Opentip.....	6-16
6.13.3 Analýza Bootstrap-u.....	6-17
6.13.4 Analýza surových packetov z Eye-trackera	6-18
7 Coldplay - 3. šprint	7-1
7.1 Prihlásenie používateľa cez LDAP.....	7-1
7.1.1 User story	7-1
7.1.2 Úloha.....	7-1
7.1.3 Implementácia.....	7-1
7.1.4 Testovanie	7-2
7.2 Podpora pre usporiadanie, filtrovanie a pagináciu dát v tabuľke.....	7-2
7.2.1 User story	7-2
7.2.2 Úloha.....	7-2
7.2.3 Analýza	7-2
7.2.4 Implementácia.....	7-3
7.2.5 Testovanie	7-3
7.3 Komunikácia add-onu s klientom na desktope	7-3
7.3.1 User story	7-3
7.3.2 Úloha.....	7-3
7.3.3 Analýza	7-4
7.3.4 Implementácia.....	7-4
7.3.5 Testovanie	7-4

Obsah

7.4 Adaptér pre klienta	7-4
7.4.1 User story	7-4
7.4.2 Úloha.....	7-4
7.4.3 Analýza	7-4
7.4.4 Návrh	7-5
7.4.5 Implementácia.....	7-6
7.4.6 Testovanie	7-6
7.5 Simulácia Eye-trackera z pohybu myši	7-6
7.5.1 User story	7-6
7.5.2 Úloha.....	7-7
7.5.3 Analýza	7-7
7.5.4 Návrh	7-7
7.5.5 Implementácia.....	7-7
7.5.6 Testovanie	7-7
7.6 Manažment používateľov CRUD	7-8
7.6.1 User story	7-8
7.6.2 Analýza	7-8
7.6.3 Implementácia.....	7-8
7.6.4 Testovanie	7-8
7.7 Import používateľov z CSV súboru do databázy.....	7-8
7.7.1 User story	7-8
7.7.2 Úloha.....	7-8
7.7.3 Implementácia.....	7-8
7.7.4 Testovanie	7-9
7.8 Pridanie používateľa k projektu	7-10
7.8.1 User story	7-10
7.8.2 Úloha.....	7-10
7.8.3 Analýza	7-10
7.8.4 Implementácia.....	7-10
7.8.5 Testovanie	7-11
7.9 Pridanie používateľa k sedeniu.....	7-11
7.9.1 User story	7-11
7.9.2 Úloha.....	7-11
7.9.3 Analýza	7-12
7.9.4 Implementácia.....	7-12
7.9.5 Testovanie	7-12
8 Deep Purple - 4. šprint	8-1

Obsah

8.1 Komunikácia s browser addonom na zistenie elementov.....	8-1
8.1.1 User story	8-1
8.1.2 Úloha.....	8-1
8.1.3 Analýza	8-1
8.1.4 Implementácia.....	8-1
8.1.5 Testovanie	8-1
8.2 Adaptér pre Klienta.....	8-2
8.2.1 User story	8-2
8.2.2 Úloha.....	8-2
8.2.3 Analýza	8-2
8.2.4 Implementácia.....	8-2
8.2.5 Testovanie	8-3
8.3 Vytvorenie nového používateľa	8-3
8.3.1 User story	8-3
8.3.2 Úloha.....	8-3
8.3.3 Analýza	8-3
8.3.4 Implementácia.....	8-3
8.3.5 Testovanie	8-4
8.4 Autorizácia používateľa	8-4
8.4.1 User story	8-4
8.4.2 Úloha.....	8-4
8.4.3 Analýza	8-4
8.4.4 Implementácia.....	8-4
8.4.5 Testovanie	8-4
8.5 Zmena stavu sedenia	8-5
8.5.1 User story	8-5
8.5.2 Úloha.....	8-5
8.5.3 Analýza	8-5
8.5.4 Implementácia.....	8-5
8.5.5 Testovanie	8-5
8.6 Ukladanie dát zo sledovania pohľadu	8-5
8.6.1 User Story	8-5
8.6.2 Úloha.....	8-5
8.6.3 Analýza	8-6
8.6.4 Implementácia.....	8-6
8.6.5 Testovanie	8-6
8.7 Pridanie používateľa k projektu.....	8-7

Obsah

8.7.1 User story	8-7
8.7.2 Úloha.....	8-7
8.7.3 Analýza	8-7
8.7.4 Implementácia.....	8-7
8.7.5 Testovanie	8-8
8.8 Pridanie používateľa k sedeniu	8-8
8.8.1 User story	8-8
8.8.2 Úloha.....	8-8
8.8.3 Analýza	8-8
8.8.4 Implementácia.....	8-8
8.8.5 Testovanie	8-8
8.9 Prihlásenie používateľa cez LDAP.....	8-9
8.9.1 User story	8-9
8.9.2 Úloha.....	8-9
8.9.3 Implementácia.....	8-9
8.9.4 Testovanie	8-10
8.10 Ukladanie a načítanie oblastí záujmu	8-11
8.10.1 Úloha	8-11
8.10.2 Implementácia	8-11
8.10.3 Testovanie	8-11
8.11 Podpora pre usporiadanie, filtrovanie a stránkovanie dát v tabuľke.....	8-12
8.11.1 Úloha	8-12
8.11.2 Implementácia	8-12
8.12 Správa oblasti záujmu.....	8-12
8.12.1 User story.....	8-12
8.12.2 Úloha	8-12
8.12.3 Implementácia	8-12
8.12.4 Testovanie	8-12
8.13 Zadefinovanie oblastí záujmu a odoslanie na server	8-13
8.13.1 User story.....	8-13
8.13.2 Úloha	8-13
8.13.3 Implementácia	8-13
8.13.4 Testovanie	8-13
9 Evanescence - 5. Šprint	9-1
9.1 Logovanie	9-1
9.1.1 User story	9-1
9.1.2 Úloha.....	9-1

Obsah

9.1.3 Analýza	9-1
9.1.4 Implementácia.....	9-1
9.1.5 Testovanie	9-1
9.2 Autentifikácia REST volaní.....	9-1
9.2.1 User story	9-1
9.2.2 Úloha.....	9-1
9.2.3 Implementácia.....	9-1
9.2.4 Testovanie	9-1
9.3 Pridanie používateľa k projektu.....	9-2
9.3.1 User story	9-2
9.3.2 Úloha.....	9-2
9.3.3 Analýza	9-2
9.3.4 Implementácia.....	9-2
9.3.5 Testovanie	9-2
9.4 Ukladanie a načítanie oblastí záujmu pre konkrétny projekt a sedenie	9-2
9.4.1 User story	9-2
9.4.2 Úloha.....	9-3
9.4.3 Implementácia.....	9-3
9.4.4 Testovanie	9-3
9.5 Prihlásenie sa cez browser add-on	9-4
9.5.1 User story	9-4
9.5.2 Úloha.....	9-4
9.5.3 Implementácia.....	9-4
PRÍLOHA A - Výstupy analýz.....	A-1
A.1 ACDC - 1. šprint	A-1
A.1.1 Formát dát zo zariadenia	A-1
A.1.2 Rest Vs Socket.....	A-6
A.2 Beatles - 2. šprint.....	A-8
A.2.1 WinApi funkcie	A-8

1 Úvod

Overenie použiteľnosti navrhnutých používateľských rozhraní je zložitý proces a často nám nestačia údaje len z pohybu myši, či z klikania. Dôvod je jednoduchý: tieto údaje samé o sebe nedokážu spoľahlivo odpovedať na otázku, či používateľ na niečo nekliká preto, lebo to nepovažuje za relevantné, alebo skôr preto, že to nevidí. Práve tu si nachádza uplatnenie sledovanie pohľadu.

Existujúce riešenia si väčšinou nedokážu poradiť so zberom dát z viacerých zariadení naraz. Okrem toho poskytujú len obmedzenú podporu pre testovanie dynamických webových aplikácií. Vzniká teda potreba pre vytvorenie platformy pre zber, spracovanie a vyhodnocovanie údajov zo sledovania pohľadu z viacerých zariadení (pracovných staníc).

V rámci fakulty sa plánujú zriadiť laboratóriá UX Lab a UX Class, kde bude viacero zariadení na sledovanie pohľadu a iné senzory, ktoré budú odosielať dáta na server, kde sa budú ďalej analyzovať.

Náš projekt veľmi úzko súvisí s testovaním aplikácií v týchto laboratóriách, konkrétne so zariadeniami na sledovanie pohľadu.

V tomto dokumente sa venujeme cieľom nášho projektu a realizáciám úloh, potrebných pre naplnenie týchto cieľov. V druhej kapitole sú spísané globálne ciele nášho projektu pre zimný semester. Nakoľko sme sa pri vývoji nášho projektu riadili metodológiou scrum-u, jednotlivé úlohy, a ich realizácie, patria konkrétnym šprintom. Týmto šprintom sa v dokumente venujú jednotlivé kapitoly.

2 Globálne ciele projektu na zimný semester

Organizácia a riadenie experimentov použiteľnosti

Hlavným cieľom nášho projektu je umožniť organizáciu experimentu, jeho vytvorenie, používateľsky jednoduché nastavenie, sledovanie a jeho centrálné riadenie.

Sledovanie pohľadu používateľa pri dynamických webových aplikáciách

Pri riešení sa zameriavame na testovanie dynamických webových aplikácií a chceme vytvoriť riešenie, ktoré nevyžaduje žiaden zásah do nich. Vychádzame pritom zo základného prípadu použitia, kedy si experimentátor nastaví takzvané oblasti záujmu, pri ktorých ho zaujíma, či sa do nich používateľ pozerá, alebo nie, v rámci jeho testovanej webovej aplikácie.

Vytvorenie funkčného celku, ktorý umožní zber dát zo sledovania pohľadu používateľa

V zimnom semestri chceme spojiť autentifikáciu a autorizáciu používateľa, vytváranie projektov a sedení (experimentov) ku ktorým si výskumníci môžu pridávať iných používateľov. Ďalej je našim cieľom, aby po zimnom semestri desktopová časť aplikácie komunikovala s webovým prehliadačom, ako aj funkčná kalibrácia, napojenie na zariadenie, sledujúce pohľad a zber dát z neho.

3 Plán na letný semester

Táto časť obsahuje náš predbežný plán na letný semester. Obsahuje dlhodobý plán, ktorý obsahuje, čo všetko by sme chceli počas letného semestra dosiahnuť. Taktiež obsahuje krátkodobý ale podrobný plán na prvé 3 šprinty, kde je podrobnejšie napísané, čomu by sme sa v tomto období chceli venovať.

3.1 Dlhodobý plán

3.1.1 Anotovať prúd dát

Jedným z našich hlavných cieľov je anotovať prúd dát, ktoré budeme získavať zo sledovania pohľadu pomocou zariadenia. Následne pomocou addonu budeme anotovať tieto dáta v závislosti k oblastiam záujmu. Nad typmi anotácií sme už premýšľali a napadli nás nasledovné možnosti:

- či bola oblasť záujmu navštívená
- počet navštívení
- dĺžka pohľadu na oblasť záujmu
- či bola oblasť záujmu navštívená po zvýraznení prvku

3.1.2 Správa zariadení

Pre výskumníka je dôležité sledovať priebeh sedenia. K tomu by mu mala napomáhať funkcionálnosť pre sledovanie zariadení. Takto by mal výskumník zistiť, či dáta od testera získal, či je možné, že získané dáta môžu byť nespoľahlivé kvôli zlej kalibrácii zariadenia. Táto funkcionálnosť by mala zabezpečiť úspešný priebeh sedení.

3.1.3 Poskytnutie API

Najdôležitejším cieľom na letný semester bude poskytnúť API pre výučbové systémy. Najskôr sa zameriame na výučbový systém ALEF aby bolo možné využívať naše riešenie pre pozorovanie spôsobu práce študentov FIIT.

3.1.4 Export získaných dát

Výskumníci si často krát chcú získané dáta analyzovať na vlastných pracovných zariadeniach, s vlastnými nástrojmi na analýzu dát v pohodlí domova. Preto by sme chceli tiež zabezpečiť, že si výskumníci budú môcť dáta exportovať do rôznych formátov.

3.1.5 Zaznamenávanie videa

Video záznam je veľmi užitočnou formou dát, ktorá vie zvýšiť presnosť analýz vykonaných nad získanými dátami. Je možné tiež zachytiť rôzne vlastnosti prostredia alebo situácie ako osvetlenie prostredia, vzdialenosť testera, dôvod prečo neboli zaznamenané oči zariadením, atď... Zaznamenávanie videa a posielanie video záznamu v reálnom čase je náročná záležitosť, napriek tomu sa ju pokúsime implementovať. Zaznamenávanie videa však patrí medzi vedľajšie ciele.

3.1.6 Vizualizácia získaných dát

Pre výskumníkov je občas nepríjemne pozerat' sa na veľké množstvo dát vo forme textu. Preto by sme chceli vizualizovať získané dáta. S dátami, ktoré sú vizualizované pomocou grafov, tepelných máp a rôznych obrázkov, sa pracuje jednoduchšie.

3.2 Krátkodobý podrobný plán

3.2.1 Prvý šprint - 1. a 2. týždeň

V prvom šprinte by sme sa chceli venovať anotácií prúdu dát a poskytnutia API. Chceli by sme identifikovať čo najviac rôznych typov anotácií a zabezpečiť anotovanie prúdu dát získaného zo sledovania zariadenia pomocou zariadenia. V súvislosti s poskytnutím API by sme sa chceli zamerať na interpretovanie anotácií a identifikovať akcie ako spôsob odporúčania.

3.2.2 Druhý šprint - 3. a 4. týždeň

V tomto šprinte by sme sa chceli venovať ďalej tvorbe API a začať s poskytovaním Správy zariadení.

V závislosti od identifikovaných akcií ako spôsobov odporúčania by sme chceli implementovať tieto akcie a začať poskytovať API pre výučbový systém ALEF.

Pri správe zariadení by sme sa spočiatku chceli zamerať na zobrazenie stavu zariadení. Táto funkcionálna by mala výskumníkovi zobrazit' v prvom rade informácie o tom, či sú používatelia prihlásení a či už majú nakalibrované zariadenia. Následne by mala výskumníkovi umožňovať "odštartovať" experiment.

3.2.3 Tretí šprint - 5. a 6. týždeň

V treťom šprinte by sme chceli dokončiť API a začať sa venovať vizualizácii.

V tomto štádiu by malo byť API nasadené a prístupné pre reálne používanie. Dokončovanie API by spočívalo v identifikovaní nových typov anotácií a implicitnej spätnej väzby a implementácia implementácia adekvátnych odporúčaní.

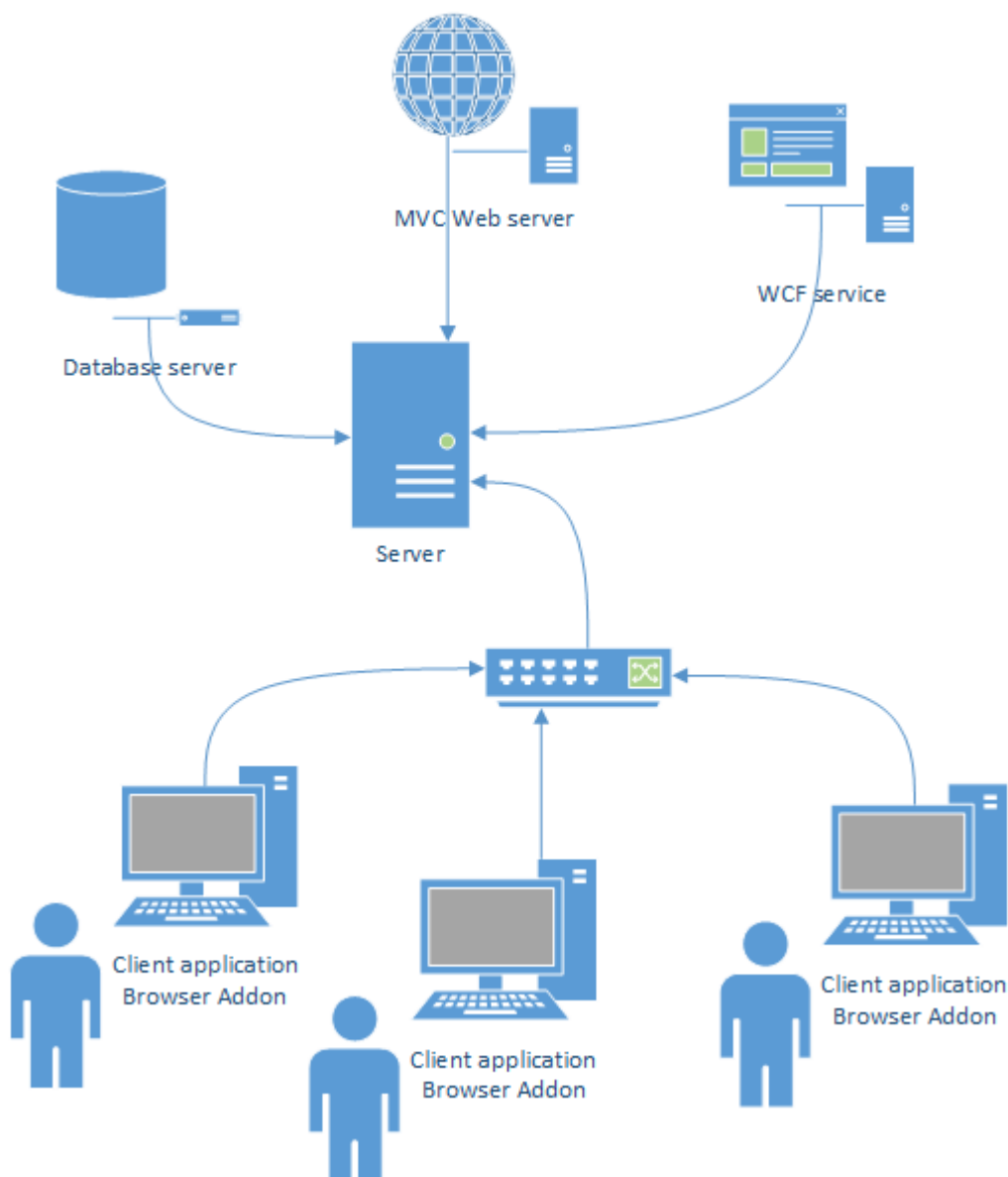
Vizualizáciu by sme chceli zo začiatku implementovať zobrazovaním grafov, vytváraných z nazbieraných dát. Tieto grafy pomôžu výskumníkovi orientovať sa v nazbieraných dátach z experimentov a jednoduchšie robiť z experimentov závery.

4 Opis prototypu

V tejto časti je opísaný náš prototyp z hľadiska jeho architektúry, funkcionálnych požiadaviek a dátového modelu.

4.1 Architektúra

Architektúra nášho prototypu je momentálne rozložená do troch vrstiev. Na nasledovnom diagrame vidíme rozloženie nášho systému.



Obrázok 1 - Diagram architektúry prototypu

Na prvej vrstve sa nachádza desktopový klient, ktorý zabezpečuje komunikáciu so zariadením na sledovanie pohľadu. Jeho úlohou je zisťovať, kam sa používateľ pozerá a odosielať tieto dáta na server.

Klientovi na prvej vrstve pomáha plniť jeho činnosť prvok druhej vrstvy - rozšírenie pre webový prehliadač. Keďže sa sústreďujeme na dynamické webové aplikácie, rozšírenie pre webový prehliadač je nevyhnutná súčasť architektúry, ktorá nám pomáha presne zisťovať, kam sa používateľ na prehliadač pozerá. Túto skutočnosť komunikuje s desktopovým klientom, ktorý následne tieto dáta a mnohé ďalšie posiela na server.

Tretia vrstva - samotný server - bude zabezpečovať zber dát získavaných z viacerých desktopových klientov pomocou REST architektúry. Dôležitou súčasťou tejto vrstvy je aj databáza, na ktorej sú rôzne dáta - od informácií o používateľoch až po sedenia a projekty navrhnuté výskumníkmi.

Ako je zrejmé z tohto opisu, náš prototyp je založený na kombinácii vrstvovej a klient-server architektúry.

4.2 Funkcionálne požiadavky

Pre prácu so systémom sme identifikovali roly, ktoré sú spomenuté v časti 4.3 Dátový model.

4.2.1 Desktopová aplikácia

S desktopovou aplikáciou môže pracovať ktokoľvek zapojený do výskumu, a teda ktokoľvek kto je aspoň “bežný používateľ”. Pod desktopovou aplikáciou rozumieme klienta, ktorý zabezpečuje prácu a komunikáciu so zariadením pre sledovanie pohľadu od firmy Tobii.

Základnou úlohou tohto klienta je teda prijímanie dát zo zariadenia a ich posielanie na server. Obohacovanie dát o ďalšie informácie je ďalšou funkcionalitou, ktorá sa týka dát získaných zo sledovania pohľadu.

Po spustení je pre používateľa dôležité *prihlásenie*. Prihlásením sa overí totožnosť používateľa, čím sa tiež vyznačí jeho prítomnosť na konkrétnom sedení experimentu. Prihlásením sa tiež používateľovi sprístupní možnosť *kalibrácie* zariadenia. Ak je kalibrácia úspešná, používateľ môže spustiť *sledovanie* pohľadu a tým odosielanie dát na server.

4.2.2 Webová aplikácia

Do webovej aplikácie sa môže prihlásiť každý používateľ systému. Roly používateľov však určujú práva a možnosti práce s webovou aplikáciou. “Bežný používateľ” má možnosť si, po prihlásení, *prezerat' experimenty*, ktorých sa zúčastnil. “Výskumník” má právo si *prezerat' experimenty*, ktorých je súčasťou, *prezerat' získané dáta* zo sedení, patriacich k danému experimentu a *viest' sedenia* daného experimentu. Nemá právo meniť údaje v rámci webovej aplikácie.

Hlavným používateľom webovej aplikácie je “projektový administrátor”. Ma všetky vyššie spomenuté práva. Ďalej mu aplikácia poskytuje *vytvárať experimenty*, v rámci nich *vytvárať sedenia* a *pridávať* alebo *odoberať používateľov* v rámci experimentov.

4.2.3 Rozšírenie pre webový prehliadač

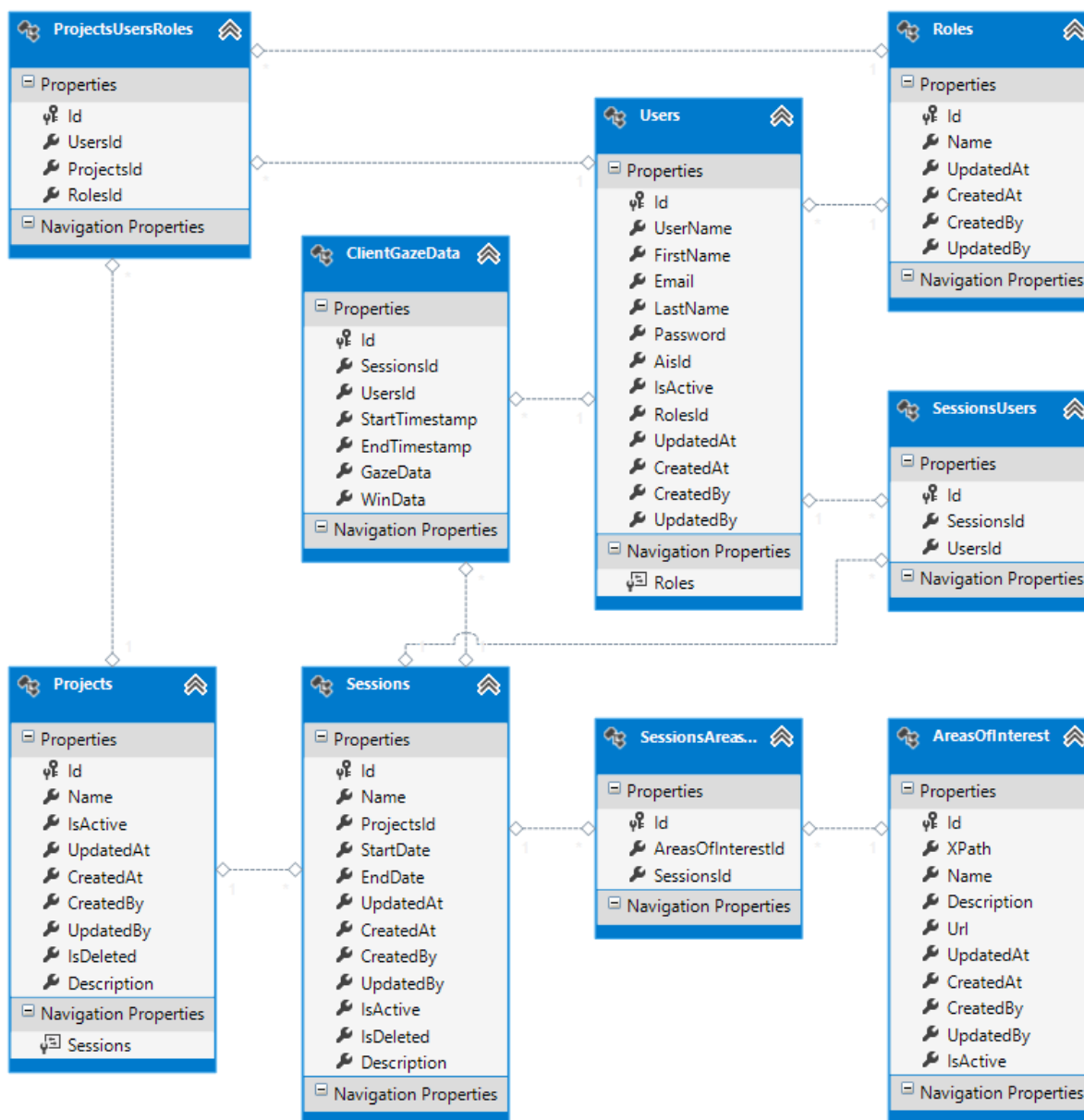
S rozšírením pre webový prehliadač pracuje priamo vedúci experimentu a nepriamo “bežní používatelia”. Vedúcemu experimentu rozšírenie poskytuje možnosť *nastavovať oblasti záujmu* a ich *dôležitosť* v rámci webovej stránky, ktorá je cieľom testovania.

“Bežní používatelia” s rozšírením pracujú nepriamo, nakoľko nevyužívajú žiadnu funkcionality. V tomto prípade jeho funkcionality využíva desktopová aplikácia, s ktorou pracujú “bežní používatelia” a ktorá získava súradnice pohľadu na obrazovke. K týmto súradniciam potom rozšírenie dokáže *namapovať* elementy webovej stránky, ktorá je predmetom experimentu.

4.3 Dátový model

Nasledujúca schéma znázorňuje dátový model nášho prototypu.

4 Opis prototypu



Obrázok 2 - Fyzický dátový model prototypu

Hlavnou entitou v nej je používateľ (Users), so svojou globálnou rolou. Ten si po prihlásení môže prezerať projekty (Projects) a ich details, ak má dostatočné práva, môže k projektu pridávať sedenia (Sessions), iných používateľov s rôznymi rolami (Roles) a pridávať používateľov k sedeniam.

V súčasnom prototypu rozlišujeme celkovo 4 typy rol. Prvou z nich je administrátor, ktorý má po prihlásení plné práva na celú implementovanú funkcionality. Zvyšné tri sa vzťahujú priamo k projektu a sedeniam:

- projektový administrátor (project admin)
 - bezprostredne po vzniku projektu je ním používateľ, ktorý projekt vytvoril

4 Opis prototypu

- môže pridávať k projektu ďalších používateľov, buď ako projektových administrátorov alebo ako výskumníkov
- môže meniť nastavenia projektu, pridávať k projektu sedenia a k sedeniam bežných používateľov
- výskumník (researcher)
 - môže si prezerať vytvorené projekty a sedenia, nemôže k nim však pridávať používateľov
 - v budúcnosti plánujeme, že bude mať prístup k štatistikám a anotovaným dátam, tiež však len na čítanie
 - tento typ používateľa je, podobne ako projektový administrátor pridaný k projektu
- bežný používateľ (common user)
 - používateľ, ktorý sa zúčastňuje sedenia/experimentu
 - môže sa prihlásiť, nechať nakalibrovať zariadenie na sledovanie pohľadu a interagovať s testovanou aplikáciou
 - nemá prístup k nastaveniam sedenia, k projektom a v budúcnosti nebude mať prístup ani ku vyhodnoteniam a štatistikám

Používateľ môže mať viacero rolí. V rámci jedného projektu však iba jednu.

4.3.1 Opis entít

Všetky nasledujúce tabuľky okrem väzobných obsahujú povinné údaje o tom kto a kedy záznam vytvoril a kedy a kým bol naposledy modifikovaný. Tieto informácie sú uložené v atribútoch CreatedAt, CreatedBy, UpdatedAt a UpdatedBy. Nakoľko mazať reálne dáta z databázy nie je vhodné každá z týchto entít obsahuje aj atribút isDeleted ktorá reprezentuje to či je daný záznam aktuálny. Každá tabuľka obsahuje atribút ID ktorý je unikátny kľúč pre daný záznam v tabuľke.

Users

```
string UserName - unikátne prihlasovacie meno používateľa
string FirstName - krstné meno používateľa
string Email - email patriaci používateľovi
string LastName - priezvisko používateľa
string Password - heslo používateľa (v prípade AIS loginu je heslo null)
int AisId - AIS ID používateľa
int RolesId - cudzí kľúč do tabuľky Roles, reprezentuje rolu používateľa
```

Session

```
string Name - názov pre dané sedenie
int ProjectsId - cudzí kľúč ktorý odkazuje na projekt ku ktorému sedenie patrí
Nullable<System.DateTime> StartDate - čas od kedy je sedenie aktuálny je len informatívny
Nullable<System.DateTime> EndDate - čas dokedy je sedenie aktuálne je len informatívny
bool IsActive - informuje či je sedenie aktívne
string Description - krátky opis sedenia
```

Roles

4 Opis prototypu

`string` Name - názov danej roly

Projects

`string` Name - reprezentuje názov projektu

`bool` IsActive - informuje či je projekt aktívny

`string` Description - opis projektu

Area of interest

`string` XPath - Xpath elementu ktorý chceme sledovať

`string` Name - Názov pre identifikáciu záznamu

`string` Description - opis záujmovej oblasti

`string` Url - URL stránky ku ktorej sa daný záznam vzťahuje

Client gaze data

`int` SessionsId - cudzí kľúč reprezentujúci sedenie ku ktorému dáta patria

`int` UsersId - cudzí kľúč reprezentujúci používateľa ku ktorému dáta patria

`string` StartTimestamp - timestamp prvého záznamu v dátach

`string` EndTimestamp - timestamp posledného záznamu v dátach

`string` GazeData - serializovaný list surových dát z Tobii

`string` WinData - serializovaný list dát z pohľadu a windowsu

Väzobné entity a teda tabuľky `ProjectsUsersRoles`, `SessionsUsers` a `SessionsAreasOfInterest` slúžia na prepojenie vyššie spomenutých entít a obsahujú len cudzie kľúče do iných tabuliek.

5 ACDC - 1. šprint

V rámci prvého šprintu okrem zabezpečenia organizačných záležitostí tímu sme si definovali niekoľko menších prevažne analytických úloh, pre potreby rozhodovania sa v ďalších šprintoch. Niektorí členovia tímu už ale začali pracovať na implementácii.

5.1 Identifikácia elementu na stránke

5.1.1 User story

Výskumník potrebuje identifikovať elementy na stránke, aby vedel zdefinovať area of interest.

5.1.2 Úloha

Získanie XPath adresy požadovaného elementu pre potreby opätovného identifikovania elementu v dokumente.

5.1.3 Analýza

Analýzou dostupnej funkcionality JQuery, ale aj samotného JavaScriptu sa dospelo k záveru, že požadovaná funkcionality nie je dostupná v rámci existujúcej implementácie a preto bolo potrebné vytvorenie vlastného dedikovaného riešenia.

5.1.4 Implementácia

Riešenie problému získania XPath adresy zvoleného elementu je zabezpečené prostredníctvom rekurzívneho prechádzania stromovej štruktúry DOM od zvoleného elementu až po koreň dokumentu. Prehľadávanie je možné ukončiť pri elemente, ktorý má zadaný parameter id, teda je v dokumente jedinečný. Pre potreby projektu však nepostačuje získanie len takejto čiastočnej adresy z dôvodu nemožnosti identifikácie elementov vo vnútri elementu, ktorý je definovaný ako oblasť záujmu. Implementovaný algoritmus umožňuje generovať oba výstupy.

Príklad úplnej a neúplnej cesty

```
html/body/div[id="wrapper"]/div[id="header"]/div[id="loginPanel"]/div[2]/form[1]
//div[id="loginPanel"]/div[2]/form[1]
```

Implementácia algoritmu

```
// -- XPATH GENERATOR -----
getXPath = function(element, fullpath = false) {
  if (tag(element) === "BODY") {
    return "/HTML/BODY";
  }
  if (tag(element) === "HTML") {
    return "/HTML";
  }
}
```

5 ACDC - 1. Šprint

```
if (id(element) !== "") {
    var p = tag(element) + "[@id='" + id(element) + "']";
    if(fullpath == false){
        return "/" + p;
    }
    else {
        return getXPath(element.parentNode) + "/" + p;
    }
}

var n = 0, index = 0;
n = countPrevSib(element);
index = n + 1;

return getXPath(element.parentNode) + "/" + tag(element) + "[" + index + "];
};
```

5.1.5 Testovanie

Testovanie generovania XPath pre dané elementy spočívalo v porovnaní očakávanej XPath cesty s vygenerovanou cestou.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	zavolanie funkcie pre generovanie XPath adresy na element <form> (s atribútom id)	//div[id='loginPanel']/div[2]/form[1]	//div[id='loginPanel']/div[2]/form[1]
2.	zavolanie funkcie pre generovanie XPath adresy na element <h1> (bez atribútu id)	//html/body/div[1]/h1	//html/body/div[1]/h1

5.2 Vytvorenie Add-onu pre browser

5.2.1 Úloha

Vytvorenie základného add-onu do prehliadača.

5.2.2 Implementácia

Na vytvorenie Add-onu sa použil framework Crossrider, ktorý dokáže vytvoriť Add-on do viacerých prehliadačov súčasne. Vytvára sa cez webovú stránku <http://crossrider.com>. Postup je jednoduchý aj keď nasadenie je rozličné napríklad v prehliadačoch Firefox a Chrome. Zmeny sa dajú ukladať dvoma spôsobmi a to Staging a Production. Staging je uloženie len v rámci počítača vývojára, teda pre testovanie, či všetko funguje ako má. Production je teda konečné uloženie, ktoré je viditeľné už aj pre používateľa.

5.3 Kalibrácia zariadenia

5.3.1 User story

Používateľ chce kalibrovat' zariadenie, aby ho mohol používať s čo najväčšou presnosťou.

5.3.2 Úloha

Vytvoriť základ desktopovej aplikácie, ktorá bude obsahovať možnosť kalibrácie pripojeného zariadenia.

5.3.3 Analýza

Pre vytvorenie základnej desktopovej aplikácie boli analyzované príklady aplikácií v Tobii SDK. Bol preštudovaný ich zdrojový kód, čím sme sa naučili pracovať aj so samotným Tobii SDK.

5.3.4 Implementácia

Desktopová aplikácia je implementovaná ako Windows Forms aplikácia v jazyku C#. Pre prvú verziu tejto aplikácie sme sa rozhodli znovupoužiť kód z jednej z príkladových aplikácií v Tobii SDK.

Pre kalibráciu sme sa rozhodli použiť 5 kalibračných bodov, ktoré sú podľa Tobii dostatočným počtom pre kalibráciu. Vyšší počet kalibračných bodov síce spresňuje kalibráciu, no už len veľmi mierne. 5 kalibračných bodov taktiež zaručuje, že kalibrácia nebude trvať príliš dlho a obťažovať používateľa.

Ako inšpirácia pre prvú verziu kalibrácie slúžil príklad aplikácie z Tobii SDK, v ktorom sa jednoducho na obrazovke objavovali a mizli kalibračné body. Tento spôsob kalibrácie sme ale neskôr zmenili na spôsob podobný kalibrácii v Tobii Studio.

Pri tomto spôsobe kalibrácie je stále viditeľný kurzor, ktorý plynulo prechádza medzi kalibračnými bodmi, takže používateľ má čas sa zorientovať a pripraviť sa na kalibráciu konkrétneho bodu. Pred kalibrovaním konkrétneho bodu sa kurzor zmenší a po kalibrácii znova zväčší na pôvodnú veľkosť, čo napomáha používateľovi pozerat' sa presne na daný bod.

5.3.5 Testovanie

Kalibrácia bola testovaná priamo so zariadením pre sledovanie pohľadu. Úspešne sa dokázala pripojiť na zariadenie a vykonať kalibráciu. Po prechode na spôsob kalibrácie podobný Tobii studio sa nám podarilo výsledok kalibrácie zlepšiť vďaka dôvodom popísaným v časti Implementácia.

Testovanie teda môžeme zapísať nasledovným akceptačným testom.

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
-------------	-------	-------------------	------------------

1.	Spustenie klienta	Klient sa spustí	Klient sa spustil
2.	Kliknutie na spustenie kalibrácie	Spustí sa kalibrácia	Spustila sa kalibrácia
3.	Sledovanie kalibračných bodov	Zobrazenie výsledku úspešnej kalibrácie	Zobrazili sa panely s vizualizáciou kalibrácie

5.4 Autentifikácia a autorizácia

5.4.1 User story

Používateľ vyžaduje prihlasovanie do webovej aplikácie tak aby bol schopný rozlišovať roly na rôznych úrovniach a tiež vytvoriť viacero projektov pre rôzne druhy aplikácie.

5.4.2 Úloha

Vytvoriť základnú kostru všetkých projektov v .net. Vytvoril som projekt ViewTracking.Web kde bude situovaná webová aplikácia slúžiaca na analýzu a spracovanie dát z klientskej aplikácie.

5.4.3 Implementácia

Vytvoril som tiež ostatné projekty ako ViewTracking.WinForms a Wcf. Ďalšou pod úlohou bolo vytvoriť logiku prihlasovania v používateľov v Web aplikácii, ktorá si berie dáta z databáze. K úlohe prihlásenie bolo treba napísať testy, ktoré slúžia aj ako predloha pre ostatné testy. Keďže postupujeme metódou riedenou testovaním testy sa písali pred napísaním reálneho kódu.

5.4.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie zobrazenia Webového projektu po prihlásení	Zobrazenie mena pri prihláseného používateľa	Zobrazené meno po prihlásení

5.5 Analýza typov anotácií

5.5.1 User story

Výskumník chce anotovať prúd dát zo sledovania pohľadu aby ich mohol vyhodnocovať ale nechce to robiť ručne.

5.5.2 Úloha

Analyzujte a identifikujte možné typy anotácií, ktoré možno generovať automaticky, zo získaných dát

5.5.3 Analýza

Z dát, získaných zo sledovania pohľadu je možné automatizovane vydedukovať niektoré fakty, ktoré môžu byť použité ako anotácie, ak používateľ povolí ich pridávanie. Anotáciu pre účely tohto projektu rozumieme **označenie určitého úseku dátového prúdu s priradenie mu určitého kľúčového slova/skupiny slov.**

Pridávanie automatických anotácií by malo byť závislé od nastavení - používateľ experimentátor si zadefinuje hraničné hodnoty parametrov a na ich základe sa budú automatické anotácie generovať

Požiadavky zo strany zákazníka na vytváranie anotácií:

- Možnosť nastavenia automatických anotácií
- Možnosť výberu z existujúcich kľúčových slov alebo pridanie nového kľúčového slova, resp. slovného spojenia.
- Pri zadávaní kľúčových slov má systém odporúčať posledné zadané kľúčové slová.

Parametre ktoré je možné sledovať a na základe nich, identifikované možné typy automatických anotácií:

1. dĺžka (čas) pohľadu na jedno miesto
 - používateľ sledoval / nesledoval oblasť záujmu, prípadne hodnoty medzitým
 - časové - číselné alebo slovné vyjadrenie
 2. pohyb očí vzhľadom na pohyb myši
 - nesledoval/nenasledoval očami pohyb myši
 3. frekvencia, rýchlosť zmeny pohľadu
 - čítal/nečítal text
 - bol zmätený/vedel sa na stránke rýchlo zorientovať
 4. spomenuté metriky vzhľadom na počet používateľov
 - koľko používateľov... (napr. sledovalo oblasť záujmu dlhšie ako určitý čas)
- koľko percent používateľov... - opäť by mohla byť zaujímavá možnosť slovného vyjadrenia - kľúčové slovo nastavené používateľom experimentátorom, alebo číselného

5.6 Komunikácia pluginu s desktopovou aplikáciou

5.6.1 User story

Používateľ sa do aplikácie chce prihlásiť len raz a zúčastňovať sa experimentu.

Výskumník chce spájať súradnice pohľadu s reálnymi oblasťami webovej stránky, kde prebieha experiment.

5.6.2 Úloha

Je potrebné identifikovať prihláseného používateľa cez desktopovú aplikáciu, kde prebehne kalibrácia zariadenia, ako aj cez webový prehliadač, cez ktorý sa používateľ účastní experimentu. Je tiež potrebné, aby sa dáta - súradnice pohľadu párovali s oblasťami záujmu, ktoré chce sledovať výskumník. Analyzujte možnosti komunikácie webového prehliadača (Mozilla Firefox, Google Chrome) s desktopovou .NET aplikáciou.

5.6.3 Analýza

Identifikovali sme dve možnosti komunikácie:

1. možnosť
 - nájsť knižnicu, ktorá umožní prístup z aplikácie k prehliadaču cez inštanciu triedy (podobne ako komunikácia s IE), bez potreby pridávania komponentov alebo add-on-ov do prehliadača.
 - Problémy: síce existujú knižnice na prácu s prehliadačom Firefox, ale väčšinou ide o vytvorenie komponentu prehliadača v rámci aplikácie, a nie

možnú komunikáciu so zvlášť otvoreným prehliadačom. Navyše už niekoľko nemajú podporu.

2. možnosť

- vytvoriť add-on a v rámci neho implementovať komunikáciu
- posielanie socketov alebo JSON-ov, medzi Add-onom v prehliadači a desktopovým klientom, najjednoduchšie cez localhost
- po analýze a zvážení sme sa rozhodli pre toto riešenie

5.7 Ďalšie úlohy na ktorých sa pracovalo počas šprintu

5.7.1 Komunikácia so zariadením

Úlohou bolo vykonať analýzu nad komunikáciou medzi zariadením od firmy Tobii a počítačom, na ktorý je toto zariadenie pripojené. Hlavným cieľom bolo zistiť formát dát, ktoré posiela Tobii zariadenia pri sledovaní pohľadu. Výsledkom analýzy je dokument, ktorý je v prílohe tohto dokumentu v časti A.1.1.

5.7.2 Analýza prehliadačov Firefox a Chrome

Analýza prehliadačov Firefox a Chrome, slúžila na uľahčenie výberu prehliadača pre vývoj rozšírenia, ktoré má za úlohu komunikovať s klientsku časťou našej aplikácie.

Kritéria, v ktorých boli porovnávané tieto prehliadače:

- spôsob inštalácie pluginov:
 - Firefox - inštalácia je možná za behu ale je nutné reštartovať prehliadač po inštalácii
 - Chrome - inštalácia pluginu je potrebná cez Google Store alebo cez windows inštalátor avšak nie je potrebné reštartovať
- rýchlosť vykonávania JavaScriptu:
 - Chrome je rýchlejší pri vykonávaní JavaScriptu avšak samostatné operácie s DOM sú rýchlejšie v Firefoxe. V konečnom dôsledku je ale Chrome rýchlejší.
- poskytované api + dokumentácia:
 - Aj Firefox aj Chrome sú dobre zdokumentované a majú veľké vývojárske komunity.
- možnosti integrácie do GUI prehliadača:
 - Firefox - GUI je robená cez XUL.
 - Chrome - elementy browser action môžu byť napríklad icon, tooltip, badge, popup, kontextové menu, notifikácie.
- použité technológie pre rozšírenia:
 - Firefox - XUL, JavaScript

- Chrome - HTML, JavaScript
- iné výhody/nevýhody:
 - Firefox - v samostatnom procese bežia len rozšírenia. Môže ho spomaľovať väčšie množstvo otvoreného obsahu. Má jednoprocessorový režim a teda spotrebuje menej pamäte, čo však spôsobuje problémy s rýchlym uvoľňovaním pamäte
 - Chrome - beží kompletne samostatne (aj rozšírenia aj karty).
- frameworky pre tvorbu crossbrowser rozšírení:
 - **kango**: nie je úplne free, slabá dokumentácia
 - **crossrider**: free, otestovaný > ukázkový plugin pre prehliadače
 - **crossbrowser**: podobný online framework, chyba automatické aktualizovanie

babelext: len BETA verzia

5.7.3 Analýza a porovnanie socketov a RESTu

Pre čo najjednoduchšiu a jednotnú komunikáciu medzi klientom a serverom bolo potrebné učiť spôsob akým budú komunikovať. Bolo potrebné sa rozhodnúť medzi použitím socketov a REST architektúry. Výsledkom analýzy bolo rozhodnutie že komunikácia bude na báze REST. Výstup analýzy je dokument, ktorý sa nachádza v prílohe dokumentu, v časti A.1.2

6 Beatles - 2. šprint

6.1 Identifikácia elementov na základe súradnice

6.1.1 User story

Výskumník potrebuje vedieť identifikovať elementy na základe súradnice, aby vedel, na ktorý element sa používateľ pozeral.

6.1.2 Úloha

Rozšíriť identifikácie elementu o možnosť identifikovať element na základe zadanej pozície.

6.1.3 Analýza

V referenčnej príručke k jazyku JavaScript bola počas analýzy problematiky tejto úlohy identifikovaná funkcia, ktorá je vhodná pre implementovanie riešenia.

6.1.4 Implementácia

Pre implementovanie požadovanej funkcionality bola využitá JavaScript funkcia `elementFromPoint(x, y)`, kde `x` a `y` sú jednotlivé súradnice.

Z dôvodu rozdielnosti návratovej hodnoty funkcie `elementFromPoint(x, y)` a vstupného argumentu funkcie `getXPath(element)`, bolo potrebné pôvodne implementovaný kód funkcie `getXPath(element)` refaktorovať priamo do jazyka JavaScript, namiesto pôvodne použitého JQuery. Táto zmena mala za následok prehodnotenie použitia knižnice JQuery. Výsledkom refaktoringu celého kódu je odstránenie závislosti na knižnici JQuery, zabezpečenie kompatibility implementovaných funkcií a urýchlenie vykonávania celého skriptu.

Bola vytvorená nová funkcia `getXPathFromPos(x, y)`, ktorá priamo vracia XPath adresu elementu podľa súradnice.

6.1.5 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	zavolanie funkcie <code>getXPathFromPos(x,y)</code> na požadovaný element	XPath adresa požadovaného elementu	XPath adresa požadovaného elementu
2.	zavolanie funkcie <code>getXPathFromPos(x,y)</code> na telo stránky	XPath adresa /HTML/BODY	XPath adresa /HTML/BODY

6.2 Používateľ sa chce prihlásiť (autentifikovať)

6.2.1 User story

Používateľ sa chce prihlásiť aby výskumník vedel, že sa zúčastnil experimentu.

6.2.2 Úloha

Súčasťou tejto úlohy bolo kompletne prerobenie používateľského rozhrania desktopovej aplikácie. Aplikácia mala doteraz rovnaké používateľské rozhranie, ako príklad aplikácie z Tobii SDK, z ktorého sme vychádzali pri prvotnom vytvorení aplikácie.

Hlavnou časťou tejto úlohy potom bolo vytvorenie formulára pre prihlásenie používateľa.

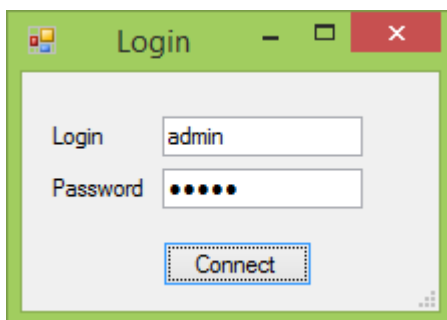
6.2.3 Analýza

Počas stretnutia tímu bol vytvorený papierový návrh nového používateľského rozhrania. Toto nové používateľské rozhranie zahŕňa formulár pre prihlásenie, formulár pre výber zariadenia, formulár pre kalibráciu a formulár pre zobrazenie stavu zariadenia pre sledovanie pohľadu.

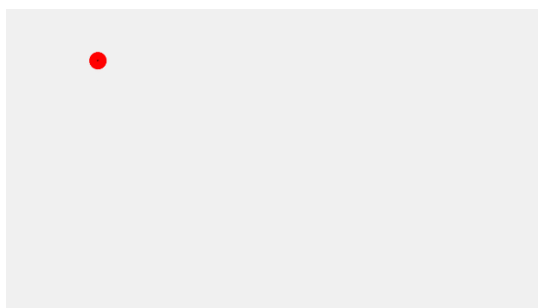
6.2.4 Implementácia

Konkrétne formuláre boli implementované pomocou Windows Forms. Rozhodli sme sa znovupoužiť niektoré komponenty z predchádzajúcej verzie programu. Patrí medzi ne napr. čierne okno, v ktorom je vidno pozíciu očí používateľa, alebo panely zobrazujúce výsledok kalibrácie.

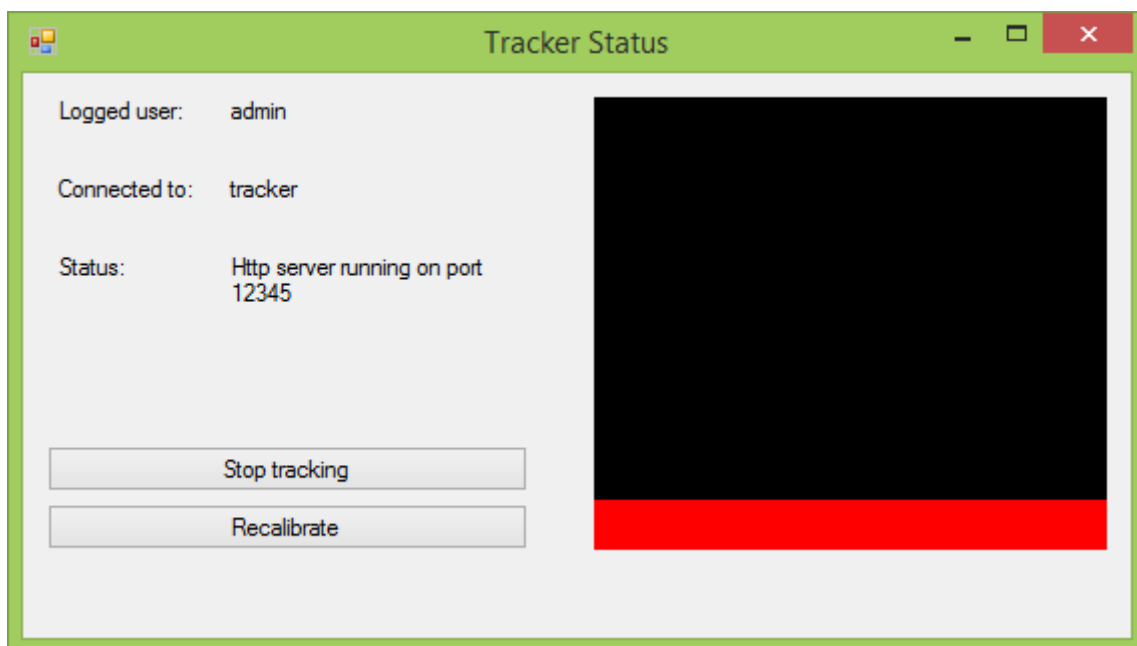
Používateľské rozhranie vyzerá nasledovne:



Obrázok 3 - Formulár pre prihlásenie používateľa



Obrázok 4 - Kalibrácia zariadenia



Obrázok 5 - Formulár zobrazujúci stav programu

6.2.5 Testovanie

Po zmenách používateľského rozhrania bol program znovu testovaný so skutočným zariadením pre sledovanie pohľadu. Bolo odhalených niekoľko menších bugov, ktoré boli v zápätí opravené.

Prihlásenie bolo testované nasledovnými akceptačnými testami.

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Spustenie klienta	Klient sa spustí	Klient sa spustil
2.	Vloženie správneho prihlasovacieho mena a hesla	Používateľ bude prihlásený	Používateľ bol prihlásený

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Spustenie klienta	Klient sa spustí	Klient sa spustil
2.	Vloženie nesprávneho prihlasovacieho mena a hesla	Používateľ nebude prihlásený	Používateľ nebol prihlásený

6.3 Prepočítavanie súradníc vzhľadom na aktívne okno

6.3.1 User story

Výskumník chce mať prepočítané súradnice zo zariadenia na aktuálne aktívne okno, aby vedel, kam sa presne na okno používateľ pozerá.

6.3.2 Úloha

Na strane desktopového klienta analyzovať, čo všetko vieme zistiť o konkrétnom okne a implementovať funkcie, ktoré by nám dávali o okne tieto informácie.

Na strane browser add-onu prepočítavať súradnice pohľadu tak, aby v ľavom hornom rohu obsahu stránky boli súradnice 0,0 a v pravom dolnom rohu súradnice 1,1(po normalizácii), resp. súradnice v pixeloch pre maximálnu šírku a výšku contentu stránky.

6.3.3 Analýza

Súčasťou tejto úlohy bola analýza WinApi funkcií. Potrebovali sme z Windows API zistiť informácie o okne. Výsledkom tejto analýzy je dokument „Analýza WinApi funkcií“, ktorý sa nachádza v prílohe A v časti A.2.1.

Druhou časťou tejto úlohy je prepočítavanie súradníc na okno browsera pomocou add-onu. Tejto problematike sme sa už venovali v inej úlohe a preto môžeme znovupoužiť riešenie z tejto úlohy.

6.3.4 Implementácia

Pomocou externých WinApi funkcií bolo vytvorených niekoľko funkcií pracujúcich s informáciami, ktoré WinApi poskytuje. Použité boli nasledovné WinApi funkcie na nasledovné účely:

GetForegroundWindow - používame pre zisťovanie, či je dané okno v popredí

GetWindowRect - používame, keď chceme presnú pozíciu okna

GetWindowPlacement - používame, keď chceme vedieť pozíciu okna a aký je jeho stav (minimalizované, maximalizované...)

Funkcie sú použité vo funkcii LookingAtBrowser, ktorá na základe súradníc pohľadu vracia bool hodnotu, podľa toho, či sa používateľ pozerá na web browser (konkrétne FireFox).

6.3.5 Testovanie

Pre testovanie WinApi funkcií bol vytvorený unit test, ktorý využíva funkciu LookingAtBrowser. Pre jeho úspešné zbehnutie je potrebné mať otvorený FireFox, musí byť v popredí a musí byť na súradniciach 50,50 (prípadne maximalizovaný). Súradnice 50,50 sú simulované súradnice pohľadu. Funkcia dokáže úspešne zistiť, či sa používateľ pozerá na FireFox alebo nie (či sa nachádza na súradniciach 50,50 a či je v popredí).

6.4 Experimentátor chce vidieť možné oblasti dynamicky pri hoveri

6.4.1 User story

Ako experimentátor chcem pri definovaní oblastí záujmu vidieť zvýraznenie zvoleného elementu, aby som vedel overiť správnosť výberu oblasti záujmu.

6.4.2 Úloha

V rámci vznikajúceho add-onu pre Firefox implementujte funkcionality zvýrazňovania elementov pri prechádzaní myšou nad daným elementom. Ako inšpirácia slúži funkcionality prieskumníka kódu vo Firefoxe.

Požiadavky na riešenie:

- ohraničiť oblasť tak, aby nebolo ovplyvnené rozloženie stránky
- zobrazit tooltip s identifikáciou elementu (XPath)

Analyzovať možnosti tvorby tooltipov pomocou knižnice Opentip a možnosti tvorby tooltipov natívne vo Firefoxe.

6.4.3 Implementácia zvýraznenia

Zvýraznenie je implementované využitím posledného princípu spomenutého v analýze zvýraznenia elementu.

Plugin pri udalostiach mouseover a mouseout pridá, respektíve odoberie CSS triedu elementu, nad ktorým bola udalosť registrovaná.

```
// -- HIGHLIGHTER -----  
  
setMarker = function(element, cssClass){  
    element.classList.add(cssClass);  
}  
  
unsetMarker = function(element, cssClass){  
    element.classList.remove(cssClass);  
}  
  
hasMarker = function(element, cssClass){  
    return element.classList.contains(cssClass);  
}
```

V CSS triede je definované zvýraznenie elementu.

```
.highlighterGaze {  
    background:          url('images/h2px.png'),url('images/h2px.png'),  
                        url('images/h2px.png'),url('images/h2px.png');  
    background-position: left top, right top, left top, left bottom;  
    background-repeat:   repeat-y, repeat-y, repeat-x, repeat-x;  
}
```

6.4.4 Implementácie tooltipu

```
// -- TOOLTIP -----  
  
createTooltip = function(element, content){  
    bounding = element.getBoundingClientRect();
```

6 Beatles - 2. Šprint

```
// set position of the tooltip above, or under selected element
var verticalPos = bounding.top - 40;
if(verticalPos < 0){
    verticalPos = bounding.bottom + 5;
}

verticalPos += window.pageYOffset;

// create element and style for tooltip
var newElem = document.createElement('div');
newElem.setAttribute("id", "carrotsTooltip");
newElem.innerHTML = content;
newElem.style.cssText = "position: absolute; top: " + verticalPos + "px; left: "
+ bounding.left + "px; background-color: #ffa500; border: 1px solid white;
border-radius: 6px; color: black; padding: 6px; font-family: Consolas;
font-size: 10pt;";

document.body.appendChild(newElem);

return;
};

removeTooltip = function(){
    // Remove multiple tooltips
    while(e = document.getElementById("carrotsTooltip")){
        e.remove();
    }
};
```

6.4.5 Testovanie

Pri testovaní pôvodne zamýšľaného riešenia zvýraznenia, ktoré spočívalo vo vytváraní prekrývajúcich elementov nad zvýrazňovanými elementami boli identifikované nedostatky, ktoré mali za následok nepoužiteľnosť riešenia. Po zhodnotení možností opravy padlo rozhodnutie pre nájdenie iného riešenia. Novo identifikované riešenie popísané v analýze ako posledná možnosť bola implementovaná a úspešne otestovaná. Testovanie prebiehalo formou overenia funkčnosti pridávania a odoberania atribútov *class* požadovaného elementu. Následne prebiehal test z používateľskej perspektívy, či sa zvýrazňovanie správa podľa definovaných požiadaviek a či je dostatočne reaktivné. Zvýraznenie spĺňa všetky predpoklady.

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	pridanie atribútu triedy zvolenému elementu	pridanie orámovania	pridanie orámovania
2.	odobranie atribútu triedy zvolenému elementu	odobranie orámovania	odobranie orámovania
3.	odobranie atribútu triedy elementu, ktorý ho nemá	žiadna akcia	žiadna akcia

6.5 Simulácia Eye-trackera z pohybu myši

6.5.1 User story

Používateľ vytvára aplikáciu a chce použiť SDK poskytované firmou Tobii. Nevlastní však zariadenie od firmy Tobii, preto nemá žiadne prostriedky na otestovanie funkčnosti svojej aplikácie. Potrebuje preto virtuálne zariadenie, ktoré mu dokáže nasimulovať správanie reálneho zariadenia od firmy Tobii.

6.5.2 Úloha

Vytvorte základnú štruktúru simulátora a simulujte sledovanie pohľadu, kde pohľad je simulovaný myšou. Analyzujte existujúce riešenia.

6.5.3 Analýza

Text 2.0 Framework - jedná sa o projekt napísaný v Jave. Simulátor pozostáva z dvoch komponent, Tobii Eye Tracker Server (TET Server) a aplikácie, ktorá komunikuje s TET Serverom a získava údaje o sledovaní, pričom sledovanie je simulované myšou.

TET Server - táto časť projektu vykonáva simuláciu pohľadu pomocou myši, sledovaním jej súradníc a odosiela dáta aplikácií, ktorá s ňou komunikuje.

Diagnosis - aplikácia, ktorá komunikuje s TET Serverom, prijíma dáta a vykonáva kalibráciu.

Sledovanie pohľadu nie je nutné simulovať. *Text 2.0 Framework* poskytuje aj prácu s reálnymi zariadeniami od firmy Tobii.

6.5.4 Návrh

Pre simulovanie pohľadu sa použijú súradnice myši, ktoré sa pred odoslaním budú normalizovať. Zvyšné údaje týkajúce sa dát získavaných zo sledovania pohľadu pomocou zariadenia od firmy Tobii sa budú generovať. Generovanie bude spočívať v manuálnom nastavení hodnoty a vyznačenia možnosti náhodného generovania.

6.5.5 Implementácia

Aplikáciu pre simulovanie pohľadu pomocou myši sme naimplementovali v jazyku C#. Pre sledovanie súradníc myši sme použili projekt KeyLogger dostupný na stránke:

<http://cskeylogger.codeplex.com>. Aplikáciu tvorí jedno okno, kde je možné nastavovať hodnoty jednotlivých údajov, ktoré sa generujú.

6.5.6 Testovanie

Počas vývoja aplikácie sme testovali, či súradnice, ktoré sú zaznamenávané sú správne. Zistili sme rozlíšenie displeja a sledovali sme hodnoty, ktoré sa zaznamenali v rohoch monitora. Aplikácia zaznamenala správne hodnoty súradníc vo všetkých rohoch.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Spustenie a automatické odosielanie údajov o simulátore	Simulátor sa spústi, a začne odosielať packety obsahujúce výrobné údaje o simulátore	Simulátor sa spustil a odosiela údaje hneď po spustení
2.	Testovanie pripojenia	Simulátor nahlási, či je alebo nie je pripojený ku klientovi	Simulátor správne hlásil stav pripojenia
3.	Spustenie "sledovanie pohľadu" pomocou myši	Simulátor spustí sledovanie, súradnice sa budú prepisovať na základe súradníc získaných z myši	Súradnice sa po spustení prepisovali

6.6 Prihlásenie ako REST

6.6.1 Úloha

Vytvoriť API ktoré bude môcť jednoducho a rýchlo komunikovať s inými aplikáciami

6.6.2 Analýza

Vybrať vhodnú technológiu pre servis ako je rest alebo sockety atď.

6.6.3 Implementácia

Po schválení a skúmaní toho, aká technika je vhodná pre WCF projekt. RESTFull servis sme vybrali ako najvhodnejší pre naše potreby. Preto jadrom tejto úlohy bolo implementovať úlohu „Prihlásenie používateľa“ do RestFull architektúry.

6.6.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie odoslania rest servis	Odoslanie požiadavky na servis	Odoslaná požiadavka na servis

6.7 Vytvorenie nového používateľa

6.7.1 User story

Zákazník si želá prostredie pre jednoduché pridávanie používateľov s prehľadnou validáciou zadaných hodnôt.

6.7.2 Analýza

Treba vytvoriť Formulár pre pridávanie nového používateľa kde bude možné vyberať roly a vyplňať ďalšie atribúty používateľov. V tomto formulári je nutné vytvoriť jQuery validáciu pre kontrolu vstupu používateľa. Nakoniec som vytvoril logiku zapisovanie modelu používateľa do databáze.

6.7.3 Implementácia

vytvoril som model ktorý generuje podľa jeho vlastností formulár pre pridávanie používateľa. Tieto dáta sa že Post request posielajú do databáze.

6.7.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie zobrazenia formulára	zobrazenie formulára	Zobrazený formulár
2.	Testovanie zobrazenia pre výber rolí vo formulári	zobrazenie rolí vo formulári	zobrazené rolí vo formulári

6.8 Zadeľovanie rolí

6.8.1 User story

Požiadavka bola vytvoriť používateľa ktorý bude môcť pristupovať a spravovať všetky moduly webovej aplikácie.

6.8.2 Analýza

Pridávať pomocou SQL kódu do databáze

6.8.3 Implementácia

pri manipulácii s sql kódom som prišiel n problém chyby v MS sql express tak som vytvoril databázu externú v súbore MDF. Cez VS som už len pridal dáta pomocou prostredia v textovom editore Bolo treba minimálne zadeľovať roly v databáze, ktoré sme definovali na stretnutí:

- admin
- researcher
- user

Zabezpečil som, aby pre účely vývoja fungoval default user bez nutnosti ich manuálneho pridania do DB s rolou, čo som implementoval zoznamom List ktorý sa vkladá ak je databáza pregenerovaná :

- admin
- user

Pridával som len dáta písaním SQL kódov do databáze. Tiež som pridával do formulárov pre prihlasovanie a pridávanie používateľa hashovaciu funkciu ktorá zapisovala do databáze už bezpečný reťazec. Použil som funkciu implementovanú v C#, ktorá používa kryptovanie SH1.

6.8.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie zobrazenia záznamu v databáze	zobrazenie záznamu v databáze	Zobrazený záznam v databáze

6.9 Zadeinovanie formy protokolu pre odosielanie dát na server

6.9.1 Úloha

Vytvoriť základnú štruktúru protokolu pre ľahkú manipuláciu a odosielanie dát medzi klientom a serverom. Naštudovanie štruktúry JSON a jeho použitie v protokole. Protokol by musí byť ľahko rozšíriteľný pre pridanie ďalších parametrov s klientskej aplikácie poprípade Tobii zariadenia.

6.9.2 Implementácia

JSON (JavaScript Object Notation) je alternatíva XML formátu. Slúži na serializáciu objektov na textový tvar ktorý je následne na strane klienta respektíve servera deserializovaný na pôvodný objekt. Skladá sa s dvojíc atribút - hodnota. JSON formát bol špecifikovaný Douglasom Crockfordom a celý je popísaný v dokumente RFC 4627¹. Pri navrhnutí bolo potrebné vychádzať s požiadaviek od zákazníka, a teda odosielať dáta ohľadom pohľadu používateľa ako aj informácie priamo o danom používateľovi a prebiehajúcim sedení respektíve projektu. Pre spresnenie informácií sme do protokolu pridali aj informácie z operačného systému a teda informácie o polohe kurzora myši a aplikácii ktorá bola v danej chvíli zameraná. Taktiež odosielame dáta ohľadom zameraného elementu na stránke ak bola v popredí, odosiela sa jej XPath.

štruktúra tried pre serializáciu a deserializáciu.

```
[DataContract]
public class ViewData
{
    [DataMember]
    public int userID { get; set; }
    [DataMember]
    public int testID { get; set; }
    [DataMember]
    public int projectID { get; set; }
    [DataMember]

```

¹ Odkaz na špecifikáciu JSON - <http://tools.ietf.org/html/rfc4627>

6 Beatles - 2. Šprint

```
public string IPAddress { get; set; }
[DataMember]
//TOBII DATA
List<TobiiData> viewData;
[DataMember]
//WIN DATA
List<WinData> winData;
}
[DataContract]
public class TobiiData
{
    public string str_timeStamp {get;set;}
    [DataMember]
    public float[,] eyePosition; //poradie je left right treba ho
dodrzat --> pole 2x3
    [DataMember]
    public float[,] relativeEyePosition; //poradie je left right treba
ho dodrzat --> pole 2x3
    [DataMember]
    public float[,] pointOfView3D; //poradie je left right treba ho
dodrzat --> pole 2x3
    [DataMember]
    public float[,] pointOfView2D; //poradie je left right treba ho
dodrzat --> pole 2x2
    [DataMember]
    public int leftEyeValidate {get;set;}
    [DataMember]
    public int rightEyeValidate { get; set; }
    [DataMember]
    public float leftLenseDiameter { get; set; }
    [DataMember]
    public float rightLensediameter { get; set; }
}
[DataContract]
public class WinData
{
    [DataMember]
    public float[] mouseCoordination; //x,y
    [DataMember]
    public string focusedApplication;
}
}
```

Serializovaný objekt na JSON potom vyzerá takto:

Štruktúra dát bola odprezentovaná a schválená na spoločnom stretnutí tímu.

6.10 Odosielanie dát na server

6.10.1 User story

Výskumník chce posielat' dáta z klienta na server, aby mal dáta zo všetkých klientov pozbierané na rovnakom mieste.

6.10.2 Úloha

Odosielanie zozbieraných dát na server pomocou REST architektúry.

6.10.3 Implementácia

Pre riešenie bolo potrebné naprogramovať funkciu na odosielanie dát pomocou REST a taktiež funkciu v službe na servery ktorá dáta spracuje. Dáta sú na server posielané v správe typu Request na ktorý príde Reply či dáta boli v poriadku a správne deserializované. Pre serializáciu objektov bola použitá knižnica Newtonsoft.Json.dll ktorá podporuje spomínanú serializáciu nulových atribútov. Na stretnutí tímu boli funkcie odprezentované a schválené.

Funkcia na odosielanie a serializáciu dát:

```
string json = JsonConvert.SerializeObject(FillData(), Formatting.Indented,
    new JsonSerializerSettings { DefaultValueHandling =
DefaultValueHandling.Ignore });
HttpWebRequest request =
(HttpWebRequest)WebRequest.Create("http://localhost:51367/ViewTrackingServi
ce.svc/processData");
request.Method = "POST";
System.Text.UTF8Encoding encoding = new System.Text.UTF8Encoding();
Byte[] byteArray = encoding.GetBytes(json);
request.ContentLength = byteArray.Length;
request.ContentType = @"application/json";
using (Stream dataStream = request.GetRequestStream()) {
    dataStream.Write(byteArray, 0, byteArray.Length);
}
long length = 0;
try {
    using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse()) {
        length = response.ContentLength;
    }
}
catch (WebException ex) {
    // Log exception and throw as for GET example above
}
```

Funkcia na strane servera pre prijatie dát:

```
public string SaveData(ViewData data)
{
    Console.WriteLine(data);
    return "Data accepted by server";
}
```

Pre správne fungovanie Request so strany klienta musí byť definovaný na strane servera:

```
[OperationContract]
[WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Bare,
UriTemplate = "processData")]
string SaveData(ViewData data);
```

6.10.4 Testovanie

Bolo potrebné aby serializácia a deserializácia prebehla aj v prípade, ak majú niektoré atribúty hodnotu null. Správna serializácia atribútu s hodnotou null (nebola mu pridelená hodnota), sa v JSON stringu neobjaví. Ak by tam hodnota bola ako napríklad

"timestamp:null", pri deserializácii by sa atribút timestamp nenastavil na hodnotu null ale na string null. V prípade že by daný atribút nebol typu string deserializácia by sa nepodarila a vrátila nulový pointer.

Bol vykonaný nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Serializácia objektu s null hodnotami	Pri serializácii v JSONe atribút nie je	Pri serializácii v JSONe atribút nie je
2.	Odosielanie na server keď nie je dostupný	Zobrazí sa upozornenie o nedostupnosti servera	Zobrazí sa upozornenie o nedostupnosti servera

6.11 Vytvorenie projektu

6.11.1 User story

Výskumník chce vytvoriť projekt, v rámci ktorého bude vykonávať experimenty.

6.11.2 Úloha

Umožniť v rámci webovej aplikácie vytváranie projektov.

6.11.3 Analýza

Na realizáciu tejto úlohy bolo potrebné:

- navrhnuť spôsob reprezentácie projektu v databáze
- vytvoriť používateľské rozhranie pre zobrazenie a úpravu existujúcich projektov a formulár pre pridávanie nového projektu

6.11.4 Implementácia

Projekt v našej databáze obsahuje meno, príznak, či sa jedná o aktívny alebo neaktívny projekt, identifikátor používateľa, ktorý ho vytvoril (a je jeho vlastníkom), identifikátor používateľa, ktorý ho ako posledný upravil (bezprostredne po vytvorení sú tieto atribúty totožné) a dátum a čas vytvorenia aj poslednej úpravy. V rámci projektu bude možné vytvoriť niekoľko sedení, čím vznikol v databáze vzťah 1:N medzi projektom a sedením.

Riešenie bolo implementované pomocou MVC. Modely sa generujú pomocou Entity Framework 6 z databázy. Z controllera sú volané potrebné metódy, ktoré komunikujú s databázou, zabezpečujú prenos dát v podobe JSON a o ich správne zobrazenie sa starajú Views v adresári projekt.

Používateľské rozhranie bolo navrhnuté ako tabuľka existujúcich projektov. Po kliknutí na projekt sa zobrazia jeho detaily používateľa a sedenia. Druhou časťou je jednoduchý formulár pre vytvorenie projektu, kde používateľ vyplní a potvrdí minimálne povinné polia. Grafika rozhrania sa riadi navrhnutou šablónou a využíva štýly bootstrap.

6.11.5 Testovanie

Pri testovaní bolo dôležité kontrolovať, či polia, ktoré sú pri pridávaní povinné, naozaj aplikácia vyžaduje vyplniť. Ďalej bolo nutné overovať, či vložené dáta majú požadovaný typ a formát a v neposlednom rade či sa dáta správne odosielať na server a ukladajú do databázy.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klik na položku Projects v hlavnom menu	Zobrazí sa zoznam všetkých projektov z databázy	Zobrazil sa zoznam všetkých projektov z databázy
2.	Klik na tlačidlo Add project	Zobrazí sa formulár na vyplnenie	Formulár sa zobrazil
3.	Pokus o vytvorenie projektu - správne vyplnenie všetkých polí	Vytvorí sa projekt a zobrazia sa jeho detaily	Projekt sa vytvoril
4.	Pokus o vytvorenie projektu - nevyplnenie polí	Projekt sa nevytvorí, povinné polia sa zvýrazia	Projekt sa nevytvoril, povinné polia sa zvýraznili
5.	Pokus o vytvorenie projektu - nesprávne vyplnenie polí (zlý formát)	Projekt sa nevytvorí, nesprávne vyplnené polia sa zvýrazia s upozornením pre používateľa	Projekt sa nevytvoril, nesprávne vyplnené polia sa zvýraznili s upozornením pre používateľa
6.	Zobrazenie zoznamu projektov po úspešnom pridaní nového projektu	Zobrazí sa tabuľka so zoznamom všetkých projektov, vrátane nového pridaného projektu	Zobrazila sa tabuľka so zoznamom všetkých projektov, vrátane nového pridaného projektu

Z akceptačného testu vyplýva, že aplikácia správne zobrazí všetky projekty, ktoré sú uložené v databáze a zabezpečí správne pridanie nového projektu.

6.12 Vytvorenie sedenia v rámci projektu

6.12.1 User story

Výskumník chce vytvoriť v rámci projektu sedenie, ktoré predstavuje jeden experiment so skupinou používateľov.

6.12.2 Úloha

Umožniť vo webovej aplikácii výskumníkovi vytvoriť konkrétny experiment - sedenie.

6.12.3 Analýza

Na realizáciu tejto úlohy bolo potrebné:

- navrhnuť spôsob reprezentácie sedenia v databáze
- vytvoriť používateľské rozhranie pre zobrazenie a úpravu existujúcich sedení a formulár pre pridávanie nového sedenia

6.12.4 Implementácia

Entita sedenie v našej databáze obsahuje názov, identifikátor príslušného projektu, dátum a čas začiatku a konca sedenia, identifikátor používateľa, ktorý ho vytvoril (a je jeho vlastníkom), identifikátor používateľa, ktorý ho ako posledný upravil (bezprostredne po vytvorení sú tieto atribúty totožné) a dátum a čas vytvorenia aj poslednej úpravy.

Podobne ako vytváranie projektu, aj toto riešenie využíva model MVC.

Existujúce sedenia si môže používateľ prezrieť pri detailoch projektu, kde je možné ich, podobne ako projekty, upravovať. GUI sa riadi tými istými navrhnutými štýlmi a rozložením, ako pri zobrazovaní a pridávaní projektov.

6.12.5 Testovanie

Rovnako ako pri vytváraní projektu, pri testovaní bolo najdôležitejšie kontrolovať správnosť formátu zadaných údajov ako aj komunikáciu so serverom a databázou. Ak v databáze nie je pre projekt uložené žiadne sedenie, bolo nutné ošetriť, aby sa zobrazila prázdna tabuľka sedení a nie chyba z databázy.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klik na položku Details v zozname projektov pri ľubovoľnom projekte	Zobrazí sa stránka s dvoma záložkami: detaily projektu a zoznam sedení	Zobrazila sa stránka s oboma záložkami
2.	Klik na záložku Sessions	Zobrazí sa zoznam sedení k danému projektu	Zobrazil sa zoznam sedení k danému projektu
3.	Klik na tlačidlo Add session	Zobrazí sa formulár na vytvorenie sedenia k projektu	Zobrazil sa formulár na vytvorenie sedenia k projektu
4.	Pokus o vytvorenie sedenia - správne vyplnenie všetkých polí	Vytvorí sa sedenie a zobrazia sa jeho detaily na ďalšiu editáciu	Vytvorilo sa sedenie a zobrazili sa jeho detaily na ďalšiu editáciu
5.	Pokus o vytvorenie sedenia - nevyplnenie polí	Sedenie sa nevytvorí, povinné polia sa zvýraznia	Sedenie sa nevytvorilo, povinné polia sa zvýraznili
6.	Pokus o vytvorenie sedenia - nesprávne vyplnenie polí (zlý formát)	Sedenie sa nevytvorí, nesprávne vyplnené polia sa zvýraznia s upozornením pre používateľa	Sedenie sa nevytvorilo, nesprávne vyplnené polia sa zvýraznili s upozornením pre používateľa
7.	Klik na tlačidlo Back to project details po vytvorení sedenia	Zobrazia sa detaily projektu, v zozname sedení pribudne nové	Zobrazili sa detaily projektu, v zozname sedení pribudlo nové

Z akceptačného testu vyplýva, že aplikácia správne zobrazí detaily projektu a umožní k nemu pridať sedenie.

6.13 Ďalšie úlohy na ktorých sa pracovalo počas šprintu

6.13.1 Analýza možných riešení zvýraznenia elementov na stránke

Pre potreby identifikovania oblastí záujmu v rámci sledovania pohľadu, výskumník potrebuje dostávať pri nastavovaní požadovaných oblastí spätnú väzbu o zvolených prvkoch stránky pomocou zvýraznenia aktuálneho elementu (teda aj oblasti).

Niektoré prístupy pre zvýraznenie elementov.

Prepísanie CSS vlastnosti elementu

Najjednoduchším prístupom k zvýrazneniu elementu na stránke, je pridanie css vlastnosti danému elementu. Pridaním vlastnosti, napríklad *border*, deštruktívne zmeníme pôvodnú vlastnosť elementu. Ak budeme chcieť zrušiť zvýraznenie, vieme vlastnosť *border* nastaviť akurát na hodnotu *none*, ale nevieme sa dopracovať k pôvodnej vlastnosti (ak nejaká bola), čo má za následok odstránenie pôvodnej vlastnosti *border* a teda zmenu vzhľadu stránky.

Priradenie/odobratie CSS triedy elementu

Ak zvolenému elementu, namiesto prepísania vlastnosti pridáme CSS class, vieme pôvodnú vlastnosť prekryť nami zadefinovanou. Pridanie a odobratie konkrétnej triedy v elemente je možné vykonávať jednoducho pomocou JavaScriptu.

Problém spôsobov 1. a 2.

Oba spomínané spôsoby neriešia spoločný problém. Akonáhle pridáme elementu vlastnosť *border* akýmkoľvek spôsobom, zväčší sa jeho vonkajšia veľkosť, čo spôsobuje posuny obsahu na stránke, prípadne v niektorých prípadoch rozhodenie celej stránky.

Vytvorenie zvýrazňovacieho elementu

Pre zvýraznenie elementu je možné použiť takú metódu, že po vykonaní akcie nad elementom zistím polohu elementu a jeho veľkosť, čo následne využijem pre vytvorenie nového elementu, ktorý je absolútne pozicionovaný nad ostatnými elementami stránky. Tento element má rovnakú pozíciu a veľkosť ako pôvodný, nad ktorým bola udalosť zachytená a je zvýraznený pomocou okraju. Celé to pôsobí, ako keby bol zvýraznený pôvodný element.

Problém však vzniká, keď zvýrazníme element, ktorý obsahuje vnútri aj iné elementy. Tieto elementy sa nám bohužiaľ nedá zvýrazniť, pretože keď sa vytvorí zvýrazňovací element nad pôvodným elementom prekryje tie elementy vnútri, keďže Firefox má jednotlivé elementy štruktúrované vo vrstvách.

Pridanie okraju použitím obrázku na pozadí

Vytvoríme si obrázok veľkosti 2x2 pixely, ktorý použijeme 4x ako pozadie elementu tak, že sa 2x opakuje vo vodorovnom smere, čo vytvorí horný a dolný okraj a 2x v zvislom smere pre ľavý a pravý okraj. Toto riešenie na rozdiel od pridania reálneho okraja nemení veľkosť elementu a teda nemení layout stránky.

6.13.2 Analýza knižnice Opentip

Opentip je JavaScript framework, ktorý slúži na vytváranie tooltipov na web stránke. Ponúka viacero preddefinovaných štýlov, ale zároveň umožňuje nastavenie vlastného vzhľadu. Ako už názov napovedá, zdrojový kód je otvorený.

Oficiálna stránka projektu uvádza hneď niekoľko jednoduchých príkladov, ako vytvoriť tooltipy (ako element, alebo ako JS objekt). K jednotlivým tooltipom je možné priradiť element, na ktorý sa majú pripnúť. Taktiež je možné samozrejme nastaviť obsah, ktorý má zobrazovať, alebo napríklad delay pre zobrazenie. Ďalej nasleduje popísanie funkcií API, ktoré je pomerne bohaté.

Existuje viacero spôsobov vytvorenia tooltipov ako element alebo ako JavaScript objekt. Príklady týchto spôsobov sa dajú pozrieť na oficiálnej stránke Opentip knižnice(opentip.org). Je možné k jednotlivým tooltipom priradiť element, na ktorý sa majú pripnúť. Ďalej je tiež možné nastaviť obsah, ktorý sa má zobrazovať alebo delay pre zobrazenie.

Implementácia Opentipu je jednoduchá. Stačí do stránky vložiť JavaScript kód, CSS súbor a následne už len volať jednotlivé tooltipy.

Pri skúšobnom implementovaní bol tooltip nasadený do rozšírenia tak, že pri každej mouseover akcii, sa vytvoril nad elementom a zobrazoval jeho XPath. Celkovo však toto riešenie spomaľovalo interakciu na stránke a pôsobilo ťažkopádne. Pri prechádzaní myši cez viacero elementov, vznikalo niekoľko tooltipov súčasne.

Zo spomenutých dôvodov, navrhujem framework Opentip nepoužiť a nahradiť ho vlastným JavaScript kódom, ktorý bude zobrazovanie tooltipov zabezpečovať oveľa pružnejšie.

6.13.3 Analýza Bootstrap-u

Bootstrap je framework umožňujúci vytváranie webového obsahu rýchlejšie a intuitívnejšie. Obsahuje rozsiahle CSS, za pomoci ktorého je možné vytvárať používateľsky prívetivé rozhranie jednoduchým zadefinovaním patričnej triedy požadovanému prvku. Framework zahŕňa aj JavaScript a JQuery knižnice, ktoré dokážu jednotlivé prvky oživiť požadovanou funkcionalitou. Aj menej skúsený tvorca webu má možnosť pomocou Bootstrapu tvoriť moderný web s profesionálnym vzhľadom.

Klady použitia frameworku

- nie je nutné navrhovať štylovanie stránky od základov
- je vytvorený jednotný vzhľad
- podporuje responzívny design
- možnosť pomocou CSS vlastností vytvárať prvky, ktoré nie sú elementárne
 - ikonky s logom
 - drop-down menu
 - navigácia
 - textové polia s ikonkou
 - pagination a iné
- obsahuje sadu ikoniek
- JavaScript a JQuery funkcionalita
- free

Zápory použitia frameworku

- ~~+ 350Kb ktoré treba načítať pri zobrazení stránky (dá sa upraviť podľa potrieb)~~
- ~~zbytočne rozsiahla knižnica pre naše využitie (dá sa upraviť podľa potrieb)~~
- potrebné sa naučiť ako framework využívať

Z analýzy vyplynulo, že framework Bootstrap poskytuje rozsiahlu paletu prvkov a funkcionalít pre web, ktoré je možné využiť aj pre potreby nášho projektu. Pomocou tejto knižnice, za predpokladu, že tvorca je s ňou patrične oboznámený, je možné vytvárať web efektívne a taktiež aj efektne. Z tohto dôvodu padlo rozhodnutie pre použitie frameworku na tvorbu webového rozhrania.

6.13.4 Analýza surových packetov z Eye-trackera

Úloha sa týka vytvorenia simulátora zariadenia od firmy Tobii. Cieľom bolo analyzovať celú komunikáciu zariadenia s počítačom, ktorá zahŕňa identifikovanie zariadenia, sledovanie jeho stavu, napojenie sa na zariadenie a posielanie dát z pozorovania pohľadu. Pre vytvorenie simulátora bolo nutné podrobne analyzovať packety, obsahujúce dáta zo sledovania, aby sme ich boli schopný reprodukovať. Nebolo však možné túto úlohu splniť, nakoľko nebola prístupná žiadna dokumentácia ani metóda na analyzovanie týchto packetov.

7 Coldplay - 3. Šprint

7.1 Prihlásenie používateľa cez LDAP

7.1.1 User story

Výskumník chce, aby sa používatelia mohli prihlasovať pomocou AIS, aby nemusel vytvárať ručne konto pre každého používateľa.

7.1.2 Úloha

Implementovať autentifikáciu používateľa cez LDAP. Pri neúspechu pokračovať s autentifikáciou používateľa cez vlastnú databázu.

7.1.3 Implementácia

Prihlasovanie bolo implementované v podobe funkcie, ktorá priamo overuje, či sa používateľ nachádza v systéme a zároveň sa overuje, či zadal správne heslo.

```
public static bool isAuthenticated(String user, String password)
{
    bool authenticated = false;

    String uid = "uid="+user+",ou=people,dc=stuba,dc=sk";
    String _path = "LDAP://ldap.stuba.sk";

    DirectoryEntry de = new DirectoryEntry(_path, uid, password,
        AuthenticationTypes.None);

    try
    {
        object connected = de.NativeObject;
        authenticated = true;
        //Console.WriteLine("Login successful.");
    }
    catch (Exception ex)
    {
        //Console.WriteLine("Login failed.");
    }

    return authenticated;
}
```

Funkcia bola pridaná do projektu v zmysle zadania, teda že po neúspešnom pokuse autentifikovať sa cez LDAP nasleduje pokus o prihlásenie cez lokálnu databázu.

7.1.4 Testovanie

Autentifikačný kód bol overený s využitím dvoch rôznych AIS loginov, pri ktorých boli zadané správne, ale aj nesprávne heslá. Rovnaký postup sa opakoval aj pre vymyslené prihlasovacie mená. Výstupom bola vždy očakávaná návratová hodnota, teda úspešné, respektíve neúspešné prihlásenie.

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	zadanie jedného z dvoch AIS loginov so správnym heslom	úspešné prihlásenie	úspešné prihlásenie
2.	zadanie jedného z dvoch AIS loginov s nesprávnym heslom	neúspešné prihlásenie	neúspešné prihlásenie
3.	zadanie vymysleného loginu so správnym heslom	neúspešné prihlásenie	neúspešné prihlásenie
4.	zadanie vymysleného loginu s nesprávnym heslom	neúspešné prihlásenie	neúspešné prihlásenie

7.2 Podpora pre usporiadanie, filtrovanie a pagináciu dát v tabuľke

7.2.1 User story

Výskumník chce mať možnosť tabuľky v grafickom rozhraní filtrovať, zoradovať a rozdeliť na stránky, aby sa v nich vedel lepšie orientovať.

7.2.2 Úloha

Implementovať funkcionality pre možnosť manipulácie dát v tabuľke. Umožniť usporiadanie podľa stĺpca, filtrovanie a pagináciu záznamov v tabuľke. Riešenie musí byť implementované na strane serveru.

7.2.3 Analýza

Z požiadavky v zadaní na serverové riešenie vzniká nutnosť analýzy možných riešení manipulácie s tabuľkou, nakoľko pôvodne zvažované riešenie využitím DataTables od JQuery vyžaduje odoslanie celého výsledku z dopytu na klientskú stranu a následne prácu nad celou dátovou štruktúrou, čo vyžaduje prenos veľkého množstva dát. V rámci .NET platformy bola identifikovaná funkcionality pod názvom WebGrid, ktorá na základe modelu poskytnutého ako argument funkcie dokáže generovať tabuľku s podporou usporadúvania elementov a paginácie. Toto riešenie bolo implementované.

7.2.4 Implementácia

```
@{
    var grid = new WebGrid(Model, canPage: true, rowsPerPage: 15);
    @grid.GetHtml(tableStyle: "table table-hover", footerStyle: "footerStyle",
        columns: grid.Columns(
            grid.Column("Name", "Name", canSort: true),
            grid.Column("CreatedAt", "Created at", canSort: true),
            grid.Column("CreatedBy", "Administrator", canSort: true),
            grid.Column("IsActive", "State", format: (item) => (item.IsActive) ?
                Html.Raw("Áno") : Html.Raw("Nie"), canSort: true),
            grid.Column("Id", " ", format: (item) => Html.ActionLink("x", "Delete",
                new { id=item.Id }, new { @class = "btn btn-danger btn-sm" })))
    });
}
```

7.2.5 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	usporiadanie tabuľky podľa definovaného stĺpca	preusporiadanie záznamov podľa stĺpca	preusporiadanie záznamov podľa stĺpca
2.	opačné usporiadanie tabuľky podľa aktuálne definovaného stĺpca	opačné preusporiadanie záznamov podľa stĺpca	opačné preusporiadanie záznamov podľa stĺpca
3.	zobrazenie ďalších záznamov tabuľky	zobrazenie ďalšej časti záznamov v tabuľke	zobrazenie ďalšej časti záznamov v tabuľke
4.	zobrazenie posledných záznamov tabuľky	zobrazenie poslednej časti záznamov v tabuľke	zobrazenie poslednej časti záznamov v tabuľke

7.3 Komunikácia add-onu s klientom na desktope

7.3.1 User story

Výskumník chce vedieť, kam sa presne v okne pozerá používateľ, aby vedel zistiť, či sa pozerá na zadanú oblasť záujmu.

7.3.2 Úloha

Zabezpečiť komunikáciu medzi desktopovým klientom a browser add-onom pre počítanie súradníc.

7.3.3 Analýza

Na stretnutí tímu sme rozmýšľali nad rôznymi spôsobmi komunikácie medzi add-onom a desktopovým klientom. Ako najvhodnejší spôsob sme vybrali vytvorenie lokálneho http servera na desktopovom klientovi, od ktorého si bude add-on pýtať súradnice pomocou GET requestu a posielat' mu prepočítané súradnice pomocou POST requestu.

7.3.4 Implementácia

http server je v desktopovom klientovi implementovaný pomocou .NET triedy HttpListener. Vďaka tomu je server jednoducho implementovaný v samostatnom vlákne a neblokuje zvyšnú činnosť desktopového klienta.

Po GET requeste odošle server pomocou JSON triedu, obsahujúcu súradnice pohľadu. V POST requeste očakáva, že sa mu daná trieda vráti naspäť, ale so súradnicami prepočítanými add-onom.

7.3.5 Testovanie

Server bol testovaný posielaním GET a POST requestov pomocou Google Chrome aplikácie „Postman - REST Client“. Server úspešne dokázal prijímať a posielat' súradnice pohľadu, posielané ako JSON. Pri testovaní klient zobrazoval okno s prijatým POST requestom, čo pri bežnej prevádzke nebude robiť.

Tento test je popísaný nasledovným akceptačným testom.

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Spustenie klienta	Klient sa spustí	Klient sa spustil
2.	Poslanie GET requestu z Postman aplikácie klientovi	V Postman aplikácii uvidíme odpoveď na GET request	Odpoveď bola zobrazená
3.	Poslanie POST requestu z Postman aplikácie klientovi	Zobrazí sa okno s prijatým POST requestom	Okno sa zobrazilo

7.4 Adaptér pre klienta

7.4.1 User story

Používateľ vytvára aplikáciu a chce použiť SDK poskytované firmou Tobii. Nevlastní však zariadenie od firmy Tobii, preto nemá žiadne prostriedky na otestovanie funkčnosti svojej aplikácie. Potrebuje preto virtuálne zariadenie, ktoré mu dokáže nasimulovať správanie reálneho zariadenia od firmy Tobii.

7.4.2 Úloha

Vytvorte knižnicu, ktorú bude možné použiť v akomkoľvek projekte. Knižnica by mala simulovať funkčnosť Tobii SDK. Musí zabezpečiť spojenie so simulátorom a prijímanie dát ním poslaných. Súčasťou knižnice má byť parser, ktorý dokáže prijaté dáta namapovať do štruktúr Tobii SDK. Úloha súvisí so simuláciou Eye-trackera z pohybu myši.

7.4.3 Analýza

Bolo potrebné analyzovať Tobii SDK, aby sme mohli navrhnúť naše riešenie simulácie. Potrebovali sme zistiť v akých štruktúrach si uchováva informácie o zariadení a dáta zo sledovania pohľadu.

class EyeTrackerInfoEventArgs - trieda, ktorá sa používa pri vyhľadávaní zariadení. Obsahuje triedu *EyeTrackerInfo*, ktorá predstavuje štruktúru pre uchovávanie informácií o zariadení. Jej atribúty sú:

Factory (tento atribút nás nezaujíma)

- Generation
- GivenName
- Model
- ProductId
- Status
- Version

Dáta pohľadu sa ukladajú do rozhrania *IGazeDataItem*, ktoré je uložené v triede *GazeDataEventArgs*. Atribútmi rozhrania sú:

- RightValidity
- LeftValidity
- RightPupilDiameter
- LeftPupilDiameter
- RightGazePoint2D
- LeftGazePoint2D
- RightGazePoint3D
- LeftGazePoint3D
- RightEyePosition3D
- LeftEyePosition3D
- RightEyePosition3DRelative
- LeftEyePosition3DRelative

Všetky spomenuté atribúty majú označenie “readonly”, tým pádom ich nie je možné nastavovať.

7.4.4 Návrh

Nakoľko nie je možné nastavovať atribúty vytvorených inštancií, vytvoríme vlastné štruktúry, ktoré budú dediť od vyššie spomenutých tried. Ich funkcie však prepíšeme aby bolo možné nastavovať hodnoty atribútom vytvorených inštancií. Dáta budeme prijímať ako text, ktorý sa bude parsovať do jednotlivých atribútov. Nakoľko je potrebné rýchle posielanie správ a nie je potrebná vysoká spoľahlivosť komunikácie, posielanie správ bude realizované pomocou socketov nad UDP protokolom.

Správa by mala nasledovnú štruktúru:

Flag;Atribút1=hodnota;Atribút2=hodnota1,hodnota2;...

Flag - určuje o aký druh správy sa jedná, môže nadobudnúť hodnoty *GazeDataString* pre dáta vygenerované zo simulácie sledovania.

Atribút - atribúty sú oddelené bodkočiarkou. Hodnota atribútu nasleduje po znamienku “=”. Atribút môže mať viac hodnôt, potom sú hodnoty oddelené čiarkou.

Hodnota - hodnota atribútu.

7.4.5 Implementácia

Knižnica bola implementovaná v jazyku C#. Využíva tiež knižnicu Tobii SDK. Na komunikáciu využívame sockety nad protokolom UDP so zvoleným protokolom.

7.4.6 Testovanie

Bolo nutné vykonať testovanie prijímania správ a parsovania dát do atribútov vytvorených tried. Na testovanie prijímania správ sme vytvorili jednoduchú aplikáciu, ktorá po kliknutí na tlačidlo odoslala správu. Prijímanie dát prebiehalo bez problémov. Na otestovanie parsovania dát do atribútov vytvorených tried sme posielali pomocou spomenutej aplikácie správy, ktoré simulovali dáta získane z pozorovania pohľadu. Ak boli dáta korektné, parsovanie prebehlo bez problémov. Ak prišli chybné dáta, parsovanie bolo prerušené a prijímanie dát sa zastavilo. Tento problém sme vyriešili kontrolovaním *Flagov*.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Prijímanie správ a ukladanie do buffera	Adaptér prijme správy, odfiltruje nežiaduce správy a zvyšné uloží do buffera správ	Adaptér správy prijal, filtroval nežiaduce správy a zvyšné ukladal do buffera
2.	Parsovanie správ a vytváranie inštancií Tobii SDK	Vyberie správu z buffera rozparsuje, vytvorí inštanciu <i>GazeDataItem</i> a naplní ju rozparovanými dátami	Vybral správu, rozparsoval, vytvoril inštanciu a naplnil ju. Pri neznámom údaji nachádzajúcom sa v dátach zobrazil výnimku

7.5 Simulácia Eye-trackera z pohybu myši

7.5.1 User story

Používateľ vytvára aplikáciu a chce použiť SDK poskytované firmou Tobii. Nevlastní však zariadenie od firmy Tobii, preto nemá žiadne prostriedky na otestovanie funkčnosti svojej

aplikácie. Potrebuje preto virtuálne zariadenie, ktoré mu dokáže nasimulovať správanie reálneho zariadenia od firmy Tobii.

7.5.2 Úloha

Rozšírte a upravte doteraz vytvorenú štruktúru simulátora sledovania pohľadu pomocou myši. Vytvorte klienta a zabezpečte odosielanie generovaných dát.

7.5.3 Analýza

Bolo potrebné analyzovať Tobii SDK, aby sme mohli navrhnúť naše riešenie simulácie. Potrebovali sme zistiť v akých štruktúrach si uchováva informácie o zariadení a dáta zo sledovania pohľadu aby sme vedeli, ktoré dáta je nutné simulovať.

Tu sa môžeme odkázať na analýzu úlohy Adaptér pre klienta.

7.5.4 Návrh

Nakoľko by sme chceli simulovať samotné zariadenie aj sledovanie pohľadu, posielali by sme dva druhy správ. Jedna správa by obsahovala informácie týkajúce sa zariadenia. Tieto údaje by sme umožnili používateľovi nastaviť prostredníctvom text boxov. Druhým typom správy by bola správa obsahujúca simulované dáta sledovania pohľadu. Hodnoty GazePoint2D by sa nastavovali zo súradníc získaných z myši. Zvyšné údaje by sme povolili používateľovi nastavovať pomocou textboxov. Pri každom atribúte by bol checkbox, ktorý by predstavoval možnosť náhodného generovania odchýlok. Tieto odchýlky by sa pripočítavali alebo odčítavali daným hodnotám. Komunikácia by prebiehala pomocou socketov nad protokolom UDP so zvoleným portom.

7.5.5 Implementácia

Aplikácia bola implementovaná v jazyku C#. Na komunikáciu využívame sockety nad UDP protokolom so zvoleným portom.

7.5.6 Testovanie

Bolo potrebné otestovať posielanie generovaných dát. Na otestovanie sme použili vytvorenú knižnicu, ktorú sme integrovali do pomocnej aplikácie. Následne sme sledovali komunikáciu medzi simulátorom a serverom, ktorý predstavuje aplikácia s knižnicou. Aplikácia prijala všetky dáta poslané simulátorom.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Generovanie dát a vytvorenie správy na odoslanie	Simulátor vygeneruje dáta podľa nastavených hodnôt a vytvorí z nich textovú správu	Simulátor vygeneroval dáta podľa nastavených hodnôt a vytvoril z nich správu
2.	Odoslanie dát a ich prijatie na strane klienta	Simulátor odošle správu dát, ktorú klient prijme a uloží sa	Komunikácia a teda posielanie dát prebehla úspešne

7.6 Manažment používateľov CRUD

7.6.1 User story

Požiadavka na prostredie pre manipuláciu s používateľmi a s prehľadom zapísaných používateľov

7.6.2 Analýza

Vytvoriť formuláre a prehľady na web aplikácii pre zobrazovanie a pridávanie, editovanie a mazania používateľov. Validácia formulárov bola aplikovaná cez javascript jquery

7.6.3 Implementácia

Implementovaná bola aj logika repozitárov pre zjednodušenie unit testovania a aj preto že volanie metód bude na jednom mieste pričom sa predíde redundancii. Pre repozitár používateľského modelu som vytvoril CRUD funkcie. Používame entity framework, čiže nám vtom to prípade generuje model a databázu sám framework, čo uľahčuje prácu písania kódu. Generovanie formulárov z modelov bolo jednoduchšie keďže používame MVC kde je možnosť nastaviť si v modely akú vlastnosť treba generovať. Ďalej som vytvoril logiku zapisovanie do databáze, čo mi uľahčila vstavaná knižnica v C# linq.

7.6.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie zobrazenia detailu používateľa	Zobrazenie záznamu v databáze	Zobrazený záznam v databáze
2.	Testovanie vymazania používateľa	Vymazaný záznam používateľa	Vymazaný záznam používateľa
	Testovanie zobrazenia používateľov v tabuľke	Zobrazení používateľa v tabuľke	Zobrazení používateľa v tabuľke

7.7 Import používateľov z CSV súboru do databázy

7.7.1 User story

Výskumník chce importovať používateľov z CSV súboru, aby ich nemusel pridávať ručne.

7.7.2 Úloha

Vytvoriť metódu pre importovanie používateľov s CSV súboru cez webové rozhranie.

7.7.3 Implementácia

Riešenie bolo implementované pomocou MVC. Na webovej stránke sa vytvoril formulár na vloženie súboru s obmedzením na súbory s koncovkou CSV. V Controllery pre daný View sa súbor spracuje rozparsuje a následne sa pridelia hodnoty pre atribúty objektu UserDto. Následne sa zavolá metóda na servery ktorá ako parameter dostane pole objektov UserDto a vloží ich do databázy.

Štruktúra POST Requestu

7 Coldplay - 3. Šprint

```
[OperationContract]
[WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare,
UriTemplate = "AddUsers")]
string AddUsers(List<User> users);
```

Formulár pre upload súboru na server

```
@using (Html.BeginForm("Import", "UserImport", FormMethod.Post, new { enctype =
"multipart/form-data" }))
{
    <input type="file" name="file" accept=".csv" />
    <input type="submit" name="Submit" id="Submit" value="Upload" />
}
```

Metóda v Controllery ktorá posiela POST Request na server

```
[HttpPost]
public ActionResult Import(HttpPostedFileBase file)
{
    if (file != null && file.ContentLength > 0)
    {
        StreamReader b = new StreamReader(file.InputStream);
        string line="";
        List<UserDto> importThis = new List<UserDto>();
        while ((line=b.ReadLine())!=null)
        {
            string[] list = line.Split(',');
            UserDto u = new UserDto();
            u.UserName = list[0];
            u.Email = list[1];
            u.Password = list[2];
            importThis.Add(u);
        }
        string sURL =
ConfigurationManager.ConnectionStrings["WcfConnnection"].ToString() + "AddUsers";
        WebRequest wrGETURL;
        wrGETURL = WebRequest.Create(sURL);
        wrGETURL.Method = "POST";
        wrGETURL.ContentType = @"application/json; charset=utf-8";
        wrGETURL.ContentLength = 0;
        string json = JsonConvert.SerializeObject(importThis,
Formatting.Indented, new JsonSerializerSettings { DefaultValueHandling =
DefaultValueHandling.Ignore });
        wrGETURL.ContentLength = json.Length;
        using (StreamWriter writer = new
StreamWriter(wrGETURL.GetRequestStream()))
        {
            writer.Write(json);
        }
        WebResponse response = wrGETURL.GetResponse();
        Stream reader = response.GetResponseStream();
        StreamReader sReader = new StreamReader(reader);
        string outResult =
JsonConvert.DeserializeObject<string>(sReader.ReadToEnd());
        sReader.Close();
    }
    return RedirectToAction("Index", "Home");
}
```

7.7.4 Testovanie

Implementácia bola testovaná posielaním súborov so zlou štruktúrou a neybratím súboru pri kliknutí odoslať.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klikneme na tlačidlo Send bez toho, aby sme vybrali súbor	Zobrazí sa upozornenie pre vybratie súboru	Zobrazí sa upozornenie pre vybratie súboru
2.	Vyberieme súbor so zlou štruktúrou	Zobrazí sa upozornenie, že súbor má zlú štruktúru	Zobrazí sa upozornenie, že súbor má zlú štruktúru

7.8 Pridanie používateľa k projektu

7.8.1 User story

Výskumník chce pridať k vytvorenému projektu používateľa a nastaviť mu práva.

7.8.2 Úloha

Umožniť používateľovi pri úpravách vlastností projektu pridať používateľa a priradiť mu rolu, od ktorej sa odvíjajú jeho práva nad daným projektom.

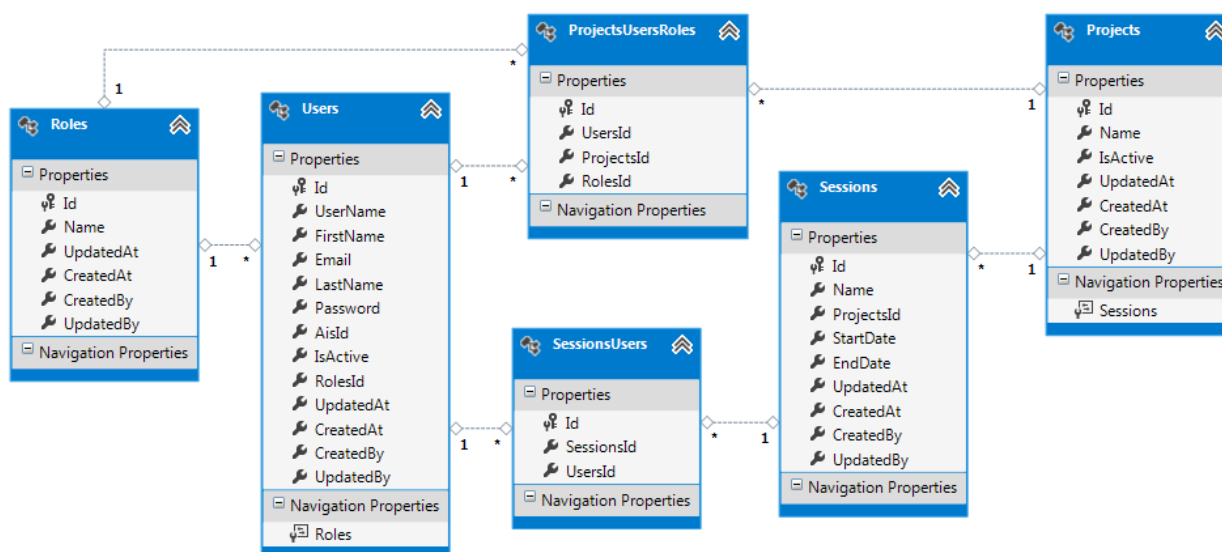
7.8.3 Analýza

Okrem úprav v používateľskom rozhraní bolo treba identifikovať a zdefinovať roly používateľov. K projektu sme sa rozhodli pridávať 2 rôzne typy používateľov:

- projectAdmin - má právo upravovať projekt, pridávať používateľov a sedenia k projektu
- researcher / výskumník - má právo prezerať si detaily projektu a štatistiky nad projektom

7.8.4 Implementácia

Obrázok predstavuje čiastočný fyzický model databázy. Znázorňuje implementované vzťahy medzi používateľom, projektom, sedením a používateľskými rolami.



Obrázok 6 - čiastočný fyzický model databázy

K zobrazovaniu detailov projektu pribudlo zobrazovanie zoznamu pridaných používateľov. K projektu je možné pridať len existujúceho používateľa, preto sme sa rozhodli implementovať vyhľadávanie medzi existujúcimi používateľmi na základe mena. Po potvrdení sa zobrazia používatelia so zhodujúcim sa menom a je možné ich po jednom vybrať a prideliť im rolu. Časom plánujeme pre efektívnejšie vyhľadávanie používateľa implementovať autocomplete. V prvom kroku je funkčné vyberanie a pridávanie používateľa zo všetkých používateľov v databáze, bez vyhľadávania.

7.8.5 Testovanie

Podobne, ako pri pridávaní projektu a sedenia bolo tiež potrebné testovať správnosť údajov, ich formát a komunikáciu s databázou.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klik na záložku Users v rámci detailov projektu	Zobrazí sa zoznam používateľov, pridaných k projektu	Zobrazil sa zoznam používateľov
2.	Klik na tlačidlo Add user	Zobrazí sa zoznam všetkých používateľov, pri každom z možnosťou nastavenia roly a tlačidlom Add	Zobrazil sa zoznam podľa očakávaní
3.	Klik na tlačidlo Add pri používateľovi bez zmeny prednastavenej roly	Používateľ sa pridá k projektu s danou rolou a aplikácia upozorní používateľa hláškou. Tlačidlo add sa zmení na Remove.	K projektu sa pridal používateľ s danou rolou a zobrazila sa hláška, že bol pridaný. Tlačidlo Add sa zmenilo.
4.	Klik na tlačidlo Back to project details	V zozname používateľov pribudne novopridaný	V zozname používateľov bol viditeľný nový používateľ aj so svojou rolou na projekte

Z akceptačného testu vyplýva, že aplikácia správne umožní pridať používateľa k projektu.

7.9 Pridanie používateľa k sedeniu

7.9.1 User story

Výskumník chce pridať k sedeniu používateľa alebo skupinu používateľov.

7.9.2 Úloha

Umožniť výskumníkovi pri úpravách sedenia pridávať používateľov.

7.9.3 Analýza

V tomto prípade je pridávanie používateľa o niečo jednoduchšie, než pre projekt, pretože používatelia, pridaní k sedeniu, majú všetci jednotnú rolu s najnižšími právami. Ide o účastníkov experimentu, ktorý budú používať testovanú aplikáciu, pričom budú sledovaní.

7.9.4 Implementácia

K sedeniu je možné pridávať používateľov 2 spôsobmi: Bud' existujúce skupiny používateľov alebo existujúcich používateľov jednotlivo. Pridávanie používateľov jednotlivo je implementované analogicky ako pri pridávaní používateľa k projektu, okrem vyberania role. Tú majú všetci používatelia, priradení k sedeniu rovnakú. V rámci GUI je pripravené vyhľadávanie a pridávanie skupín používateľov.

7.9.5 Testovanie

Pri testovaní bolo najdôležitejšie kontrolovať práva používateľa (pridávať a editovať sedenia môže len používateľ s právami aspoň projectAdmin) a správnosť vrátených a posielaných údajov z databázy.

8 Deep Purple - 4. šprint

8.1 Komunikácia s browser addonom na zistenie elementov

8.1.1 User story

Výskumník chce vedieť, na aký element sa používateľ pozeral, aby mohol tieto informácie využiť vo svojich experimentoch.

8.1.2 Úloha

Táto úloha sa presunula z minulého šprintu do súčasného. Komunikácia totiž fungovala v poriadku, ale v add-one bolo zistené, že naša metóda prepočítavania súradníc nefunguje tak, ako sme si predstavovali.

8.1.3 Analýza

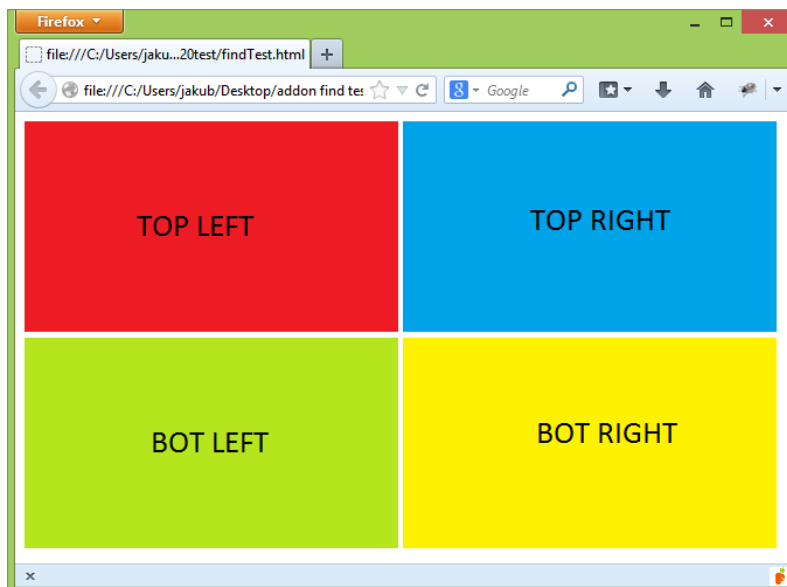
Pri testovaní prepočítavania súradníc bolo zistené, že addon nedokáže priamo pristupovať ku obsahu webovej stránky, ale potrebuje si na to vytvoriť tzv. workerov. Naša metóda ale vytvárala viac workerov na jednu stránku, rozkúskovanú na časti.

8.1.4 Implementácia

Podarilo sa nám nakoniec spraviť funkciu, ktorá vytvára jedného workera, na celú stránku. Workera vytvoríme vždy pre jednu sadu súradníc, po prepočítaní, súradnice pošleme späť na server a workera zničíme.

8.1.5 Testovanie

Pri testovaní bez zariadenia bolo prepočítavanie testované pomocou súradníc získavaných z myši a testovacej stránky vytvorenej na tento účel. Testovacia stránka obsahovala 4 obrázky, ktorých XPath bol vrátený addonom, keď sa naňho používateľ pozeral.



Obrázok 7 - testovacia stránka pre prepočítavanie súradníc

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Spustenie klienta	Klient sa spustí	Klient sa spustil
2.	Testovacie spustenie addonu - spustí sa Firefox s nainštalovaným addonom a konzolou pre testovanie	Firefox sa spustí	Firefox sa spustil
3.	Prechod na testovaciu stránku	Vo Firefoxe sa načíta testovacia stránka	Stránka sa načítala
4.	Presun kurzora myši nad obrázok „TOP LEFT“	V konzolovom okne addonu sa vypíše XPath obrázka „TOP LEFT“	Vypísal sa XPath obrázka „TOP LEFT“
5.	Presun kurzora myši nad obrázok „TOP RIGHT“	V konzolovom okne addonu sa vypíše XPath obrázka „TOP RIGHT“	Vypísal sa XPath obrázka „TOP RIGHT“
6.	Presun kurzora myši nad obrázok „BOT LEFT“	V konzolovom okne addonu sa vypíše XPath obrázka „BOT LEFT“	Vypísal sa XPath obrázka „BOT LEFT“
7.	Presun kurzora myši nad obrázok „BOT RIGHT“	V konzolovom okne addonu sa vypíše XPath obrázka „BOT RIGHT“	Vypísal sa XPath obrázka „BOT RIGHT“

Z tohto testu môžeme vidieť, že addon prepočítava súradnice správne.

8.2 Adaptér pre Klienta

8.2.1 User story

Používateľ chce využívať funkcie nášho klienta, nemá však k dispozícii zariadenie pre sledovanie pohľadu. V takomto prípade potrebuje simulátor, ktorý sa správa ako reálne zariadenie. Pre dosiahnutie pocitu, že sa jedná o reálne zariadenie, je nutné zabezpečiť priehľadnosť simulátora na strane klienta, k čomu by mal slúžiť práve tento adaptér.

8.2.2 Úloha

Vytvorte adaptér pre klienta tak, aby bolo nutné pridať či nahradiť čo najmenej funkcií a metód v rámci klienta a tým dosiahol priehľadnosť adaptéra, a dokázal pritom pracovať so simulátorom.

8.2.3 Analýza

Bolo nutné opäť preštudovať Tobii SDK, tentoraz však na podrobnejšej úrovni. Museli sme identifikovať metódy, s ktorými náš klient pracuje a preštudovať všetky objekty, s ktorými pracuje daná metóda aby bolo možné vytvoriť kópie týchto metód a objektov.

8.2.4 Implementácia

Vytvorili sme dll knižnicu, ktorá simuluje Tobii SDK. To znamená že boli implementované všetky identifikované metódy a objekty, pod rovnakým menom. Nie je teda potrebné prepisovať všetky metódy v zdrojovom kóde pre klienta.

8.2.5 Testovanie

Testovanie sme vykonali pomocou aktuálnej verzie klienta a simulátora tak, že sme zahrnuli knižnicu do projektu ku klientovi, drobnými zmenami prispôbobi zdrojový kód a za pomoci simulátora sme testovali funkcionálnosť klienta.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Spustenie klienta	Klient sa spustí	Klient sa spustil
2.	Vyhľadanie simulátora a jeho zobrazenie v zozname zariadení	Klient nájde a zobrazí údaje o simulátore	Klient našiel a zobrazil údaje o simulátore
3.	Pripojenie na simulátor	Klient sa pripojí na simulátor, bude možné spustiť sledovanie pohľadu	Klient sa pripojil na simulátor, ktorý umožnil sledovanie pohľadu
4.	Spustenie kalibrácie	Po spustení sa zobrazí správa, že kalibrácia bola úspešná	Po spustení sa zobrazila správa o úspešnej kalibrácii
5.	Spustenie sledovania pohľadu a zaznamenávanie dát	Spustí sa sledovanie pohľadu, budeme získavať dáta	Sledovanie pohľadu sa spustilo a dáta boli zaznamenávané

Z tohto testu môžeme vidieť, že implementácia adaptéra prebehla úspešne.

8.3 Vytvorenie nového používateľa

8.3.1 User story

Zákazník potrebuje webovú aplikáciu kde by sa mohlo prihlasovať viacero používateľov, preto chce mať prostredie pre vkladanie používateľov.

8.3.2 Úloha

Pridanie používateľa.

8.3.3 Analýza

Hľadal som riešenia pre automatické vytváranie formulára z modelu a jQuery validácie na formulári.

8.3.4 Implementácia

Na tento prístup som našiel viacero riešení ale použil som v stavané funkcie v MVC pre validáciu formulárových polí z modelu.

8.3.5 Testovanie

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Odoslanie formulára	Zapísanie dát na server	Dáta boli zapísané do tabuľky users
2.	Testovanie nevyplneného password	Výpis chyby pri poli password	Chyba bola vypísaná
3.	Testovanie nevyplneného email	Výpis chyby pri poli email	Chyba bola vypísaná

Z tohto testu môžeme vidieť, že pridávanie používateľa funguje správne.

8.4 Autorizácia používateľa

8.4.1 User story

Zákazník chce získať možnosť personalizovať webovú aplikáciu podľa rolí a zobrazovať časti stránky relevantným používateľom

8.4.2 Úloha

Vytvoriť logiku autorizácie v MVC.

8.4.3 Analýza

Zistoval som typy autorizácie v MVC. Zistil som že existujú vstavané knižnice na autorizáciu a autentifikáciu do session a cookies ale aj externé knižnice

8.4.4 Implementácia

Zvolil som možnosť implementovania vstavanej funkcie.

8.4.5 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie uloženia roli po prihlásení	Uložená rola do cookies	Dáta boli uložené po prihlásení
2.	Testovanie výpisu používateľského mena a ID z cookies	Vypísané používateľského mena a ID z cookies	ID aj meno vypísané
3.	Testovanie obmedzenia prístupu podľa role	Nezobrazenie Admin sekcie ak nemá práva ADMIN	Admin sekcia nezobrazená

Z tohto testu môžeme vidieť, pridávanie autorizácia funguje správne.

8.5 Zmena stavu sedenia

8.5.1 User story

Experimentátor chce zmeniť stav sedenia, tak aby o tom testovaný používatelia vedeli.

8.5.2 Úloha

Umožniť zmenu sedenia cez webové rozhranie, a na klientovi zabezpečiť informovanosť a to tak aby si používateľ vedel vybrať ku ktorému sedeniu posiela dáta.

8.5.3 Analýza

Bolo potrebné overiť PUSH notifikácie zo strany servera. Popri analýze som prišiel na to že by bolo potrebné vytvoriť nový service na klientovi na ktorý by sa notifikácia poslala, nakoľko HTML v momentálnej verzii ešte tieto PUSH notifikácie nepodporuje.

8.5.4 Implementácia

Zmenu stavu sedenia sa pridala k editovaniu sedenia ktoré už bolo naimplementované. Do tabuľky sa pridalo pole isActive ktoré reprezentuje či je aktuálne sedenie aktívne. Na klientovi bolo potrebné naimplementovať metódu na posielanie dopytov v určitom časovom intervale. V odpovedi na dopyt sa posiela list aktívnych sedení ktorých členom je aktuálne prihlásení používateľ. Aby sa dopyty neposielali zbytočne stále, doimplementovalo sa tlačítko po ktorom sa dopyty začnú posielat'. Zobrazí sa okno v ktorom sa zobrazia všetky aktívne sedenia.

8.5.5 Testovanie

Implementácia bola testovaná zmenami stavu sedení a vypnutou službou.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klikneme na tlačidlo session	Zobrazí sa list sedení	Zobrazí sa list sedení
2.	Zmeníme stav sedenia počas zobrazeného listu sedení	Sedenie sa buď z listu odstráni alebo pridá v závislosti od zmeny stavu	Sedenie sa buď z listu odstráni alebo pridá v závislosti od zmeny stavu
3.	Vypneme service počas posielania dopytov	Zobrazí sa informujúca správa o nedostupnosti servera	Zobrazí sa informujúca správa o nedostupnosti servera

Z tohto testu môžeme vidieť, že funkcia pracuje správne aj v prípade, že nenájde aktuálny server na ktorý posiela dopyty, a aj v prípade zmeny sedenia počas otvoreného okna.

8.6 Ukladanie dát zo sledovania pohľadu

8.6.1 User Story

Experimentátor chce ukladať dáta zo sledovania na servery pre neskoršie analyzovanie.

8.6.2 Úloha

Zanalyzovanie možných spôsobov ukladania dát na servery, implementácia funkcií na servery a vytvorenie tabuľky na skladovanie dát.

8.6.3 Analýza

Bolo potrebné zistiť možné spôsoby ukladania veľkého množstva dát do databázy nakoľko to bola požiadavka vedúceho tímu. Pre možnosť neskoršej zmeny formy dát je najvhodnejšie do databázy ukladať serializované objekty. V prípade zmeny dát (pridá sa nový atribút) je deserializácia možná bez akejkoľvek úpravy. Do databázy sa ukladajú skupiny objektov ktoré majú určité atribúty spoločné a tie charakterizujú dané skupiny.

8.6.4 Implementácia

Na servery bola upravená metóda na spracovanie dát nakoľko doteraz sa dáta zahadzovali a nikam sa neukladali. Vytvorili sme tabuľku v databáze na skladovanie dát. Určili sme štruktúru a charakteristické vlastnosti.

Štruktúra tabuľky:

```
[Key]
[DataMember]
public int Id { get; set; }
[DataMember]
[ForeignKey("SessionDto")]
public int SessionsId { get; set; }
[DataMember]
[ForeignKey("UserDto")]
public int UsersId { get; set; }
[DataMember]
public string StartTimestamp { get; set; }
[DataMember]
public string EndTimestamp { get; set; }
[DataMember]
public string GazeData { get; set; }
[DataMember]
public string WinData { get; set; }
```

Reťazce GazeData a WinData sú serializované listy dát. Dáta sa ukladajú do listu a každú sekundu sú odoslané na server.

8.6.5 Testovanie

Funkcie boli testované pri vypnutom servery, ako aj pri nedoslaní potrebných špecifických údajov pre dané dáta vid'. SessionID a UserID.

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klikneme na tlačidlo Start tracking pričom nebolo vybrané sedenie ku ktorému dáta patria	Zobrazí sa upozornenie pre vybratie sedenia	Zobrazí sa upozornenie pre vybratie sedenia
2.	Server nie je dostupný	Zobrazí sa hlásenie o nedostupnosti servera	Zobrazí sa hlásenie o nedostupnosti servera
3.	Prepneme sedenie do stavu inactive počas odosielania dát	Zobrazí sa informujúca správa ukončení sedenia	Zobrazí sa informujúca správa ukončení sedenia

8.7 Pridanie používateľa k projektu

8.7.1 User story

Výskumník chce pridať používateľa k vytvorenému projektu používateľa a nastaviť alebo zmeniť mu práva.

8.7.2 Úloha

Dopracovať vyhľadávanie používateľa pri pridávaní používateľa k projektu, umožniť administrátorovi zmeniť jeho rolu na projekte.

8.7.3 Analýza

V predchádzajúcom šprinte táto úloha nebola úplne dokončená, kvôli neočakávaných problémom, ktoré sa pri implementácii vyskytli. Okrem toho bolo potrebné dopracovať vyhľadávanie medzi používateľmi a umožniť tiež menenie roly používateľa a projekte.

8.7.4 Implementácia

Pomocou javascriptu a jQuery sme implementovali zobrazovanie a miznutie zoznamu používateľov. Používateľ pri začatí pridávania používateľa uvidí len textové pole, kam napíše meno používateľa, ktorého chce vyhľadať a potvrdzovacie tlačidlo „Search“. Javascriptom je tiež volaná metóda, ktorá umožní vyhľadanie medzi používateľmi. Ukážka zdrojového kódu:

```
//zobrazenie najdenych userov pri pridavani usera k session a projektu
$("#btn-search-user").click(function() {
    //ziskanie premennych
    var userName = $(".user-name").attr("value");
    $.ajax({
        type: "Get",
        dataType: "json",
        url: "FindUsers",
        data: "name="+userName, //'{ "name": '+ userName +' }',
        success: function (data) {
            //naplnenie tabulky userov
            var table = document.getElementById("table-users-found");
            var content = '';
            for (var i = 0, size = data.length; i < size; i++) {
                content += '<tr>';
                content += '<td>' + data[i].UserName + '</td>';
                content += '<td>' + data[i].FirstName + '</td>';
                content += '<td>' + data[i].LastName + '</td>';
                content += '<td>' + data[i].Email + '</td>';
                content += '<td>';
                content += '<select class="edit-mode" id="Role" name="Role">';
                content += document.getElementById("role-invisible").innerHTML;
                content += '</td>';
                content += '<td>';
                content += '<button onclick="addUser()" class="btn btn-success btn-sm adduser">Add</button>';
                content += '<input type="hidden" id="userid" value='+ data[i].Id
                +'/>';

                content += '</td>';
                table.innerHTML += '</tr>';
            }
            table.innerHTML = content;
        },
        error: function (request, error) {
            console.log(arguments);
            alert(" Can't do operation because: " + error);
        }
    });
});
```

```

    }
  });
  $("#users-found").show();
});

```

8.7.5 Testovanie

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klik na tlačidlo Add user	Zobrazí sa len textové pole a tlačidlo Search	Zobrazilo sa len textové pole a tlačidlo Search
2.	Zadanie textu a klik na Search	Zobrazí sa zoznam používateľov, ktorých meno, login alebo priezvisko obsahuje zadaný reťazec	Zobrazil sa zoznam vyfiltrovaných používateľov

Akceptačný test ukázal, že filtrovanie používateľov pri ich pridávaní k projektu funguje správne.

8.8 Pridanie používateľa k sedeniu

8.8.1 User story

Výskumník chce pridať používateľa k vytvorenému sedeniu.

8.8.2 Úloha

Dopracovať vyhľadávanie používateľa pri pridávaní používateľa k sedeniu, pripraviť pridanie skupiny používateľov. Umožniť upravovať aj ďalšie detaily sedenia.

8.8.3 Analýza

V predchádzajúcom šprinte táto úloha nebola úplne dokončená, podobne ako pridávanie používateľa k projektu, bola čistočne presunutá do ďalšieho šprintu. Bolo potrebné dopracovať vyhľadávanie medzi používateľmi a najmä upraviť editovanie a grafické rozhranie sedení.

8.8.4 Implementácia

Analogicky, ako pri projekte, vyhľadávanie medzi používateľmi je implementované pomocou javascriptu a jQuery. Používateľ pri začatí pridávania používateľa uvidí len textové pole, kam napíše meno používateľa, ktorého chce vyhľadať a poztvrdzovacie tlačidlo „Search“. Pridané bolo tiež tlačidlo Add users group, ktoré predkladá implementáciu pridávania skupín používateľov. Okrem toho bolo implementované menenie atribútov sedenia, medzi nimi aj zmena jeho stavu.

8.8.5 Testovanie

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klik na tlačidlo Add user	Zobrazí sa len textové pole a tlačidlo Search	Zobrazilo sa len textové pole a tlačidlo Search
2.	Zadanie textu a klik na Search	Zobrazí sa zoznam používateľov, ktorých meno, login alebo priezvisko obsahuje zadaný reťazec	Zobrazil sa zoznam vyfiltrovaných používateľov
3.	Zmena zobrazených údajov sedenia a kliknutie na tlačidlo Save	Uložia sa zmeny, vykonané nad sedením a prejavia sa	Zmeny sa uložili a boli viditeľné pri prezeraní detailov sedenia
4.	Kliknutie na tlačidlo Back	Vrátenie sa na detaily projektu, ktorý upravujeme	Používateľ bol vrátený na stránku príslušného upravovaného projektu

Akceptačný test ukázal, že filtrovanie používateľov pri ich pridávaní k sedeniu funguje správne, rovnako ako ukladanie ostatných zmien v rámci sedenia.

8.9 Prihlásenie používateľa cez LDAP

8.9.1 User story

Výskumník chce, aby sa používatelia mohli prihlasovať pomocou AIS, aby nemusel vytvárať ručne konto pre každého používateľa.

8.9.2 Úloha

Zabezpečiť možnosť overovania používateľov prostredníctvom AIS loginu a hesla.

8.9.3 Implementácia

Pre možnosť overenia používateľa prostredníctvom AIS loginu a hesla bola do projektu implementovaná funkcia `IsAuthenticated(String user, String password)` na overenie záznamu LDAP.

```
public bool IsAuthenticated(String user, String password)
{
    bool authenticated = false;
    String uid = "uid=" + user + ",ou=people,dc=stuba,dc=sk";
    String _path = "LDAP://ldap.stuba.sk";

    DirectoryEntry de;
    de = new DirectoryEntry(_path, uid, password, AuthenticationTypes.None);
```

8 Deep Purple - 4. Šprint

```
try
{
    object connected = de.NativeObject;
    authenticated = true;
}
catch (Exception ex)
{
}

return authenticated;
}
```

Pri overovaní používateľa sa najskôr overuje, či sa nachádza v lokálnej databáze.

V prípade o používateľovi nie je v databáze záznam, autentifikácia skončí neúspechom.

Ak sa používateľ v databáze nachádza, skontroluje sa či má definované heslo. Ak má namiesto zahashovaného hesla v hodnotu null, je nutné overiť používateľa prostredníctvom LDAP. Výsledok overenia volania funkcie `IsAuthenticated(String user, String password)` je potom výsledkom celej autentifikácie. Ak je v databáze definované heslo, pokračuje sa štandardným overením zahešovaného poskytnutého hesla so zahešovaným heslom v databáze. Ak sa rovnajú, používateľ je autentifikovaný.

8.9.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	pokus o prihlásenie používateľa s menom a heslo v databáze	prihlásený	prihlásený
2.	pokus o prihlásenie používateľa s AIS menom a heslom, ktorý je evidovaný v databáze	prihlásený	prihlásený
3.	pokus o prihlásenie používateľa s AIS menom a heslom, ktorý nie je evidovaný v databáze	neprihlásený	neprihlásený
4.	pokus o prihlásenie používateľa s menom ktoré nie je v databáze	neprihlásený	neprihlásený

8.10 Ukladanie a načítanie oblastí záujmu

8.10.1 Úloha

Umožnenie načítania a zobrazenia uložených oblastí záujmu pre danú stránku a uloženie nových/upravených oblastí záujmu.

8.10.2 Implementácia

Do projektu boli pridané nové časti. Menovite `AreaOfInterestRepository`, `AreaOfInterestDTO`, do dátového modelu boli pridané tabuľky `AreasOfInterest` a `SessionsAreasOfInterests`.

`AreaOfInterestRepository` definuje operácie nad oblast'ami záujmu ako načítanie oblastí, načítanie konkrétnej oblasti podľa id, načítanie oblastí podľa id sedenia, ku ktorému su priradené, pridanie oblasti, editovanie oblasti, zmazanie oblasti.

Jednotlivé operácie sú definované aj vrámci REST volaní, ktoré sú následne využívané pri MVC, ale aj pri manipulácií s oblast'ami záujmu addonom.

8.10.3 Testovanie

Overenie funkčnosti jednotlivých funkcií bolo vykonávané prostredníctvom volania REST služieb pomocou Google Chrome aplikácie Postman, ktorá umožňuje jednoduché posielanie get a post požiadaviek na server.

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	načítanie oblastí záujmu	načítanie všetkých oblastí	očakávaná reakcia
2.	načítanie oblasti záujmu podľa id	načítanie konkrétnej oblasti	očakávaná reakcia
3.	načítanie oblastí podľa id sedenia	načítanie oblastí priradených požadovanému sedeniu	očakávaná reakcia
4.	pridanie oblasti záujmu	nová oblasť uložená na serveri	očakávaná reakcia
5.	editovanie oblasti záujmu	zmena atribútov oblasti uložená na serveri	očakávaná reakcia
6.	zmazanie oblasti záujmu	nastavenie <code>isActive</code> na <code>false</code>	očakávaná reakcia

8.11 Podpora pre usporiadanie, filtrovanie a stránkovanie dát v tabuľke

8.11.1 Úloha

Umožnenie manipulácie zobrazenia záznamov v tabuľkách, s podporou zoradenia podľa definovaného stĺpca tabuľky a stránkovanie záznamov po definovaných kvantách..

8.11.2 Implementácia

Úloha bola implementovaná využitím dostupného rozšírenia .NET s názvom WebGrid, ktoré umožňuje automatické vytváranie tabuliek z dodaného dátového modelu. Funkcia pre vykreslenie tabuľky umožňuje určiť niekoľko parametrov, ako pridanie CSS štýlov jednotlivých častí, prioritné zoradenie tabuľky, počet záznamov na stránku, určenie jednotlivých stĺpcov a ich parametrov, prípadne formátu ich obsahu.

8.12 Správa oblasti záujmu

8.12.1 User story

Ako výskumník, chcem mať možnosť zobrazit' oblasti záujmu pre zvolení sedenie, aby som ich mohol upravovať.

8.12.2 Úloha

Rozšírenie webového rozhrania o možnosti manipulácie s oblast'ami záujmu.

8.12.3 Implementácia

Pridaním kontroléra a zobrazenia do MVC bola rozšírená funkcionálna webového rozhrania. Kontrolér využíva už vytvorený AreaOfInterestRepository a jeho definované funkcie na manipuláciu s oblast'ami záujmu. Implementácia umožňuje po zavolaní kontroléra AreaOfInterest a zobrazenia Areas zobrazit' zoznam všetkých oblastí záujmu.

Jednotlivé oblasti je možné editovať stlačením tlačidla Edit pri konkrétnej oblasti v tabuľke. Aktivuje sa zobrazenie pre editáciu, kde je možné menit' atribúty Name, XPath, Page, Description a IsActive pre zvolenú oblasť záujmu.

Zobrazenie pre editáciu je využité aj pri zobrazení session a jej oblastí záujmu.

d'alej bolo definované rozhranie na zmazanie (deaktivovanie) oblasti záujmu, pričom po stlačení tlačidla delete sa vytvorí dialógové okno s potvrdením akcie vymazania pre oblasť so zvoleným id.

8.12.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	zobrazenie zoznamu oblastí	zobrazenie oblastí	zobrazenie oblastí
2.	zobrazenie editácie konkrétnej oblasti	zobrazenie formuláru	zobrazenie formuláru

3.	zobrazenie editácie neexistujúcej oblasti	zobrazenie hlášky o neexistujúcej oblasti	zobrazenie hlášky o neexistujúcej oblasti
4.	vykonanie a uloženie zmien oblasti	prejavenie zmien	prejavenie zmien
5.	zmazanie oblasti záujmu	otvorenie potvrdzovacieho dialógu	otvorenie potvrdzovacieho dialógu
6.	potvrdenie zmazania v dialógu	zmazanie oblasti	zmazanie oblasti
7.	zrušenie zmazania v dialógu	ponechanie oblasti	ponechanie oblasti

8.13 Zadefinovanie oblastí záujmu a odoslanie na server

8.13.1 User story

Ako výskumník chcem zadefinovať oblasti záujmu prostredníctvom addonu a odoslať ich na server, aby som nemusel ručne písať XPath adresy oblastí.

8.13.2 Úloha

Rozšírenie addonu o funkcionality uchovávanie a spracovávanie oblastí záujmu. Umožniť pridávanie, editovanie a odoberanie oblastí záujmu.

8.13.3 Implementácia

Rozšírenie bolo implementované vytvorením prototypov funkcií pre javascript pole, ktoré umožňujú pridávanie nových, aktualizovanie editovaných oblastí záujmu a mazanie oblastí záujmu.

Jednotlivé akcie sú vyvolávané z dialógového okna - panelu, v ktorom sa zobrazuje formulár pre nastavenie atribútov oblastí záujmu. Panel sa zobrazí po kliknutí na požadovaný element stránky. V Panely sa automaticky vyplnia atribúty Url a XPath, ktoré nie je možné editovať.

Panel umožňuje vykonanie akcií uloženie zmien (pridanie oblasti, aktualizovanie oblasti), alebo zmazanie oblasti. Ak používateľ klikne na už definovanú oblasť, v panely sa zobrazia jej atribúty.

Definované oblasti sú zvýraznené zeleným orámovaním.

8.13.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	pridanie novej oblasti	pridaná oblasť	pridaná oblasť

8 Deep Purple - 4. Šprint

2.	editovanie pridanej oblasti	načítanie pôvodných atribútov a uloženie vykonaných zmien	načítanie pôvodných atribútov a uloženie vykonaných zmien
3.	pridanie inej oblasti	pridané 2 rôzne oblasti	pridané 2 rôzne oblasti
4.	editovanie prvej oblasti	prejavenie zmien v prvej oblasti	prejavenie zmien v prvej oblasti
5.	zmazanie prvej oblasti	zmazanie oblasti bez ovplyvnenia iných oblastí	zmazanie oblasti bez ovplyvnenia iných oblastí
6.	zrušenie panelu	žiadna zmena oblasti	žiadna zmena oblasti

9 Evanescence - 5. Šprint

9.1 Logovanie

9.1.1 User story

Používateľ chce vidieť vo webovej aplikácii všetky operácie vykonávané v reálnom čase v kóde pri používaní aplikácie, aby mohli vývojári vidieť chyby pri nasadení.

9.1.2 Úloha

Vytvorenie logovania v MVC.

9.1.3 Analýza

Vytvorenie logovania v MVC je možné riešiť už vstavanou knižnicou pre MVC, ale tiež aj svojimi metódami alebo knižnicami tretích strán.

9.1.4 Implementácia

Použil som vstavané funkcie pre asp.net MVC, ktorý podporuje logovanie na IIS a tiež trace aplikácie.

9.1.5 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Testovanie trace zobrazenia	Výpis operácií z webovej aplikácie	Operácie boli úspešne zobrazené v trace.axd stránke

9.2 Autentifikácia REST volaní

9.2.1 User story

Zákazník chce, aby dáta na servery boli uložené bezpečne, bez možnosti neautentifikovaného zobrazenia.

9.2.2 Úloha

Autentifikácia REST volaní.

9.2.3 Implementácia

Všetky metódy, ktoré sú volané cez REST boli zabezpečené, a ich volanie je možné jedine po autentifikovaní. Predbežne je pridaná aj možnosť autorizácie, ktorá zatiaľ nie je plne implementovaná. Do hlavičky HTTP dopytu bol pridaný autentifikačný atribút v podobe hesla a prihlasovacieho mena používateľa. Pred volaním funkcie sa tieto údaje overia a až potom je možné vykonať danú metódu. Pri autentifikovaní sa vracia id role, ktorej je používateľ členom, čo je možné použiť pri riešení autorizácie.

9.2.4 Testovanie

Testovanie prebiehalo posielaním HTTP dopytov na server (POST aj GET).

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Odoslanie POST dopytu na server	Vrátenie HTTP odpovede 401 Unauthorized	Vrátenie HTTP odpovede 401 Unauthorized
2.	Odoslanie GET dopytu na server	Vrátenie HTTP odpovede 401 Unauthorized	Vrátenie HTTP odpovede 401 Unauthorized

9.3 Pridanie používateľa k projektu

9.3.1 User story

Výskumník chce pridať používateľa k vytvorenému projektu používateľa a nastaviť alebo zmeniť mu práva.

9.3.2 Úloha

Doimplementovať smenu roly používateľa na projekte.

9.3.3 Analýza

Používatelia môžu mať rámci projektu rôzne roly, je vhodné umožniť ich menenie aj po pridaní daného používateľa nielen počas jeho pridávania k projektu.

9.3.4 Implementácia

Medzi používateľmi v zozname po kliknutí na tlačidlo edit role sa pomocou javascriptu používateľovi - administrátorovi sprístupní možnosť zmeniť rolu. Akonáhla zmeny uloží, aktualizuje sa záznam v databáze, v tabuľke ProjectsUsersRoles, pre konkrétneho používateľa na konkrétnom projekte.

9.3.5 Testovanie

Vykonaný bol nasledovný akceptačný test:

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Klik na tlačidlo Edit role	Zobrazí sa kontrolka na výber novej roli	Zobrazila sa kontrolka na výber novej roli, obsahujúca možnosti existujúcich rolí
2.	Potvrdenie novej roly	Zobrazí sa zoznam používateľov s aktualizovaným údajom	Zobrazil sa zoznam používateľov, pričom ten, ktorého rola bola upravená, ju mal aktualizovanú

Akceptačným testom sme overili, že zmena rolí používateľov v rámci projektov je funkčná.

9.4 Ukladanie a načítanie oblastí záujmu pre konkrétny projekt a sedenie

9.4.1 User story

Ako výskumník, chcem vedieť nastaviť addonu id sedenia prostredníctvom web rozhrania správy sedení, aby som vedel editovať oblasti záujmu pre konkrétne sedenie prostredníctvom addonu.

9.4.2 Úloha

Rozšírenie funkcionality webového rozhrania a addonu o možnosť definovania id sedenia, pre ktorú bude addon editovať oblasti záujmu. Po zmene id sedenia je nutné aktualizovať oblasti záujmu, s ktorými je možné pracovať.

9.4.3 Implementácia

Do stránky sedení bola pridaná tabuľka s oblasťami záujmu pre zvolené sedenie a tlačidlo, ktorého kliknutie vyvolá nastavenie id sedenia v addone a jeho následnú aktiváciu pre editovanie oblastí záujmu.

Tlačidlo obsahuje v atribúte id sedenia, ktoré má byť zvolené. V addone je vytvorený PageMod modul, ktorý na stránke vyhľadáva aktivačné tlačidlo a priraduje mu action listener. Po zistení stlačenia tlačidla sa aktivuje funkcia spracovania udalosti, ktorá načítá id sedenia z atribútu tlačidla a odošle správu hlavnému skriptu addonu a zaznamenaní akcie pre zmenu id sedenia a aktivovanie módu pre editovanie.

Aktivácia módu zahŕňa načítanie oblastí záujmu ktoré boli definované pre zvolené sedenie a ich zobrazenie na relevantných stránkach. Addon umožňuje editovanie existujúcich a pridávanie nových oblastí s tým, že zmeny sa synchronizujú priamo so serverom. Ak sa pridáva nová oblasť, po jej vložení sa zavolá metóda na priradenie pridanej oblasti k aktuálne nastavenému sedeniu.

Kliknutím na ikonu addonu je možné mód editovania zrušiť a následne obnoviť s tým, že id sedenia ostáva nezmenené až do stlačenia tlačidla vo webovom rozhraní na stránke správy sedenia.

9.4.4 Testovanie

Číslo kroku	Akcia	Očakávaná reakcia	Skutočná reakcia
1.	Priradenie id sedenia a aktivácia addonu	zmena id, aktivovanie módu editácie, načítanie zodpovedajúcich oblastí	očakávaná reakcia
2.	Editovanie načítanej oblasti záujmu	uloženie zmien na serveri	očakávaná reakcia
3.	Pridanie novej oblasti záujmu pre sedenie	vloženie novej oblasti záujmu na server a jej priradenie k sedeniu	očakávaná reakcia
4.	Zmazanie načítanej oblasti záujmu	Nastavenie oblasti záujmu ako neaktívnej (nebude znovu načítaná)	očakávaná reakcia
5.	Znovupriradenie id iného sedenia	Aktualizovanie načítaných oblastí pre novo zvolené sedenie	očakávaná reakcia
6.	Deaktivácia editácie kliknutím na ikonu addnou	Zrušenie zobrazovaných oblastí záujmu	očakávaná reakcia
7.	Aktivácia editácie kliknutím na ikonu addonu	Znovunačítanie oblastí záujmu posledne zvoleného sedenia	očakávaná reakcia

9.5 Prihlásenie sa cez browser add-on

9.5.1 User story

Ako používateľ sa chcem vedieť prihlásiť cez browser addon, aby som umožnil zbieranie dát o mojej aktivite na počítači

Ako výskumník sa chcem vedieť prihlásiť cez browser addon, aby som vedel definovať oblasti záujmu.

9.5.2 Úloha

Vytvorenie mechanizmu prihlasovania sa v browser addone. Ak nie je používateľ prihlásený cez addon, po kliknutí na addon sa mu v prehliadači otvorí nový tab s prihlasovacím formulárom.

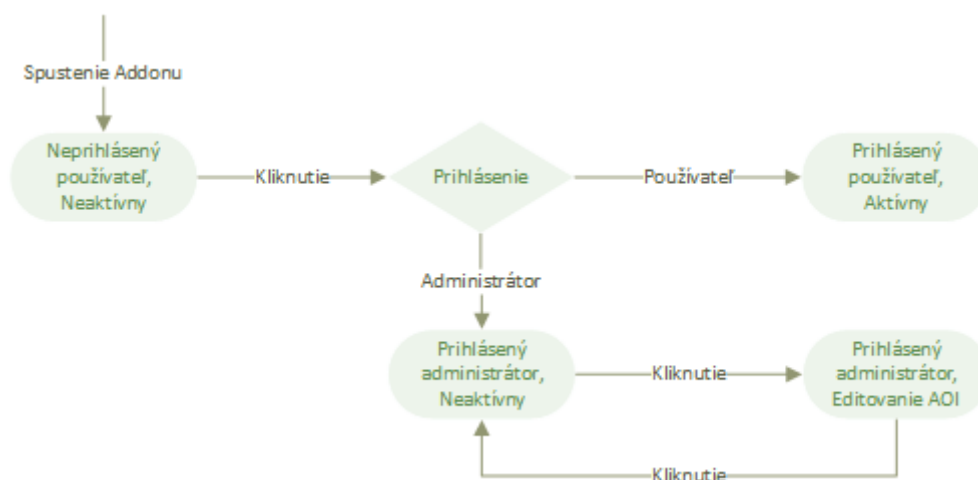
Po prihlásení si addon zapamätá používateľa.

9.5.3 Implementácia

Prihlasovanie bolo pridané do addonu v nasledujúcej podobe. Ak nie je v addone definovaný používateľ, po vykonaní akcie kliknutia na ikonu addonu, je používateľovi zobrazený nový tab s prihlasovacou stránkou. Addon je rozšírený o nový modul PageMod, ktorý sa aktivuje pri načítaní stránky po prihlásení a získa id používateľa a jeho rolu (momentálne v simulovanej podobe, údaje sú podhodnené javascriptom ktorý sa spustí po načítaní preddefinovanej stránky).

Addon rozlišuje prihlásenie používateľa a administrátora, teda toho, kto môže editovať oblasti záujmu. V rámci tohoto rozšírenia addon podporuje 4 stavy činnosti, ktorým vždy prislúcha vlastná ikona:

- 👤 Neprihlásený používateľ, Neaktívny
- 👤 Prihlásený používateľ, addon môže spracovávať dopyty od klienta
- 👤 Prihlásený administrátor, addon je neaktívny
- 👤 Prihlásený administrátor, editovanie oblastí záujmu



Obrázok 8 - Súčasná implementácia stavov addonu na základe prihláseného používateľa

PRÍLOHA A - Výstupy analýz

A.1 ACDC - 1. šprint

A.1.1 Formát dát zo zariadenia

Čo obsahuje packet získaný zo zariadenia:

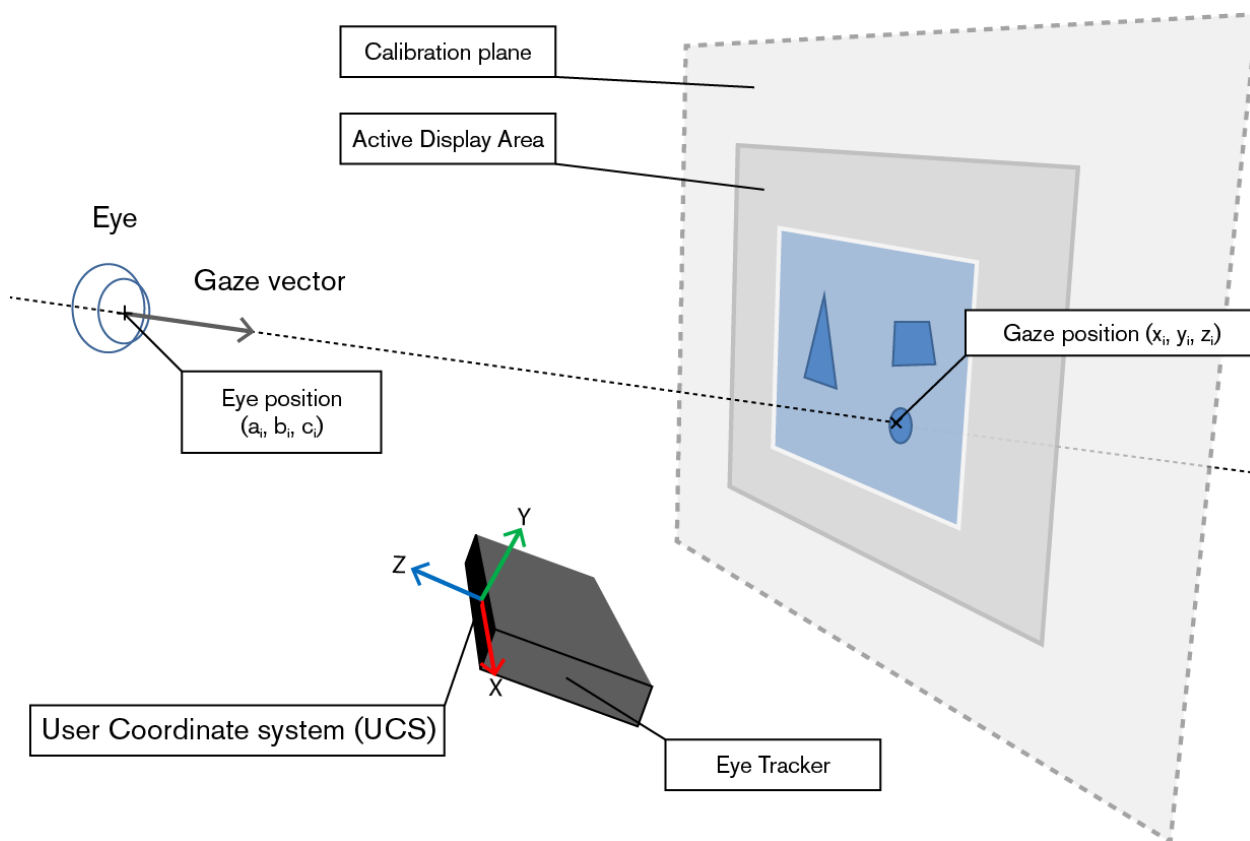
- Timestamp
- Polohu očí (UCS)
- Relatívnu polohu očí
- 3D bod pohľadu
- 2D bod pohľadu
- Validáčny kód
- Priemer zreničky

Timestamp:

Čas vygenerovania packetu zariadením. Zdrojom sú interné hodiny zariadenia. Preto je potrebné synchronizovať tieto časy s PC.

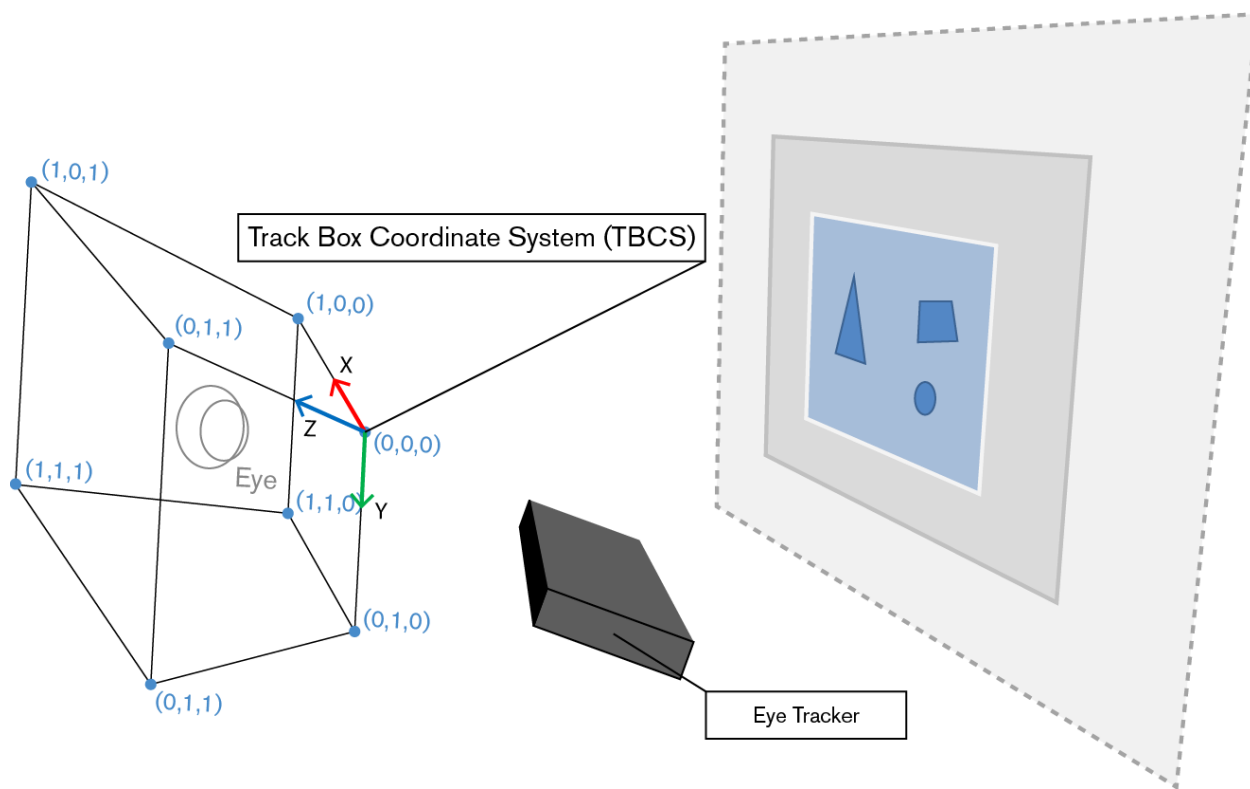
Polohu očí:

Dáta sú poskytované zvlášť pre ľavé a pravé oko. Hodnoty x, y, z v UCS - User coordinate system. Hodnoty sú vyjadrené v milimetroch.



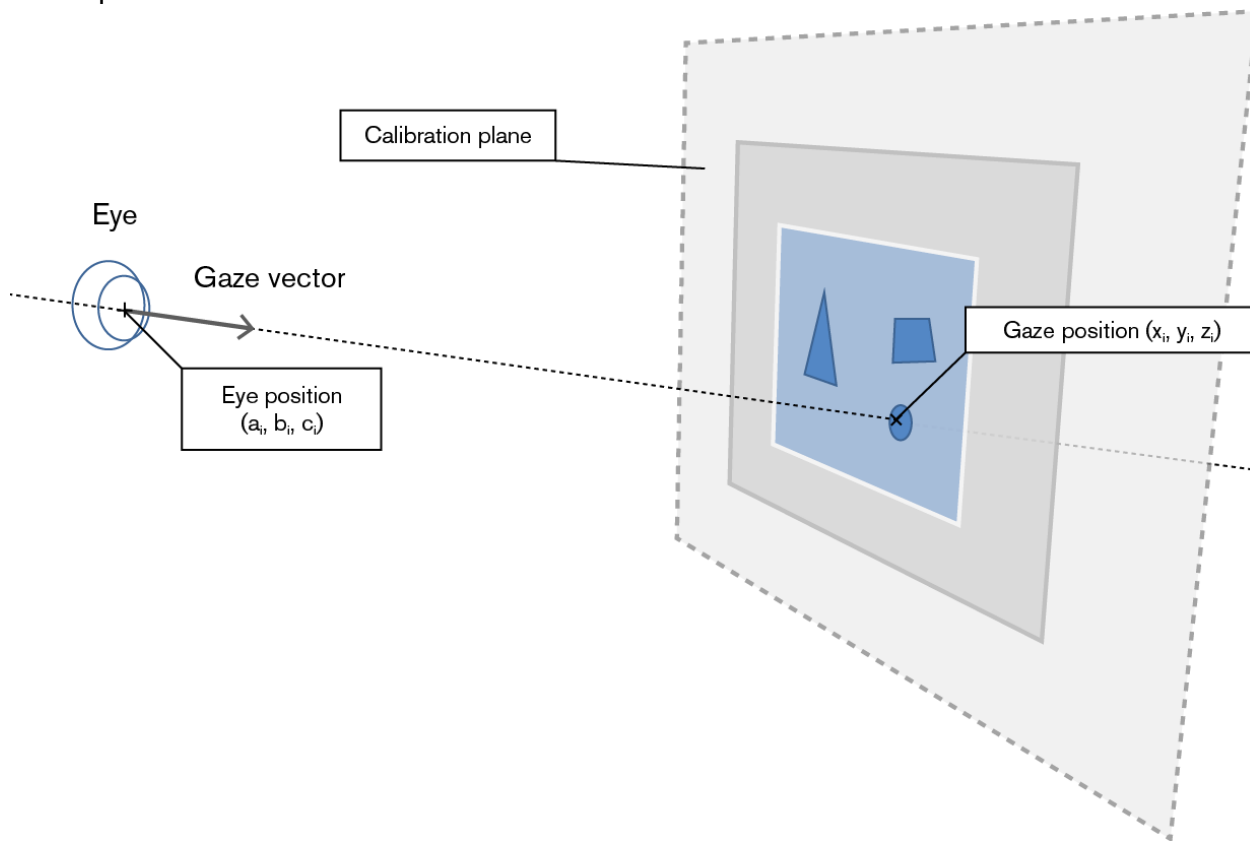
Relatívna poloha očí:

Dáta sú poskytované zvlášť pre ľavé a pravé oko. Je to poloha očí v tzv. TrackBox, čo je virtuálna krabica, v ktorej dokáže zariadenie snímať oči. Hodnoty sú normalizované, v rozmedzí od 0 do 1.



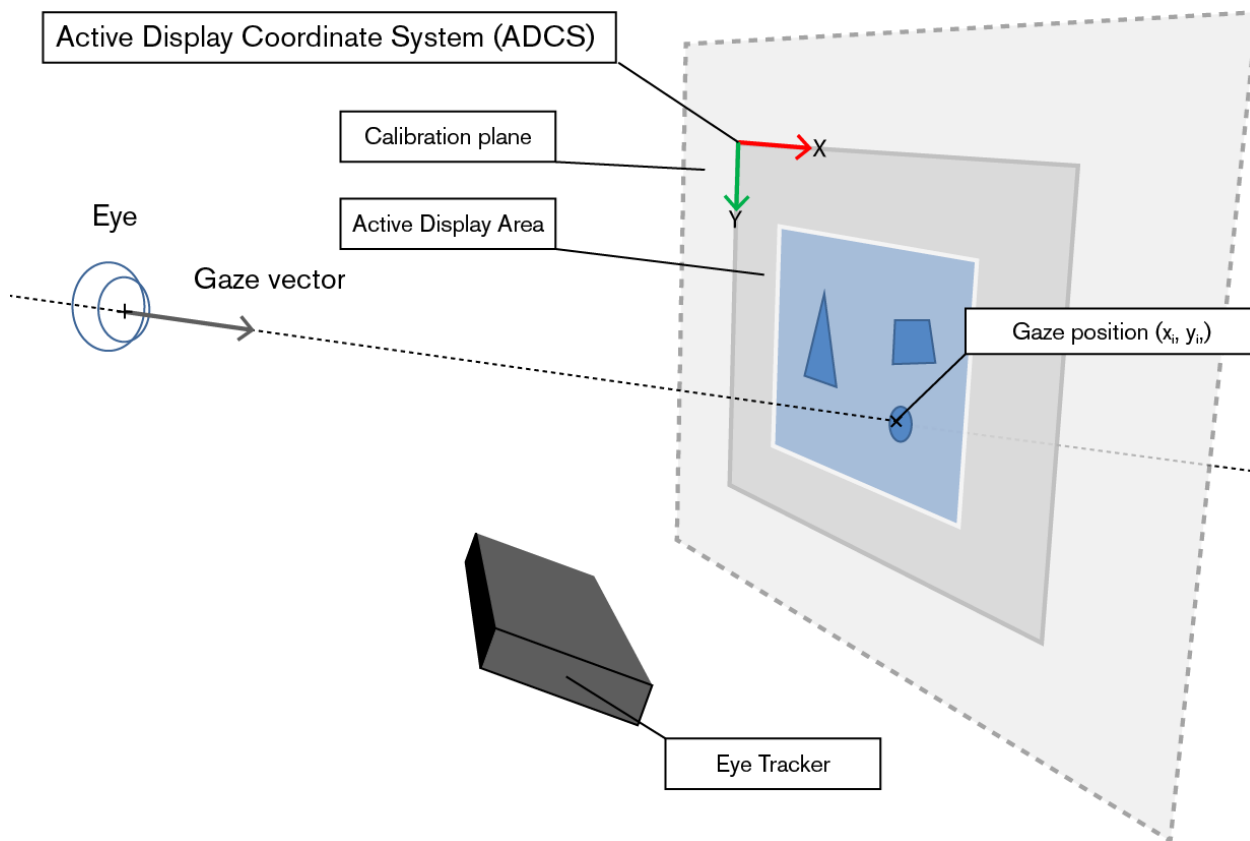
3D bod pohľadu:

Dáta sú poskytované zvlášť pre ľavé a pravé oko. Hodnoty opäť v UCS. Opisuje prienik "calibration plane" a pomyslenej čiary, vychádzajúcej z oka, ktorá má rovnaký smer ako vektor pohľadu.



2D bod pohľadu:

Dáta sú poskytované zvlášť pre ľavé a pravé oko. Predstavujú v podstate rovnaké dáta ako 3D bod pohľadu, sú však vyjadrené ADCS - Active Display Coordinate System ako dvojrozmerný bod. Hodnoty opäť normalizované. Bod (0,0) predstavuje horný, ľavý roh "Active Display Area", bod (1,1) zase dolný, pravý roh.



PRÍLOHA A - Výstupy analýz

Validačný kód:

Vyjadruje to, ako si je zariadenie isté tým, že dáta, ktoré pochádzajú z daného oka, z neho naozaj pochádzajú. Hodnoty sú v rozpätí 0-4.

Oko	Pravé oko	-//-	-//-	-//-	-//-	-//-
Ľavé oko	Hodnota	0	1	2	3	4
-//-	0	Našlo obe oči	N/A	N/A	N/A	Našlo jedno oko. Najpravde podobnejšie ľavé oko
-//-	1	N/A	N/A	N/A	Našlo jedno oko. Pravdepodobne ľavé.	N/A
-//-	2	N/A	N/A	Našlo jedno oko. Nevie ktoré.	N/A	N/A
-//-	3	N/A	Našlo jedno oko. Pravdepodobne pravé.	N/A	N/A	N/A
-//-	4	Našlo jedno oko. Najpravde podobnejšie pravé.	N/A	N/A	N/A	Nenašlo ani jedno oko

Priemer zreničiek:

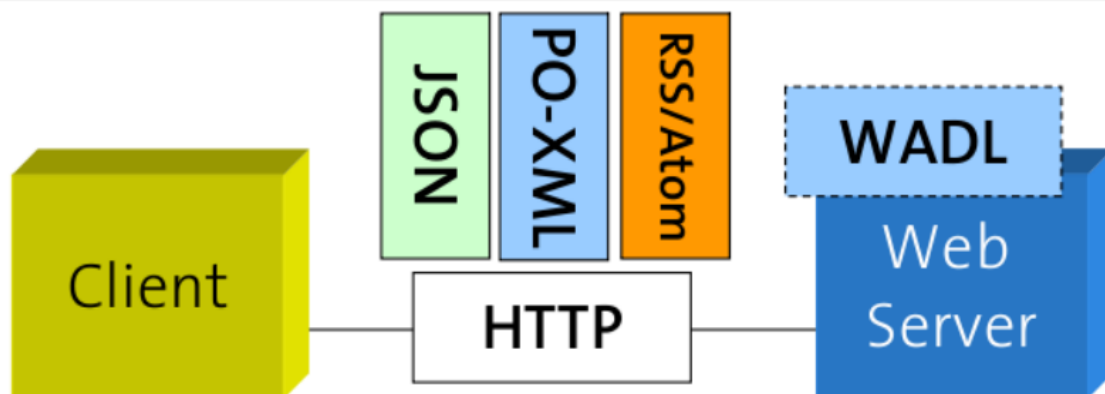
Dáta sú poskytované zvlášť pre ľavé a pravé oko. Hodnoty predstavujú veľkosť zreničky v milimetroch.

A.1.2 Rest Vs Socket

Koncept

Representational State Transfer (REST) je koncept pre design distribuovanej architektúry.

RESTful Web Services (2007)



Princíp

1. URI - identifikácia zdroj (všetko je zdroj)
2. CRUD - jednotné rozhranie pre všetky zdroje
3. Reprezentácie - rôzne podoby správy (MIME)
4. Bezstavovosť - umožňuje škálovateľnosť - Neexistuje HTTP Session!, stav cez linky
5. Hypermédiá - prelinkovanie médií/reprezentácií

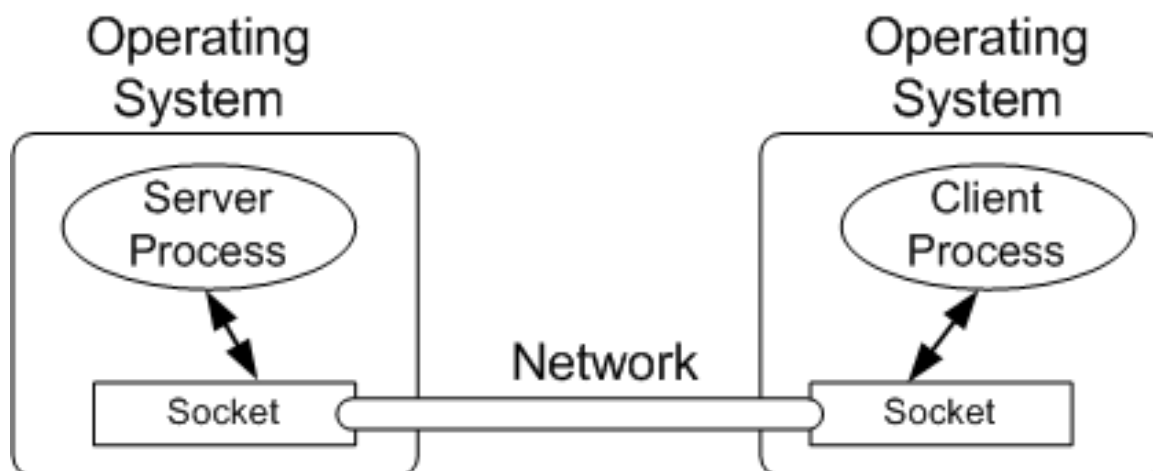
výhody:

1. jednoduchosť,
2. cacheable,
3. jednotné rozhranie pre viacero zdrojov
4. klient server - http protocol
5. škálovateľnosť medzi komponentami
6. Použite get, post, put atď.
7. json serializer

nevýhody

1. viazaný na http
2. veľké množstvo objektov
3. správa URI namespaceov môže byť ťažkopádna - závisí na architektovi

Sockety



Architektúra je rovnaká ako pri použití REST-u rozdielny je len spôsob komunikácie, nakoľko REST funguje na aplikačnej vrstve TCP/IP modelu sockety na transportnej a teda využívajú TCP respektíve UDP protokol. Ich fungovanie je jednoduché nastaví sa porty na klientskej a serverovej strane cez ktoré bude komunikácia prebiehať, na toto je potrebné vytvoriť pre poslúchanie zvlášť vlákno ktoré zachytáva prijímajúce sockety.

Výhody

- zbavíme sa payloadu http hlavičky
- prispôsobenie a škálovateľnosť
- komunikácia nie je založená na REQUEST/REPLY
- menší traffic

Nevýhody

- potrebné otvorenie portu, nakoľko na servery nemáme túto možnosť. Program by teda trebalo napísať tak, že by rátal s otvoreným portom 80 pre http komunikáciu.
- oproti REST-u oveľa zložitejšia implementácia
- bezpečnosť

A.2 Beatles - 2. šprint

A.2.1 WinApi funkcie

V tomto dokumente je opísané ako používať konkrétne WinApi funkcie v jazyku C#. Jedná sa hlavne o funkcie pracujúce s umiestnením a stavom okien vo Windows. Ku každej funkcii je krátky opis toho na čo slúži a syntax pre implementáciu v C#.

Poznámky na úvod

Handle okna - int - pointer na adresu, kde je "uložené okno"

Return - tu je napísané, čo funkcie vracajú

C# syntax - môžete nakopírovať do zdrojového kódu

Príklad použitia - návrh, na čo by sa dala funkcia použiť a jej jednoduché použitie v kóde
okno má focus = okno je aktivované

GetForegroundWindow

Pomocou tejto funkcie dostaneme handle okna, ktoré má práve focus - je v popredí.

Return: handle - ukladať do IntPtr

C# syntax:

```
[DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling = true)]  
public static extern IntPtr GetForegroundWindow();
```

Príklad použitia:

Pomocou Process získate MainWindowHandle nejakého procesu a porovnaním s hodnotou vrátenou touto funkciou, môžeme zistiť, či má okno focus.

```
IntPtr handle = GetForegroundWindow();
```

GetWindowRect

Funkcia vracia obdĺžnik, opisujúci veľkosť a umiestnenie daného okna. Pre túto funkciu je potrebné vytvoriť štruktúru, obsahujúcu 4 premenné integer - left, top, right a bottom. Tieto premenné označujú, kde je vrch, spodok, ľavá a pravá strana obdĺžnika.

Return: obdĺžnik opisujúci okno - štruktúra Rect

```
struct Rect
{
    int left, top, right, bottom;
}
```

C# syntax:

```
[DllImport("user32.dll")]
[return: MarshalAs(UnmanagedType.Bool)]
static extern bool GetWindowRect(IntPtr hwnd, out Rect lpRect);
```

Príklad použitia:

Získaním súradníc pohľadu vieme jednoducho zistiť, či sa používateľ pozerá na okno.

```
IntPtr handle = GetForegroundWindow();
Rect windowBoundingBox;
//funkcia naplní inty v Rect-e hodnotami
GetWindowRect(handle, out windowBoundingBox);
```

GetWindowPlacement

V štruktúre window placement vracia viac informácií o okne - pozíciu, informácie o viditeľnosti atď.

Return: štruktúra window placement, popísaná nižšie

```
struct WindowPlacement
{
    public int length;
    public int flags;
    public int showCmd;
    public Point ptMinPosition;
    public Point ptMaxPosition;
    public Rect rcNormalPosition;
}

struct Point
{
    int x;
    int y;
}
```

length - dĺžka tejto štruktúry v bytoch, pred volaním funkcie ju treba naplniť pomocou Marshal.sizeof();

WindowPlacement wp;

wp.length = Marshal.sizeof(wp);

flags - bitové flags pre ovládanie pozície minimalizovaného okna a metódy, ktorá ho obnovuje

flag 0x0004

WPF_ASYNCWINDOWPLACEMENT - ak volajúci thread (metódy, ktorá obnovuje okno) nie je rovnaký ako thread, ktorému patrí okno, systém preklopí request do tohto threadu, aby volajúci thread nebol blokován

flag 0x0002

WPF_RESTORETOMAXIMIZED - okno sa po obnovení maximalizuje, aj keď predtým nebolo, funguje iba na prvé následné obnovenie. Platí iba ak je v showCmd nastavená hodnota SW_SHOWMINIMIZED (showCmd opísané nižšie)

flag 0x0001

WPF_SETMINPOSITION - ak je nastavený, môžeme špecifikovať koordináty minimalizovaného okna v ptMinPosition

showCmd - aktuálny zobrazovací stav okna, môže nadobúdať nasledovné hodnoty (v zozname je názov hodnoty a samotná hodnota)

- SW_HIDE = 0
 - schová okno a zobrazí namiesto neho iné
- SW_MAXIMIZE = 3
 - maximalizuje okno
- SW_MINIMIZE = 6
 - minimalizuje okno a aktivuje ďalšie top-level okno v z-poradí (okno, ktoré bolo “za ním”)
- SW_RESTORE = 9
 - aktivuje a zobrazí okno, ak bolo minimalizované/maximalizované, obnoví ho do pôvodnej veľkosti a pozície
- SW_SHOW = 5
 - aktivuje a zobrazí okno v jeho aktuálnej veľkosti a pozícii
- SW_SHOWMAXIMIZED = 3
 - aktivuje a zobrazí okno maximalizované
- SW_SHOWMINIMIZED = 2
 - aktivuje a zobrazí okno ako minimalizované

PRÍLOHA A - Výstupy analýz

- SW_SHOWMINNOACTIVE = 7
 - zobrazí okno ako minimalizované, ale neaktivuje ho
- SW_SHOWNA = 8
 - rovnaké ako SW_SHOW, ale neaktivuje
- SW_SHOWNOACTIVE = 4
 - podobné ako SW_SHOWNORMAL, ale neaktivuje
- SW_SHOWNORMAL = 1
 - rovnaké ako SW_RESTORE, ale používa sa, keď aplikácia chce okno zobrazit' prvýkrát

ptMinPosition - súradnice ľavého horného bodu okna, keď je minimalizované

ptMaxPosition - súradnice ľavého horného bodu okna, keď je maximalizované

rcNormalPosition - súradnice okna (obdĺžnik), v jeho obnovenom stave (keď nie je maximalizované ani minimalizované). Má stále rovnaké hodnoty, aj keď okno maximalizujeme alebo minimalizujeme.

príklad použitia:

chceme zistiť v akom stave je okno

```
Process[] processList = Process.GetProcessesByName("firefox");
```

```
IntPtr handle = processList[0].MainWindowHandle;
```

```
WindowPlacement wp;
```

```
wp.length = Marshal.sizeof(wp);
```

```
GetWindowPlacement(handle, out wp);
```

```
if(wp.showCmd == 0)
```

```
{
```

```
    //rob niečo, ak je okno schované
```

```
}
```