Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

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

Obsah

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 nedoká žu 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 pod poru pre testovanie dynamických webových aplikácií. Vzniká teda potreba pre vytvorenie platformy pre zber, spracovanie a
vyhodnocovanie údajov zo sledovania pohľadu z viacerých zariadení (pracovných sta níc).

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 spísane 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 centrálne 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 spojazdniť 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 funkcionalita pre sledovanie zariadení. Takto by mal výskumník zistiť, či dáta od testera získal, či je možné, že získané dáta môžu byť nespoľahlivé kvôli zlej kalibrácií zariadenia. Táto funkcionalita 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 získane 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 pozerať 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 funkcionalita by mala výskumníkovi zobraziť v prvom rade informácie o tom, či sú používatelia prihlásený 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.

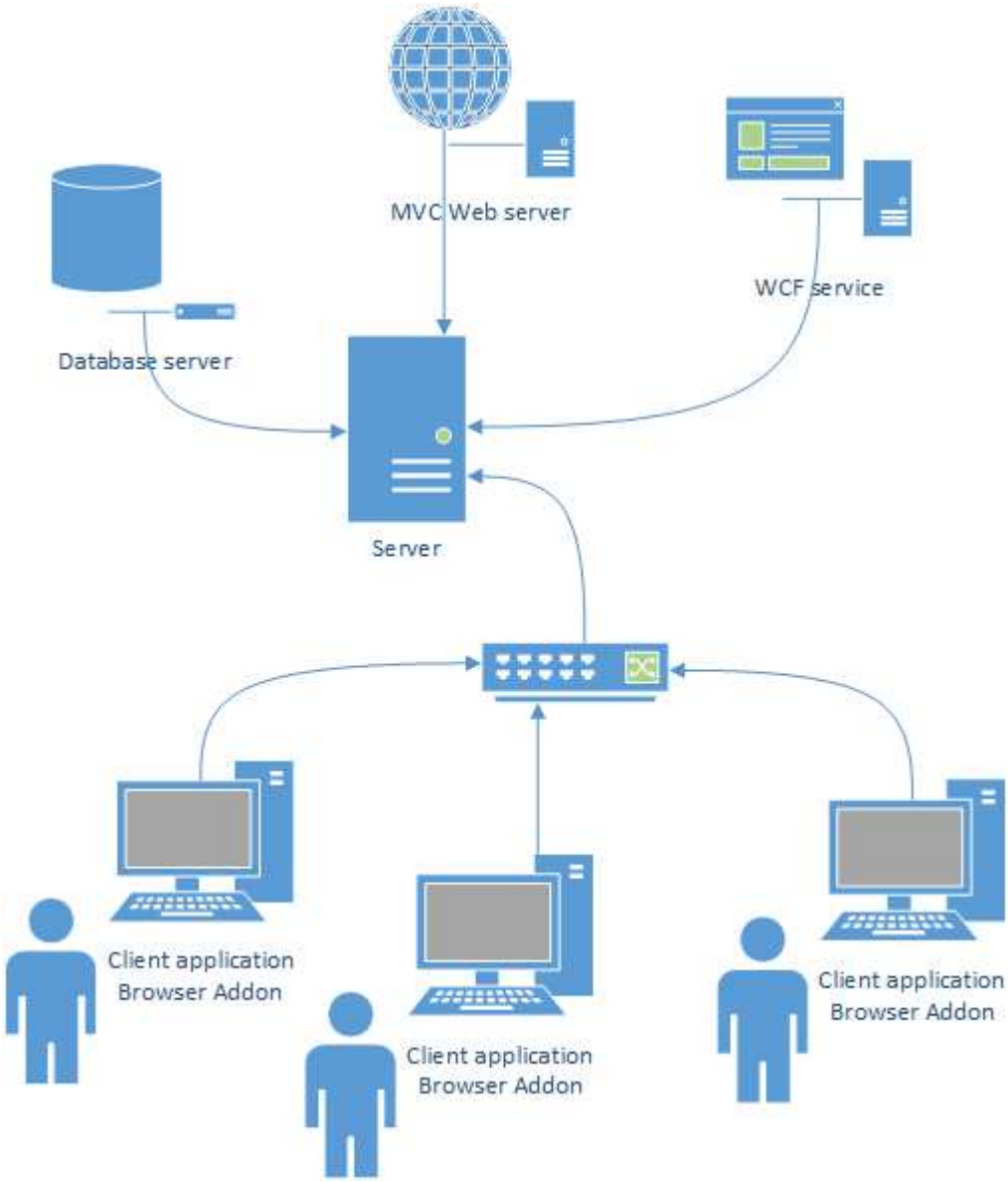
Vizualizáciu by sme chceli zo začiatku implementovať zobrazovaním grafov, vytváraných z nazbieraných dát. Tieto grafy pomôžu výskumníkom orientovať sa v nazbieraných dátach z experimentov a jednoduchšie robiť z experimentov závery.

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 informácii 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, *prezerať experimenty*, ktorých sa zúčastnil. "Výskumník" má právo si *prezerať experimenty*, ktorých je súčasťou, *prezerať získané dáta* zo sedení, patriacich k danému experimentu a *viesť 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.

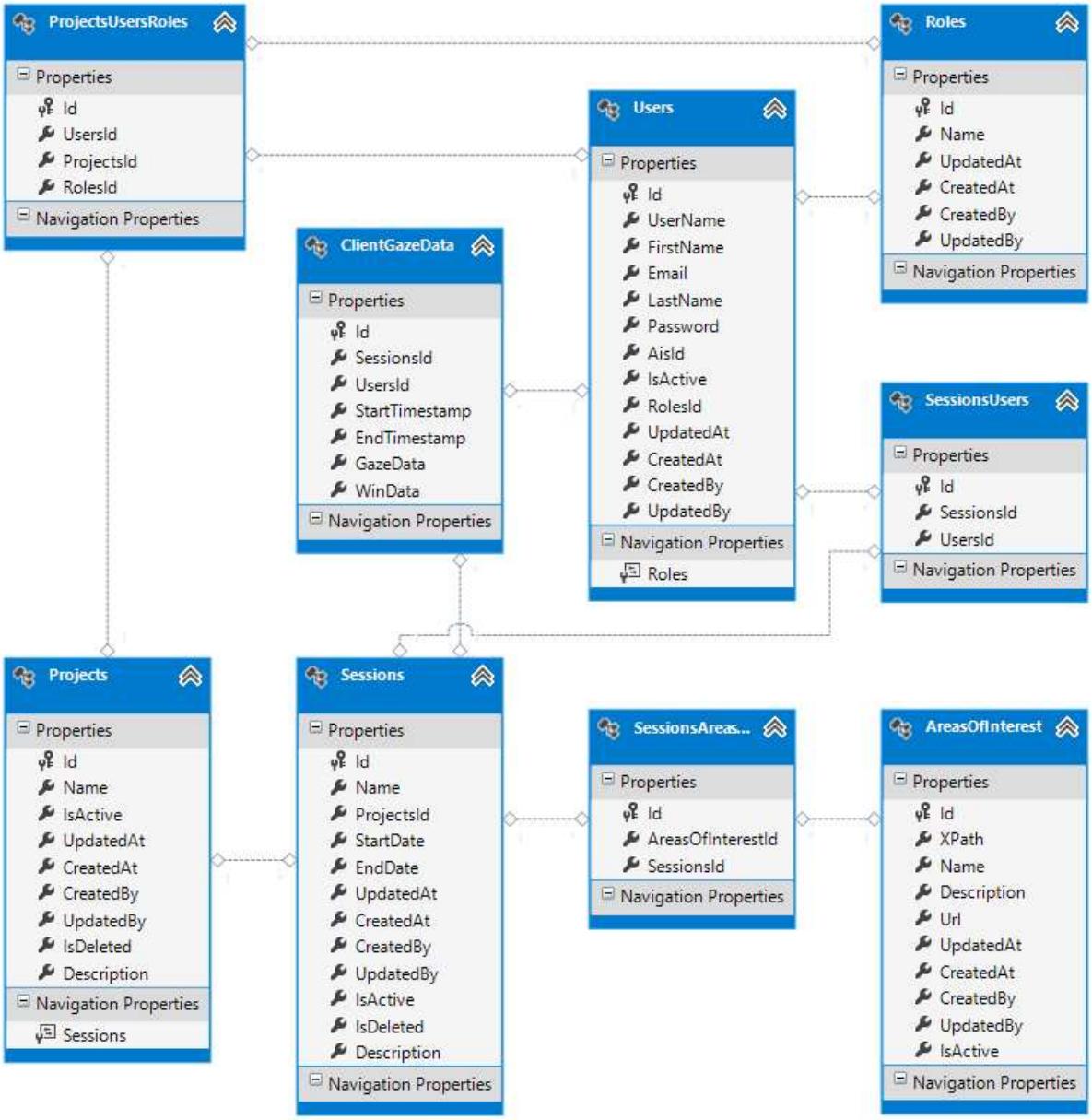
4.2.3 Rozšírenie pre webový prehliadač

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 funkcionalitu. V tomto prípade jeho funkcionalitu 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.



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 detaily, ak má dostatočné práva, môže k projektu pridávať sedenia (Sessions), iných používateľov s rôznymi rolami (Roles) a pridávať používateľov k sedeniam.

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

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

    - 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

```
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 sledovat
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 kluč reprezentujúci sedenie ku ktorému dáta patria
int UsersId - cudzí kluč 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 vyšie 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 zadefinovať area of interest.

5.1.2 Úloha

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

5.1.3 Analýza

Analyzovaním dostupnej funkcionality JQuery, ale aj samotného JavaScriptu sa dospelo k záveru, že požadovaná funkcionalita 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";
  }
```

```
  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 kalibrovať 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 aplikácii 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 pozerať 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. Anotáciou 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 inštalácií
    - 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 jednoprocesorový 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 > ukážkový plugin pre prehliadače
    - **crossbrowser:** podobný online framework, chýba 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 getXPathFromPos(x,y) na požadovaný element	XPath adresa požadovaného elementu	XPath adresa požadovaného elementu
2.	zavolanie funkcie getXPathFromPos(x,y) 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 kompletné 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.

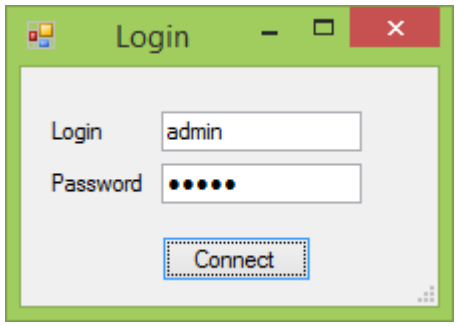
6.2.3 Analýza

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.

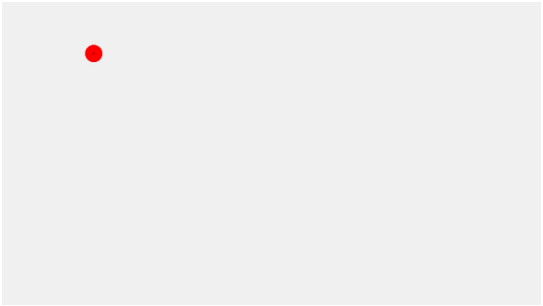
6.2.4 Implementácia

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.

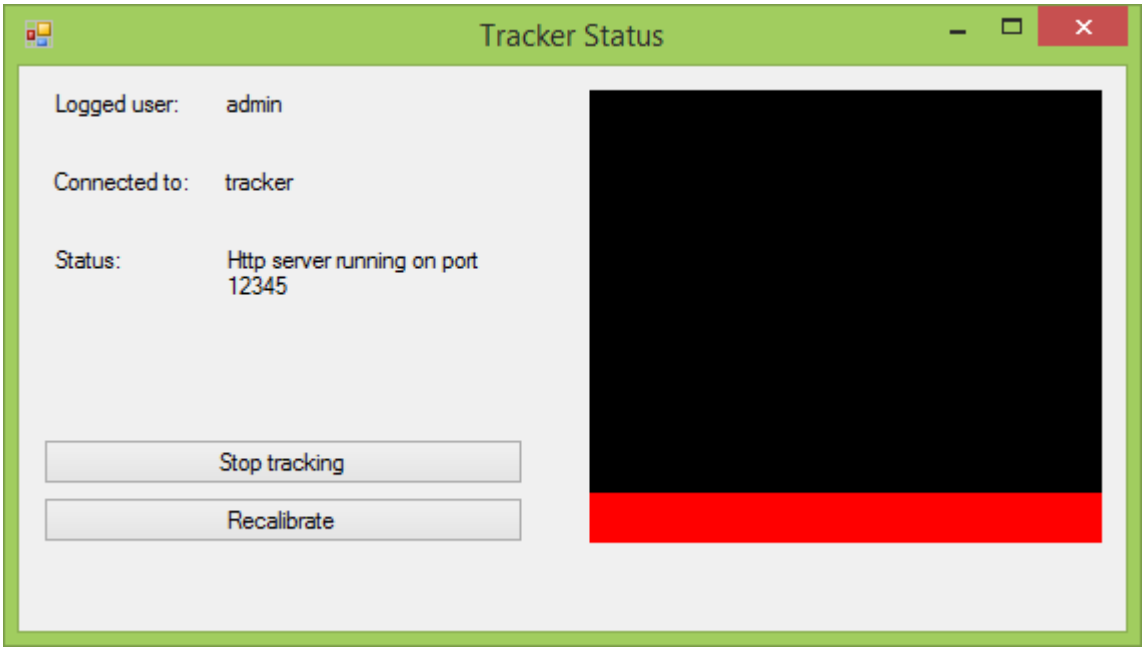
Používateľské rozhranie vyzerá nasledovne:



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 zápäti 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 funkcionalitu zvýrazňovania elementov pri prechádzaní myšou nad daným elementom. Ako inšpirácia slúži funkcionalita prieskumníka kódu vo Firefoxe.

Požiadavky na riešenie:

- ohraničiť oblasť tak, aby nebolo ovplyvnené rozloženie stránky
- zobraziť 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();
```

```
  // 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 responzívne. 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 spúsi, a začne odosielať packety obsahujúce výrobné údaje o simulátore	Simulátor sa spustil a odosielal 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 vypĺňať 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 roli vo formulári	zobrazené roli vo formulári

6.8 Zadefinovanie 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 zadefinovať 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 Zadefinovanie 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 4627[1]. Pri navrhnutí bolo potrebné vychádzať s požiadaviek od zákazníka, a teda odosielať dáta ohľadom pohľadu používateľa ako aj informácie priamo o danom používateľovi a prebiehajúcom 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

```
        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 posielať 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:

- navrhnúť 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 Framowork 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 používatelia 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 odosielajú 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 zvýraznia	Projekt sa nevytvoril, povinné polia sa zvýraznili
5.	Pokus o vytvorenie projektu - nesprávne vyplnenie polí (zlý formát)	Projekt sa nevytvorí, nesprávne vyplnené polia sa zvýraznia 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 aplikácií výskumníkovi vytvoriť konkrétny experiment – sedenie.

6.12.3 Analýza

Na realizáciu tejto úlohy bolo potrebné:

- navrhnúť 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 vlastnoti 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 poziciovaný 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 akcií, 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 Boostrapu tvoriť moderný web s profesionálnym vzhľadom.

Klady použitia frameworku

- nie je nutné navrhovať štýlovanie 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 Bootsrap 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 isAuthentificated(String user, String password)
{
  bool authentificated = 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;
    authentificated = true;
    //Console.WriteLine("Login successful.");
  }
  catch (Exception ex)
  {
    //Console.WriteLine("Login failed.");
  }

  return authentificated;
}
```

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 paginácu dát v tabuľke

7.2.1 User story

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

7.2.2 Úloha

Implementovať funkcionalitu 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 funkcionalita 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 zázanamov podľa stĺpca	preusporiadanie zázanamov podľa stĺpca
2.	opačné usporiadanie tabuľky podľa aktuálne definovaného stĺpca	opačné preusporiadanie zázanamov podľa stĺpca	opačné preusporiadanie zázanamov 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 zadefinovaný area of interest.

7.3.2 Úloha

Zabezpečiť komunikáciu medzi desktopovým klientom a browser add-onom pre prepočítavanie 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 posielať 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 posielať 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 funkcionalitu 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 súvisi 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 GazeDataItem a naplní ju rozparsovanými 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 javascirpt 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 používatelia v tabuľke	Zobrazení používatelia 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

```
[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.

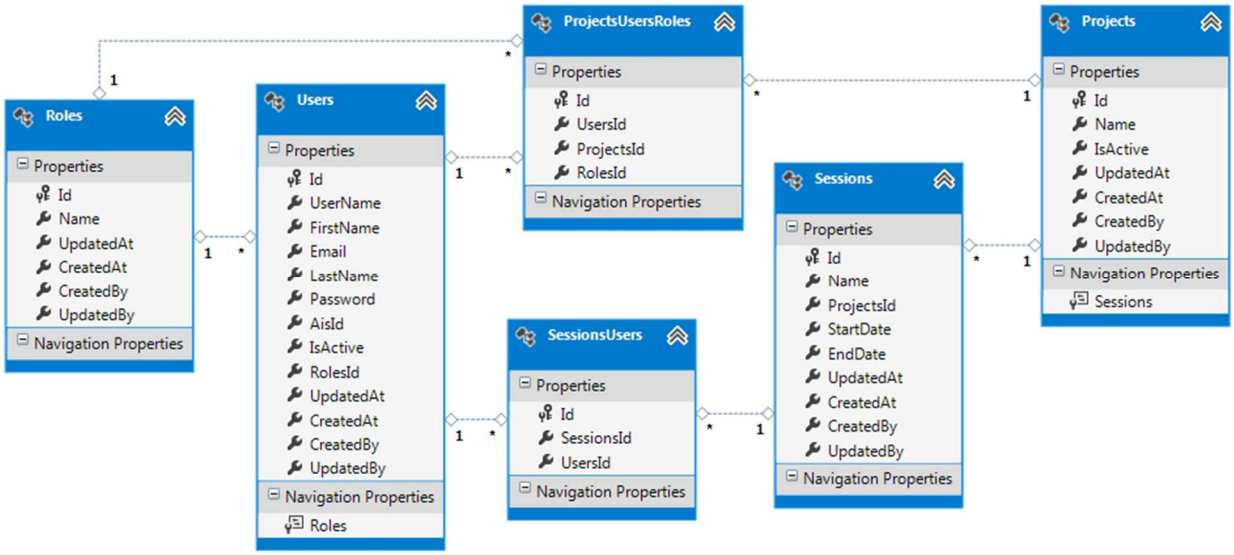
7.8.3 Analýza

Okrem úprav v používateľskom rozhraní bolo treba identifikovať a zadefinovať 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: Buď 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. Vrámci 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

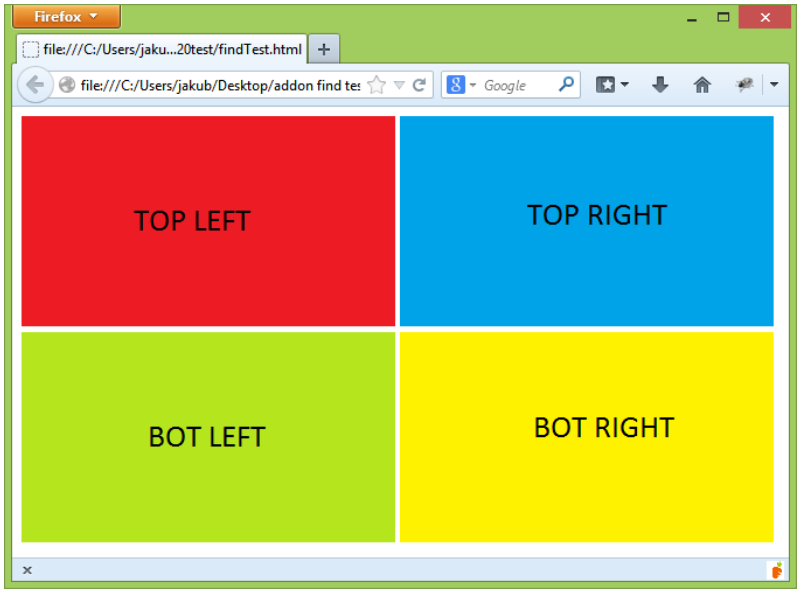
Pri testovaní prepočítavania súradníc bolo zistené, že addon nedokáže priamo pristupovať ku obsahu webovej stránky, ale potrebuje si na to vytvoriť tzv. workerov. Naša metóda ale vytvárala viac workerov na jednu stránku, rozkúskovanú na časti.

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 dispozícií 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 prispôsobili zdrojový kód a za pomoci simulátora sme testovali funkcionalitu 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 kalibrácií
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 mat prostredie pre vkladanie používateľov.

8.3.2 Úloha

Pridanie používateľa.

8.3.3 Analýza

Hľadal som riešenia pre automatické vytvárania 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 zapísanie 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 autorizacie v MVC. Zistil som že exstistúju vstavané knižnice ne 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 posielať. 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 novy 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 neodoslaní potrebných špecifických údajov pre dané dáta viď. 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 poztvrdzovacie 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 mediz 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 predokladá implementáciu pridávania skupín používateľov. Okrem toho bolo implementované menenie atribútov sedenia, medzi nimmi 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 `IsAuthentificated(String user, String password)` na overenie záznamu LDAP.

```
public bool IsAuthentificated(String user, String password)
{
    bool authentificated = 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);
```

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

    return authentificated;
}
```

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 `IsAuthentificated(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 oblasťami 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 oblasťami 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 isActive na false	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 zobraziť oblasti záujmu pre zvolení sedenie, aby som ich mohol upravovať.

8.12.2 Úloha

Rozšírenie webového rozhrania o možnosti manipulácie s oblasťami záujmu.

8.12.3 Implementácia

Pridaním kontroléra a zobrazenia do MVC bola rozšírená funkcionalita webového rozhrania. Kontrolér využíva už vytvorený AreaOfInterestRepository a jeho definované funkcie na manipuláciu s oblasťami zújmu. Implementácia umožňuje po zavolaní kontroléra AreaOfInterest a zobrazenia Areas zobraziť 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 meniť atribúty Name, XPath, Page, Description a IsActive pre zvolenú oblasť záujmu.

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

ďalej 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	zmazaní 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 funkcionalitu uchovávania a spracovávania 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

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, akutoalizuje 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 priraďuje mu action listener. Po zistení stlačenia tlačidla sa aktivuje funkcia spracovania udalosti, ktorá naťíta 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 obastí 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 oblstí 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 oblastí 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.

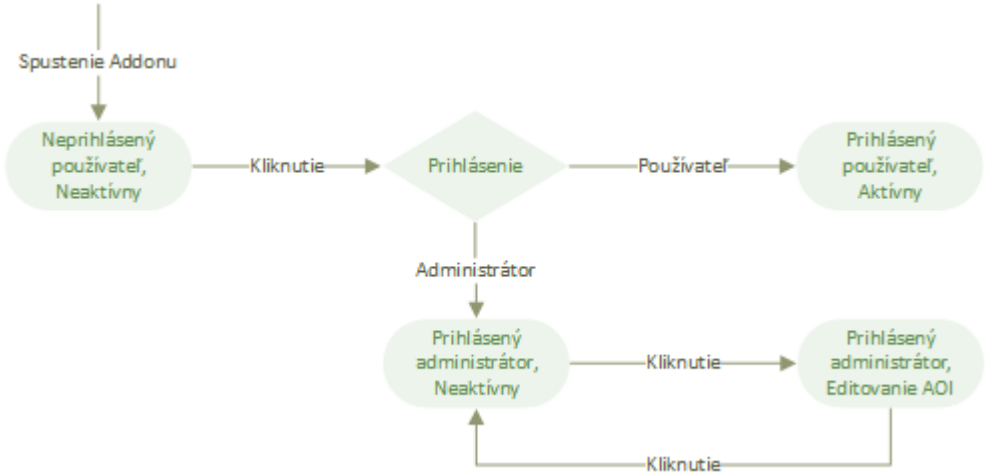
Po prihlásení si addon zapamätá používateľa.

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 podhodené 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

10 Globálne ciele projektu na letný semester

Organizácia a riadenie experimentov použiteľnosti

Hlavným cieľom nášho projektu je umožniť organizáciu experimentu, jeho vytvorenie, používateľsky jednoduché nastavenie, sledovanie a jeho centrálne riadenie. V zimnom semestri sa nám podarilo pokryť vytváranie experimentov a pridávanie používateľov k nim, v letnom semestri sa preto chceme sústrediť najmä na zdokonaľovanie toho, čo sa už podarilo. Cieľom je upraviť používateľské rozhranie webu, klientskej aplikácie ale aj rozšírenia do prehliadača tak, aby bolo intuitívnejšie, jasne definovať právomoci pre všetky používateľské roly, zabezpečiť bezpečné prihlasovanie a posielanie dát a hlavne plnú funkčnosť webového rozhrania pre administráciu experimentov a používateľov.

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 sme 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.

Cieľom v tejto časti je v letnom semestri najmä rozšíriť možnosti definovania oblastí záujmu, pridať možnosť sledovania elementov, ktoré sa na stránke dynamicky generujú a možnosť definovania viacerých elementov s rovnakým znakom naraz ako oblasť záujmu. Dôležitým cieľom je tiež zabezpečenie podpory viacerých prehliadačov a viacerých druhov zariadení pre sledovanie pohľadu.

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

Nadviazaním na prototyp zo zimného semestra by sme chceli dosiahnuť, aby náš produkt zvládal kompletné spracovanie dát, čo zahŕňa nielen ich ukladanie, ale aj porovnávanie definovaných oblastí so sledovanými oblasťami a na základe toho vytvárať automatické anotácie konkrétnych momentov sledovania a vizualizácie výsledkov sedenia priamo vo webovom rozhraní gaze admin.

Vyhodnocovanie a anotovanie nazbieraných dát

Úlohou v letnom semestri je vytvoriť webové rozhranie, pomocou ktorého bude možné zobraziť štatistické údaje pre jednotlivých používateľov počas sedenia, alebo celkové štatistiky súvisiace so sedením. Taktiež by sme chceli implementovať automatické anotovanie dát a vytvoriť rozhranie pre používateľské definovanie anotácií. Anotácie by mali byť viazané na oblasti záujmu. Ak sa následne používateľ určený čas pozerá do danej oblasti vytvorí sa k dátam anotácia.

11 Opis produktu po letom semestri

V tejto kapitole sú popísané komponenty a celková architektúra nášho systému. Popis komponentov obsahuje aj niektoré hlavné používateľské príbehy, ktorým sme sa venovali v letnom semestri.

Šprinty letného semestra nie sú popísané tak podrobne ako šprinty zimného semestra. Samotné používateľské príbehy sú spomenuté v tejto kapitole pri konkrétnych komponentoch, ktorých sa týkajú.

Bližšie informácie k samotným šprintom je možné vidieť v retrospektívach v dokumentácii riadenia v časti 6.3.

11.1 Architektúra

V tejto časti je popísaná architektúra nášho systému. V nasledovných častiach sú podrobnejšie opísané konkrétne komponenty, spolu s niektorými hlavnými používateľskými príbehmi.

Architektúra nášho prototypu je rozložená do troch základných vrstiev.

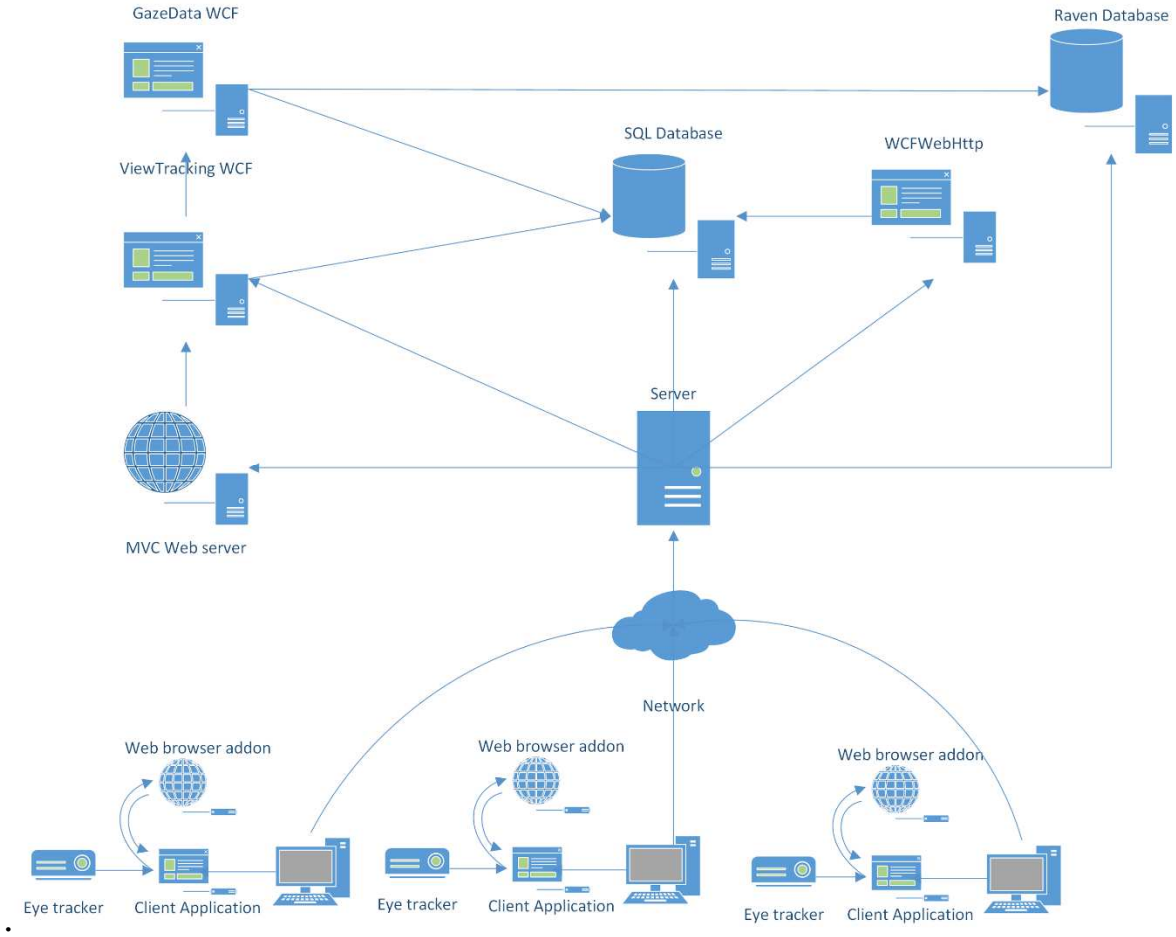
Na prvej vrstve sa nachádza desktopový klient, ktorý zabezpečuje komunikáciu so zariadením na sledovanie pohľadu. Jeho úlohou je zisťovať, kam sa používateľ pozerá a odosielať tieto dáta na server. Počas priebehu letného semestra sa nám podarilo upraviť klientskú aplikáciu tak, že podporuje dva rôzne typy eye-trackerov a to od firmy Tobii a od firmy The Eye Tribe. Pomocou tejto aplikácie sa používateľ autentifikuje voči serveru. Aplikácia je navrhnutá tak aby nebolo potrebné meniť žiadne nastavenia pri zmene eye-trackera.

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 v prehliadači pozerá. Túto skutočnosť komunikuje s desktopovým klientom, ktorý následne tieto dáta a mnohé ďalšie posiela na server. V princípe má rozšírenie za úlohu obohatiť dáta so senzora tak aby boli následne ľahko vyhodnocované. Tieto informácie sú HTML element na ktorý sa používateľ pozeral, záložka prehliadača (URL adresa otvorenej stránky) a taktiež normalizované súradnice kam sa používateľ pozeral v prehliadači. Počas letného semestra sme dostali požiadavku pre podporu viacerých prehliadačov a vytvorili sme teda rozšírenie aj pre Google Chrome a nielen pre FireFox. Toto rozšírenie je vytvorené na modulárnom princípe a teda veci ktoré sú spoločné pre rôzne typy prehliadačov sú uložené v jednom balíku, aby prípadná potreba vytvorenia nového rozšírenia bola možná čo najrýchlejšie a najjednoduchšie.

Tretia vrstva - samotný server - zabezpečuje zber dát získavaných z viacerých desktopových klientov pomocou REST architektúry. Dôležitou súčasťou tejto vrstvy je aj databáza, na ktorej sú rôzne dáta - od informácii o používateľoch až po sedenia a projekty navrhnuté výskumníkmi. Nakoľko množstvo senzorických dát je príliš veľké a MS SQL databáza by tento objem dát nezvládla museli sme na servery implementovať aj NoSQL dokumentovú databázu Raven DB. Tento druh databázy sme vybrali najme kvôli jej

vysokej priepustnosti a implementácii pomocou frameworku .Net a tým sme zabezpečili jednotnosť použitej technológie.

Na obrázku nižšie je vidieť diagram architektúry nášho riešenia



Obrázok 9 - Architektúra nášho tímového projektu

11.2 Server

Webové rozhranie GazeAdmin a klient, ktorý je nainštalovaný na desktope, využívajú webovú službu, ktorá obsahuje aplikačnú vrstvu nášho riešenia. Celá logika pre spracovávanie dopytov z gazeadmin alebo klienta je sústredená v službe Viewtracking.Wcf.

11.2.1 Aplikačná vrstva

Aplikačná vrstva obsahuje metódy pre komunikáciu s databázou. Aplikačná vrstva obsahuje repozitár, ktorý umožňuje volať metódy nezávisle od ich implementácii. To znamená, že môžu byť CRUD metódy na databázu použité viac krát rôznymi podprojektami, stačí pridať referenciu na projekt Viewtracking.Infrastructure, kde sa nachádza celá aplikačná logika. Triedy ktoré súvisia s databázou v aplikačnej vrstve sú dvojakého druhu:

1. DTO (Data transfer Object)
2. generované Entity frameworkom podľa databáze

Objekty s označením DTO sú posielané a zdielané metódami v Infrastructure projekte, alebo medzi inými projektmi cez referencie. Generované objekty slúžia len na ukladanie dát do databáze a pre potreby kontextu Entity frameworku.

11.2.2 WCF

V infrastructure projekte v časti Extensions sa nachadzajú metódy pre manažovanie dopytov na službu s názvom Viewtracking.Wcf, ktorá poskytuje dáta pre gazeadmin a desktopového klienta. Služba je naprogramovaná tak, aby komunikovala protokolom HTTPS. Každý dopyt na túto službu kontroluje voči databáze, či boli odoslané prihlasovacie údaje používateľa. O odosielanie používateľských prihlasovacích údajov sa v gazeadmine stará objekt HandleRequest, ktorý tiež odosiela dáta metódami POST a GET.

Wcf služba odosiela dáta vo formáte JSON. Pri publikovaní webovej služby je treba nastaviť Connecion string pre databázu, ktorého názov musí byť rovnaký, aký má v projekte Infrastructure objekt ModelContainer, ktorý súvisí s Entity Frameworkom.

11.2.3 API

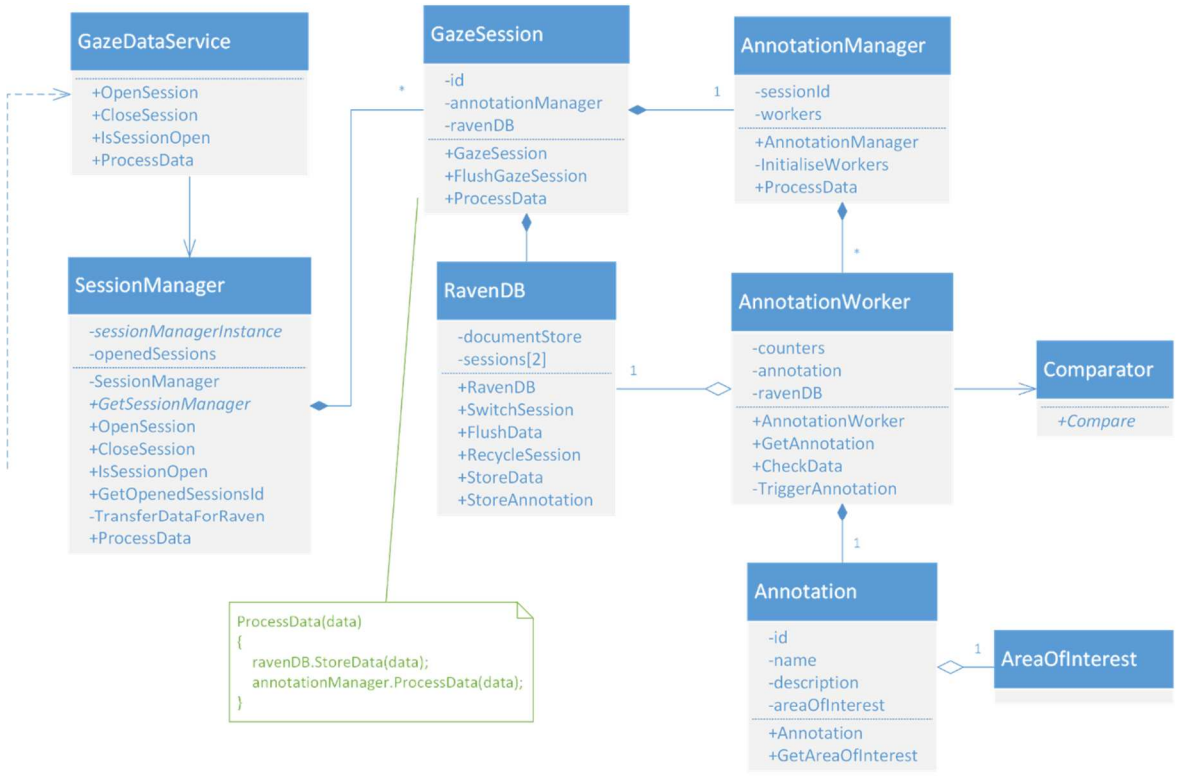
Pre aplikácie tretích strán bol vytvorený projekt , ktorý poskytuje metódy súvisiace so oblasťami záujmu, sedeniami, projektmi a pre sledovanie toho kam sa používateľ pri uvedenom sedení práve pozeral.

Táto webová služba kontroluje, či pri dopyte na jednu s definovaných metód bol odoslaný API kľúč, ktorý je kontrolovaný voči databáze. Každá aplikácia dostane priradený API kľúč v gazeadmin rozhraní. Služba API vracia dáta vo formáte JSON. Využíva databázové metódy, a preto je potrebné, aby mala vo web configu definovaný connection string na databázu a referenciu na projekt Infrastructure. Ak nebol pri dopyte na metódu odoslaný API kľúč, služba vráti naspäť html upozornenie o absencii API kľúča. Podobný scenár platí pri nesprávnom API kľúči. Pretože dopytov pri asynchrónnom volaní metód môže byť veľa, služba využíva cache pre ukladanie používaných API kľúčov, aby sa ušetril dopyt na databázu.

11.2.4 GazeDataService

Pre zber dát zo sledovania a automatické anotovanie sme potrebovali naše riešenie obohatiť o novú službu, ktorá by umožnila uchovávanie stavu, v ktorom budú uložené všetky potrebné informácie k ukladaniu a sparacovaniu dát.

Vytvorili sme singleton službu s názvom GazeDataService, ktorá má za úlohu manažovať sedenia a spracovanie dát týchto sedení. Služba komunikuje s inštanciou objektu SessionManager, ktorý na požiadanie otvára a zatvára sedenia a uchováva zoznam objektov GazeSession. Spracovanie dát je presmerované na konkrétne sedenie volaním ProcessData metódy inštancie GazeSession. Ak sedenie, pre ktoré sú dáta určené nie je otvorené, dáta sú zahodené.



Obrázok 10 - Architektúra objektov v GazeDataService

Inštancia GazeSession obsahuje inštanciu manažéra pre anotáciu a inštanciu pripojenia k Raven databáze. Spracovanie dát v rámci GazeSession vykonáva jednak uloženie dát prostredníctvom volania metódy StoreData na RavenDB objekte, ale aj porovnanie sledovaných oblastí s definovanými anotáciami.

Anotácie

Porovnávanie oblastí prebieha volaním metódy ProcessData na inštancií objektu AnnotationManager, ktorý pri vytvorení automaticky načíta anotácie patriace k zvolenému sedeniu a automaticky vytvorí inštancie objektu AnnotationWorker, jednu pre každú načítanú anotáciu.

AnnotationWorker porovnáva Xpath oblasti záujmu priradenej k anotácií a XPath oblasti, ktorú používateľ sledoval. Porovnávanie je implementované ako samostatný objekt Comparator, ktorý je možné v prípade potreby zameniť. V prípade že dôjde k dosiahnutiu podmienky potrebnej pre vytvorenie záznamu anotácie do databázy, AnnotationManager prostredníctvom referencie na RavenDB vytvorí anotáciu.

Momentálne poskytujeme vytváranie anotácií na základe časového intervalu, počas ktorého sa používateľ pozeral na danú oblasť. Riešenie je tolerantné aj na krátkodobé vychýlenie sa pohľadom mimo sledovanej oblasti.

Porovnávanie XPath bolo implementované pomocou porovnávania reťazcov s tým, že sa najskôr berie do úvahy či dané reťazce spĺňajú dĺžkové obmedzenia, ako napríklad či je XPath sledovanej oblasti rovnako dlhý, alebo dlhší ako XPath oblasti záujmu v anotácií, teda či sa môže jednať o rovnaký element, prípadne vnorený element. Následne je porovnávanie reťazcov. Okrem XPath sa samozrejme porovnáva aj URL adresa oboch elementov navzájom.

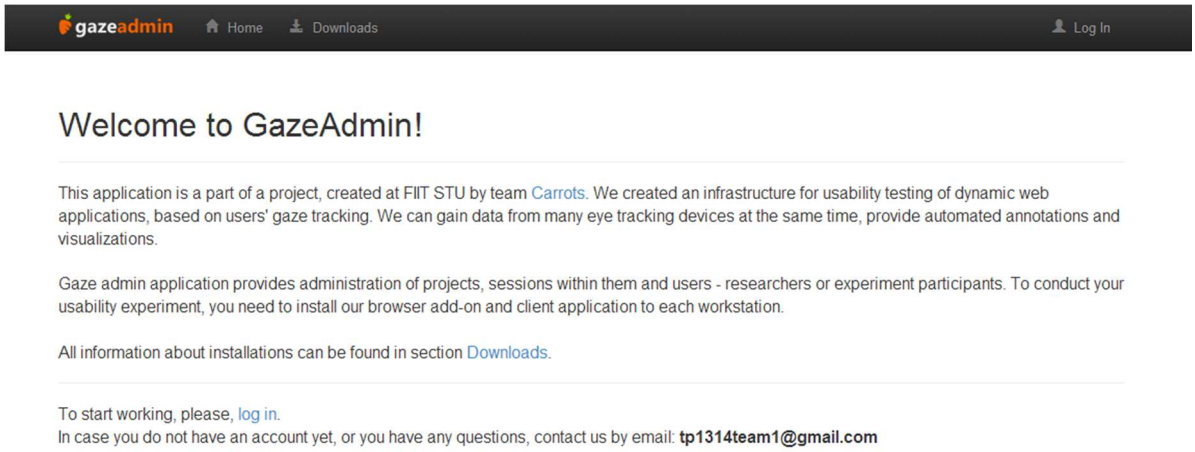
Nový spôsob zadávania oblastí záujmu v rozšírení však vyžadoval zmenu porovnávania. Pri vytvorené oblasti záujmu sa vygeneruje aj regulárny výraz, ktorý dokáže rozpoznávať viaceré elementy, ktoré sú priradené k oblasti záujmu.

Databáza

Ukladanie dát do Raven databázy zabezpečuje už spomínaný objekt RavenDB. Pri inicializácií inštancie je vytvorené spojenie na Raven databázu a dve databázové sedenia. Po zavolaní metódy StoreData sú dáta uložené do objektu databázového sedenia - operačnej pamäte a až po naplnení kvóty 1000 záznamov sú tieto záznamy reálne uložené do databázy. Aby nedochádzalo k zahlcovaniu operačnej pamäte, používame dve databázové sedenia systémom dvojitého zásobníku. Jedno databázové sedenie sa používa pre ukladanie dát a druhé sa zatiaľ vyprázdni a pripraví na znovupoužitie. Pri dosiahnutí kvóty sa databázové sedenia vymenia.

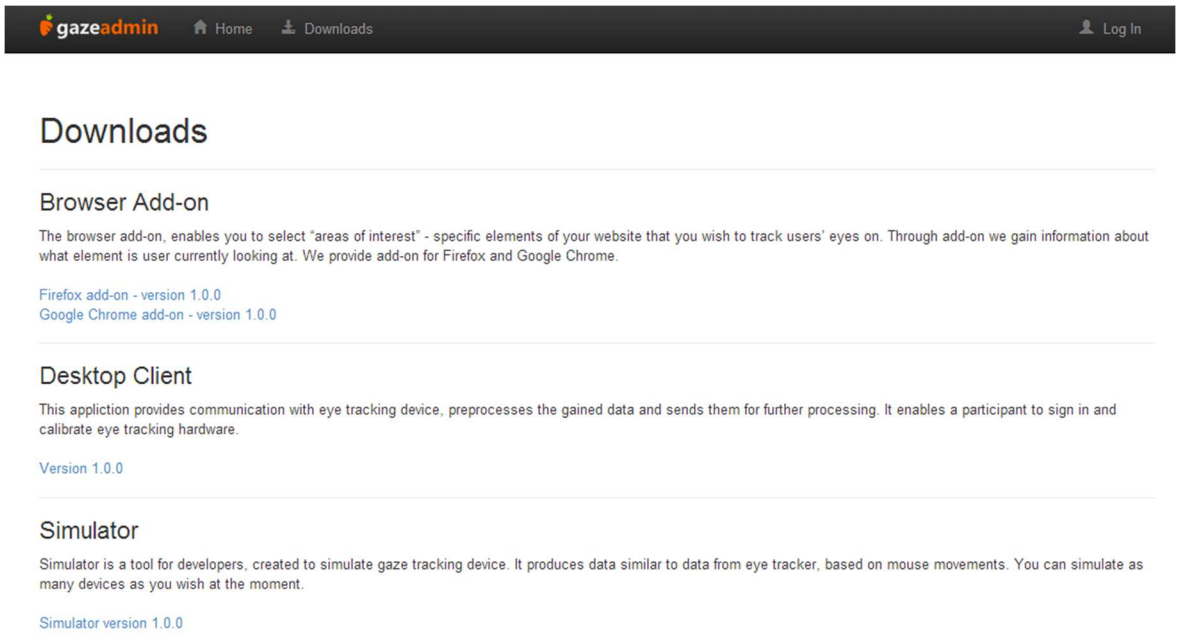
11.3 Webové rozhranie

Webová aplikácia gazeadmin, ako súčasť našej infraštruktúry, slúži najmä na administráciu experimentov a používateľov. Pomocou webového rozhrania si výskumník založí a definuje svoj projekt a konkrétne sedenie - UX experiment, pridáva k nemu používateľov a definuje oblasti záujmu vo svojej testovanej webovej aplikácií. Okrem toho aplikácia obsahuje aj rozhranie na definovanie automatických anotácií a rozhranie pre štatistiky, kde si výskumník po skončení experimentu môže prezerať a filtrovať nazbierané dáta.



Obrázok 11 - Úvodná stránka aplikácie gazeadmin

Naša webová aplikácia tiež obsahuje odkazy na stiahnutie všetkých inštalovateľných častí (Desktop klient, rozšírenia do prehliadačov a simulátor) v sekcii Downloads, ktorá je dostupná aj bez prihlásenia používateľa.



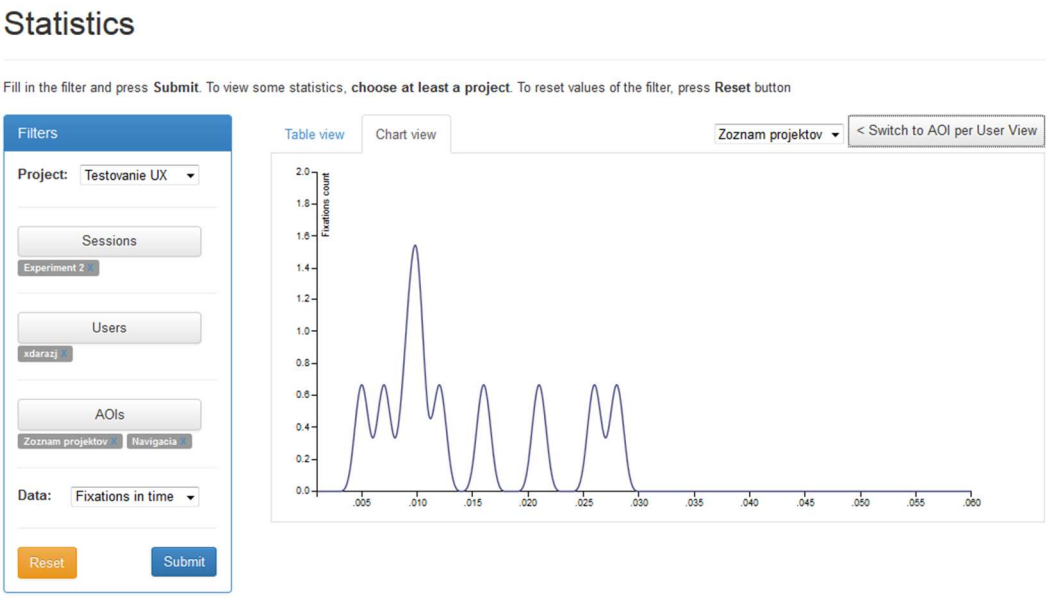
Obrázok 12 - Stránka Downloads so stiahnuteľnými inštaláciami

11.3.1 Základný požívateľský príbeh (user story)

Webová aplikácia gazeadmin sprevádza používateľa - výskumníka celým procesom usability testovania dynamickej webovej aplikácie. V tejto časti opisujeme základný používateľský príbeh, ktorý sme sa pri implementácii snažili pokryť:

Používateľ - výskumník chce otestovať svoju aplikáciu.

1. Používateľ nemá konto, preto kontaktuje administrátora, kvôli registrácii (kontakt je uvedený na úvodnej stránke). Administrátor používateľa zaregistruje a priradí mu rolu.
2. Registrovaný používateľ - výskumník sa prihlási do aplikácie gazeadmin. Podporujeme aj prihlasovanie pomocou mena a hesla používateľa z akademického informačného systému STU.
3. Vytvorí projekt pre svoju aplikáciu.
4. Pridá k projektu používateľov, ktorí budú môcť tiež upravovať alebo prezerať dáta z experimentov.
5. V rámci projektu vytvorí sedenia a pridá k nim participantov. Sedenie je časovo ohraničené a zodpovedá jednému experimentu/testovaniu s viacerými používateľmi. Participantov môže pridávať po jednom alebo po skupinách.
6. Pre sedenie (experiment) definuje oblasti záujmu, tzv. Areas of interest (AOI), ktoré chce v rámci svojej aplikácie sledovať. Definovanie AOI prebieha v spolupráci s rozšírením do prehliadača, ktoré musí byť vopred nainštalované.
7. Pred začatím sedenia používateľ definuje svoje automatické anotácie.
8. Po ukončení experimentu si výskumník chce pozrieť štatistiky, ktoré mu pomôžu jeho experiment vyhodnotiť. Je možné aj exportovať diagramy do formátu .png. Podrobnejší popis rozhrania štatistík sa nachádza v používateľskej príručke k webovej aplikácii gazeadmin.



Obrázok 13 - Rozhranie pre štatistiky

11.3.2 Administrácia používateľov

Pre efektívnu administráciu používateľov sme vytvorili v rámci aplikácie niekoľko používateľských rolí:

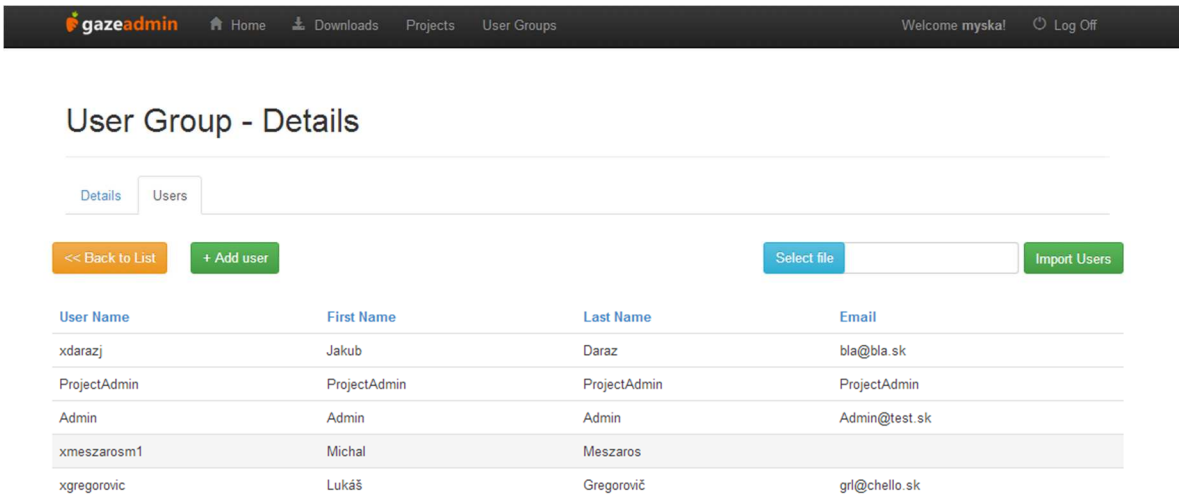
Admin - globálny administrátor, vidí všetky projekty, ktoré boli vytvorené a môže ich upravovať. Môže tiež pridávať nových používateľov s ľubovoľnou rolou.

Project admin - administrátor konkrétneho projektu, používateľ, ktorý vidí len svoje projekty a projekty, na ktoré je pridelený. Môže ich upravovať a pridávať k nim ďalších používateľov.

Researcher - výskumník, používateľ pridaný na projekte, ktorý vidí všetky dáta v ňom, ale nemôže ich upravovať. Má prístup k detailom projektov, na ktoré je pridaný, aj sedení v rámci týchto projektov. Môže tiež vidieť štatistiky k týmto projektom.

User - participant UX experimentu. Tento používateľ si nemôže prezerať projekty ani sedenia, má len vytvorené konto a prostredníctvom desktopového klienta sa na začiatku experimentu prihlási, aby bolo možné zbierať jeho dáta.

Pridávať nových používateľov s rolou Admin, Project admin alebo Researcher môže len Admin. Pridať používateľov s rolou User môže aj ľubovoľný Project admin tak, že pridá skupinu používateľov pomocou importu z .csv súboru ako skupinu používateľov.



Obrázok 14 - pridávanie skupín používateľov s možnosťou importu z .csv súboru

11.3.3 Implementácia a použité technológie

Celá webová aplikácia je implementovaná pomocou technológie ASP.NET MVC s využitím JavaScriptu a knižnice jQuery. Server je postavený na operačnom systéme Windows Server 2012. Komunikácia so serverom je prostredníctvom post a get requestov na REST službu ktorú server poskytuje navonok.

Využívame databázové riešenia MsSQL a NoSQL databázu RavenDB.

S NoSQL databázou RavenDB gazeadmin komunikuje cez priamu session, ktorej inštancia sa vytvára pre konkrétny model štatistík. Týmto spôsobom sme zabezpečili čo najrýchlejšiu komunikáciu pomocou zníženia počtu komunikačných uzlov. Pri vyhodnocovaní štatistík z dôvodu netriviálne veľkých dopytov do RavenDB, využívame predspracovanie dát na strane databázy pomocou indexov, ktoré podporuje databáza Raven.

Aplikácia umožňuje prihlásenie používateľom aj s menom a heslom z akademického informačného systému STU, pomocou knižnice LDAP. Na zrýchlenie pridávania viacerých používateľov súčasne, je implementovaný import z .csv súboru. Pomocou html5 a canvasu funguje export diagramov štatistík do formátu .png.

Štýlovanie stránky sme zabezpečili najmä pomocou front-end frameworku Boostrap.

11.4 Rozšírenie pre webový prehliadač

V našom projekte je dôležitou súčasťou aj rozšírenie pre webový prehliadač. Úlohou rozšírenia je zabezpečiť dve rôzne funkcionality, ktoré sú potrebné pre uskutočnenie celého používateľského príbehu.

11.4.1 Obohacovanie dát zo sledovania

Prvou jednoduchou funkcionalitou ktorú rozšírenie poskytuje, je obohacovanie dát zo sledovania. Dáta zo sledovania sa zbierajú v klientskej aplikácií priamo zo zariadenia na sledovanie pohľadu. Rozšírenie následne na základe požiadavky na klientskú aplikáciu získa dáta, ktoré sú vo forme viacerých záznamov. Pre každý záznam osobitne zistí, na aký element stránky sa používateľ pozeral využitím súradníc pohľadu daného záznamu. Po spracovaní všetkých záznamov sa obohatené dáta odošlú naspäť klientskej aplikácií a rozšírenie si vyžiada ďalšie dáta.

11.4.2 Manažment oblastí záujmu

Druhou funkcionalitou je poskytnutie podpory pre manažovanie oblasti záujmu, ktoré sa neskôr využívajú pre sledovanie určitých častí webovej aplikácie.

Aktivácia / deaktivácia rozšírenia

Po inicializovaní rozšírenia v prehliadači je nutná počiatočná interakcia pomocou webového rozhrania gazeadmin. Interakcia spočíva v zvolení sedenia, pre ktoré má rozšírenie vytvárať nové oblasti. V gazeadmin je potrebné zvoliť projekt a sedenie. V časti editácie sedenia, v záložke Areas of interest sa nachádza tlačidlo pre aktivovanie / deaktivovanie rozšírenia pre dané sedenie.

Po aktivácií cez gazeadmin je rozšírenie pripravené na vytváranie oblastí pre zvolené sedenie. Jeho deaktivácia a opätovná aktivácie je už možná aj pomocou Ikonky rozšírenia (mrkva). V prípade, že je potrebné aktivovať rozšírenie pre iné sedenie, opäť sa vyžaduje interakcia pomocou gazeadmin.

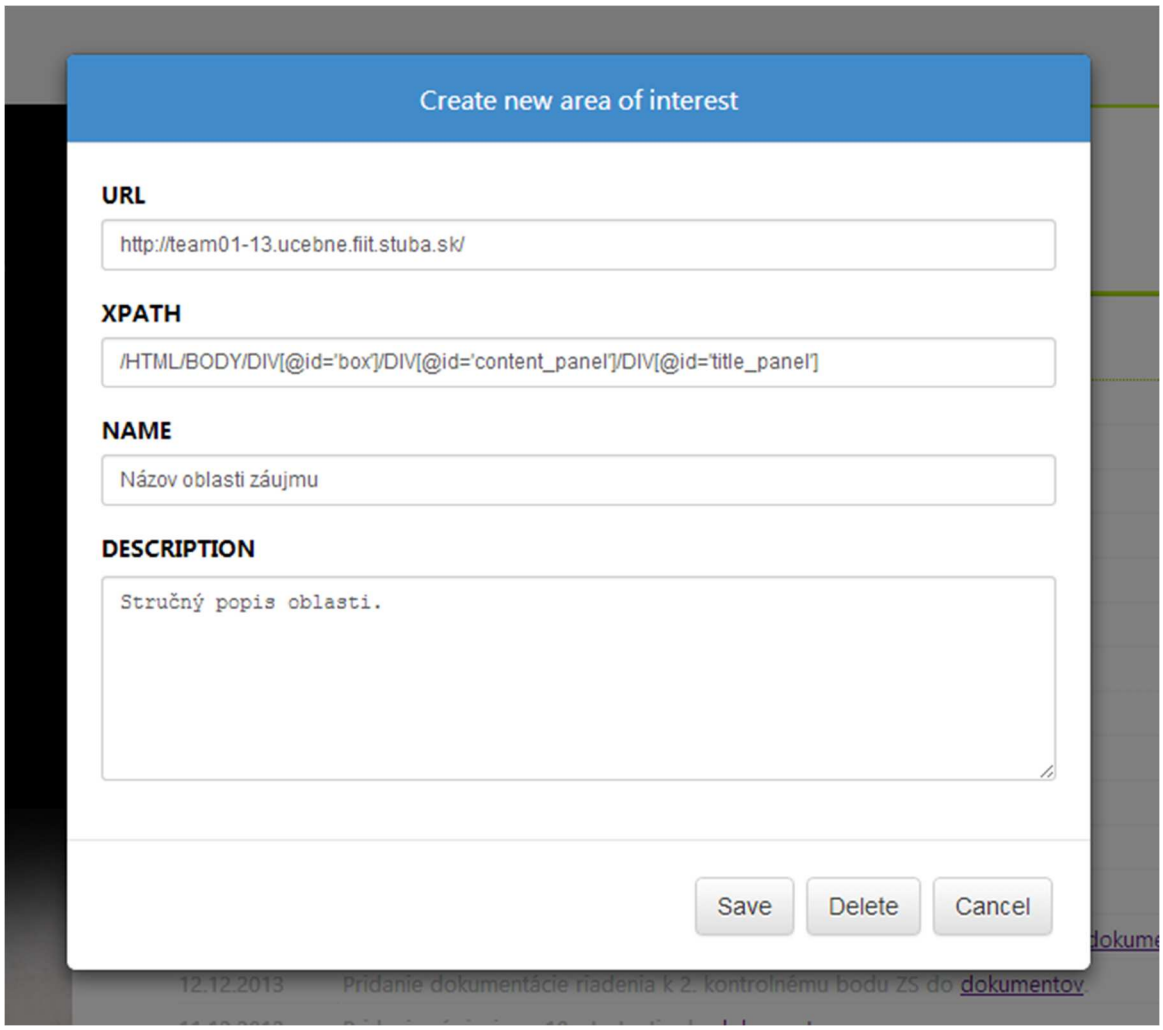
Stav rozšírenia je možné identifikovať na základe ikonky rozšírenia. Logo v odtieňoch šedej označuje stav, kedy je rozšírenie neaktívne a farebná verzia ikonky zase stav, kedy je rozšírenie aktívne.

Vytváranie oblastí záujmu

Ak je rozšírenie v aktívnom stave, prechádzaním myši nad webovou aplikáciou sa vyznačujú jednotlivé HTML elementy oranžovým orámovaním a tooltipom, v ktorom je XPath identifikátor pre vyznačený element.

Kliknutím na ktorýkoľvek element sa otvorí dialóg pre vytvorenie oblasti pre daný element. Akákoľvek onclick akcia, ktorú majú elementy k sebe priradenú vo webovej aplikácií je pri aktivovanom rozšírení ignorovaná. Pre navigovanie vo webovej aplikácií je nutné rozšírenie dočasne deaktivovať, napríklad kliknutím na jeho ikonku v prehliadači a neskôr ho následne aktivovať rovnakým spôsobom.

Dialóg obsahuje niekoľko položiek, ktoré definujú oblasť záujmu. Prvé dve preddefinované položky sú URL adresa stránky webovej aplikácie, na ktorej bol zvolený element pre oblasť záujmu a XPath identifikátor tohoto elementu. Ďalšie dve položky sú názov oblasti a jej popis, ktoré sú odporúčané pre jednoduchšiu orientáciu medzi viacerými oblasťami.



Obrázok 15 - Základný dialóg pre definovanie oblasti záujmu

Poskytované sú 3 akcie dialógu. Uloženie oblasti, zmazanie oblasti alebo zrušenie dialógu bez vykonania zmien. Uloženie alebo zmazanie oblasti zavolá patričnú webovú službu na serveri. Po vykonaní operácie znovu načíta všetky definované oblasti záujmu pre aktívne sedenie a aktualizuje zvýraznenie elementov ktoré majú definované oblasti záujmu. Výsledok operácie je zobrazený v notifikácii.

Implementácia

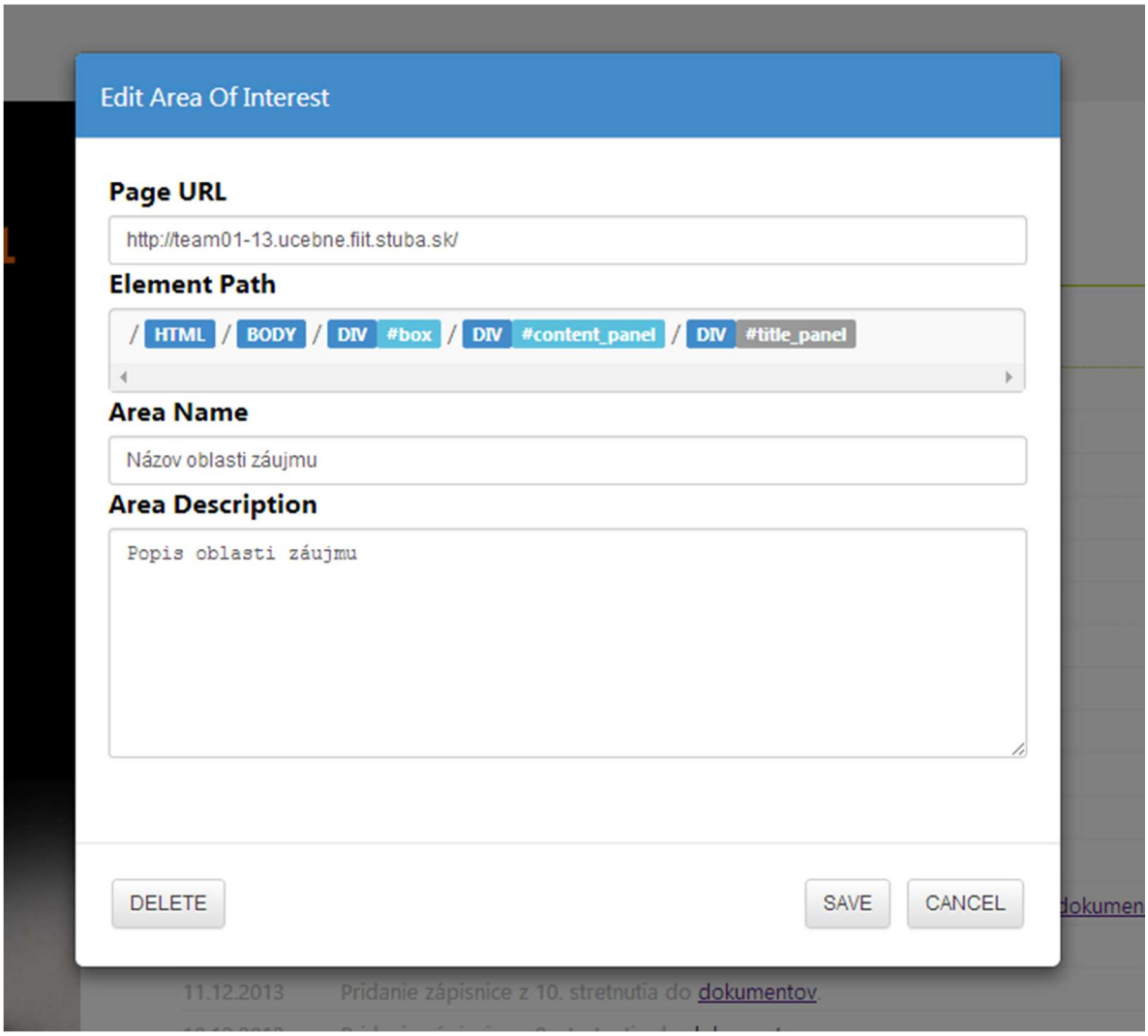
V novej verzii rozšírenia(Chrome verzia) je vyššie spomínaná funkcionalita oddelená a poskytované osobitne v dvoch verziách rozšírenia: klientská časť a administrátorská časť.

Pri tvorbe Chrome rozšírenia sme sa zameriavali najmä na oddelenie nezávislej funkcionality rozšírenia od špecifických funkcií potrebných pre samotné rozšírenie. Vznikla tak oddelená “knižnica”, ktorú je možné viacnásobne použiť. Momentálne však rozšírenie pre Firefox ostáva v pôvodnej verzií a obsahuje funkcionalitu, ktorá je implementovaná pomocou špecifických firefox api funkcií. Refaktoring Firefox rozšírenia pre nedostatok času nie je možné dokončiť do termínu odovzdávania prototypu.

Nová verzia zadávania oblastí záujmu

Na základe požiadaviek na rozšírené možnosti definovania oblastí záujmu, vznikol prototyp nového spôsobu definovania elementov, ktoré budú zodpovedať danej oblasti záujmu. Konkrétnou požiadavkou bolo napríklad definovanie oblasti pre viacero elementov súčasne.

Princípom nového prístupu je zavedenie interaktívnej grafickej reprezentácie XPath identifikátora elementov pre oblasť záujmu, ktorá sa v dialógu oblasti zobrazí namiesto bežného reťazca XPath identifikátoru.



Obrázok 16 - Dialóg s novým spôsobom definovania elementov pre oblasť záujmu

Grafická reprezentácia XPath obsahuje niekoľko prvkov. Sýto modrou farbou sú zobrazované prvky reprezentujúce jednotlivé HTML elementy v hierarchii dokumentu zoradené od najvyššej úrovne až k zvolenému elementu. Svetlomodré a sivé prvky predstavujú atribúty pre dané HTML elementy.

Nemenným základom pre vygenerovanie XPath identifikátora je postupnosť elementov. Atribúty je možné kliknutím zahrnúť, respektíve vyradiť z výsledného XPath identifikátora.

Prototyp pracuje s atribútmi id, class, ale aj poradovým číslom elementu. Deaktivovaním napríklad atribútu poradového čísla je možné oblasť záujmu definovať nielen pre konkrétny element, ale aj pre viaceré elementy, ktoré vyhovujú XPath identifikátoru (napr. všetky riadky v stĺpci tabuľky). Podobne je možné pracovať s atribútmi class, kedy je na základe webovej aplikácie využiť napríklad definovanie oblasti pre párne, respektíve nepárne položky zoznamu a podobne.

Tabuľka 1 - Príklady definovania elementov patriacim k oblasti záujmu

/ TABLE 1. / TBODY 1. / TR 5. / TD 1.	1. bunka v 5. riadku tabuľky
/ TABLE 1. / TBODY 1. / TR 5. / TD 1.	všetky bunky v 1. riadku
/ TABLE 1. / TBODY 1. / TR 5. / TD 1.	všetky bunky v 1. stĺpci
/ TABLE 1. / TBODY 1. / TR 5. / TD 1.	všetky bunky tabuľky

Generovanie XPath pomocou novej grafickej reprezentácie aplikuje niekoľko pravidiel preferencie atribútov. Najvyššiu prioritu má ID elementu. Ak element nemá definovaný parameter ID, namiesto neho sa zobrazí parameter pozície elementu, ktorý daný element jedinečne identifikuje podobne ako parameter ID. Najnižšiu prioritu majú class atribúty a sú brané do úvahy iba ak ID alebo poradové číslo elementu sú v danom elemente deaktivované.

Pri implementácii prototypu bolo nutné objekt areaOfInterest rozšíriť o ďalšie atribúty, najmä o kompletný stav cesty, ktorý je potrebná pre neskoršie zrekonštruovanie cesty v dialógu v posledne definovanom stave. Druhým atribútom, ktorý je dôležitý pre vytváranie anotácií je regulárny výraz pre rozpoznávanie zvolených elementov prislúchajúcich definovanej oblasti záujmu.

11.5 Klientská aplikácia

Pre samotné získavanie dát je najdôležitejšou súčasťou nášho riešenia klientská aplikácia. Táto aplikácia priamo komunikuje so zariadením pre sledovanie pohľadu a získava z neho dáta.

11.5.1 Komunikácia so zariadením

Najzákladnejším používateľským príbehom je pripojenie na zariadenie pre sledovanie pohľadu a získavanie a odosielanie dát z neho. Pre komunikáciu so zariadením sme najprv jednoducho používali API poskytnuté výrobcom daného zariadenia. Po získaní zariadenia od ďalšieho výrobcu a vzniku požiadavky na podporu aj tohto zariadenia už ale nebolo vhodné priamo použiť ďalšie API. Vytvorili sme teda vlastnú knižnicu, ktorá implementuje knižnice výrobcov týchto zariadení a umožňuje nám jednoducho a rovnakým spôsobom používať rôzne zariadenia, vrátane simulátora, ktorý sme vytvorili my.

11.5.2 Podpora viacerých zariadení

Podpora viacerých zariadení bol jedným z ústredných používateľských príbehov tohto semestra. Ako používateľ nášho softvéru totiž nechcem riešiť to, ktoré zariadenie mám práve pripojené a chcem ho jednoducho začať používať. Preto sme vytvorili našu knižnicu, spomínanú v predchádzajúcej časti. Táto knižnica taktiež umožňuje vývojárom jednotný prístup k viacerým rôznym zariadeniam. Bližší pohľad na samotnú knižnicu sa nachádza ďalej v dokumente v časti 11.5.6 Implementácia a použité technológie.

11.5.3 Vyhladzovanie dát

Zariadenie pre sledovanie pohľadu nám dáva k dispozícii normalizované súradnice, ktoré nám hovoria, kam na obrazovku sa používateľ pozerá. Teda ak sa používateľ pozerá presne do stredu obrazovky, súradnice by mali byť 0,5 na x-ovej osi a taktiež 0,5 aj na y-ovej osi. V skutočnosti tomu ale tak nie je. Zariadenie pre sledovanie pohľadu má určitú presnosť. Súradnice, ktoré nám dáva zariadenie teda "skáču" okolo skutočného bodu, na ktorý sa používateľ pozerá. Ak teda chceme zo súradníc zisťovať napr. konkrétny element web stránky, môže nám byť vrátený úplne iný element, než na ktorý sa používateľ pozeral.

Rozhodli sme sa teda implementovať aj vyhladzovanie súradníc prichádzajúcich zo zariadenia. K dispozícii sú momentálne dve stratégie vyhladzovania súradníc, obe založené na priemerovaní:

- prvá jednoducho priemeruje každých 5 súradníc, ktoré zo zariadenia prídu, čím vlastne klesne vzorkovacia frekvencia zariadenia 5-násobne
- druhá priemeruje vždy posledných 5 súradníc a teda neprichádza k zníženiu frekvencie

Odporúčame používať druhú stratégiu, ktorej hlavná výhoda je hlavne zachovanie frekvencie zariadenia.

11.5.4 Komunikácia s rozšírením pre prehliadač

Ďalším dôležitým používateľským príbehom je obohacovanie dát, prichádzajúcich zo zariadenia pre sledovanie pohľadu. Ako výskumník totiž nechcem úplne ručne anotovať a vyhodnocovať dáta z experimentu. Preto dáta obohacujeme pomocou rozšírenia pre prehliadač, čo je opísané vyššie. Pre komunikáciu so samotným rozšírením slúži HTTP server zabudovaný v klientskej aplikácii. Prednastavený je pre komunikáciu port 62582, ktorý je možné zmeniť po spustení aplikácie (viď príručku). Bližší pohľad na komunikáciu je v časti 11.5.6 Implementácia a použité technológie.

11.5.5 Komunikácia so serverovou aplikáciou

Po získaní dát zo zariadenia pre sledovanie pohľadu a ich obohatenia sa dáta odosielajú na server. Ako výskumník totiž nechcem tieto dáta zbierať sám, čo by zabralo zbytočne veľa času. Toto je jeden z hlavných používateľských príbehov celého nášho projektu. Viac informácii o samotnom serveri je možné nájsť v časti 11.2 Server.

Posielanie dát na server je založené na architektúre REST. Klientská aplikácia posiela v daných časových intervaloch serverovej aplikácii získané dáta pomocou REST-ového volania.

11.5.6 Implementácia a použité technológie

Klientská aplikácia je implementovaná ako Win Forms aplikácia v jazyku C#. Aplikácia sa skladá z viacerých hlavných častí:

- používateľské rozhranie
- logika spracovania dát
- knižnica pre zariadenia na spracovanie pohľadu
- HTTP server - komunikácia s rozšírením pre prehliadač

Používateľské rozhranie

Rozhranie je bližšie popísané v prílohe tohto dokumentu, konkrétne v príručke pre klientskú aplikáciu a preto sa mu nebudeme venovať podrobne. Opíšeme stručne len jeho hlavnú funkcionalitu

- prihlásenie používateľa - potrebné pred použitím aplikácie pre ukladanie dát do sedení
- výber zariadenia pre sledovanie pohľadu
- pripojenie na vybrané zariadenie
- kalibrácia zariadenia
- výber sedenia - používateľ si môže vybrať, ku ktorému sedeniu (experimentu) sa majú priraďovať získavané dáta
- zobrazenie stavu aplikácie - môžeme vidieť informácie o tom, či zariadenie vidí oči, prípadne či je pripojené rozšírenie pre prehliadač
- zobrazenie prichádzajúcich súradníc - otvorí sa okno na celú obrazovku, ktoré graficky zobrazuje prichádzajúce súradnice - červený kurzor na mieste, kam sa používateľ pozerá
- nastavenia aplikácie

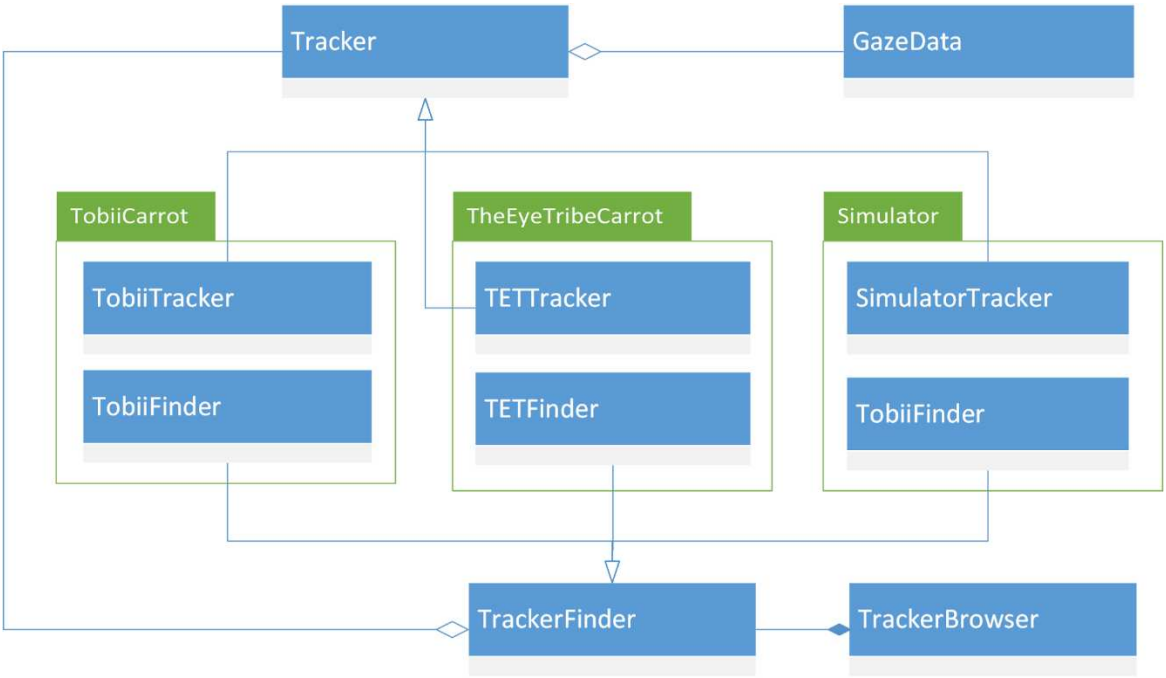
Logika spracovania dát

Dáta zo zariadenia sú spracovávané nasledovným spôsobom:

- dáta sú prijaté zo zariadenia pre sledovanie pohľadu a uložené do frontu
- dáta sú postupne vyberané z frontu a posielané rozšíreniu pre prehliadač pre obohacovanie
- po prijatí obohatených dát sú uložené do druhého frontu pre server
- v časových intervaloch sa dáta z druhého frontu posielajú na server

Knižnica pre zariadenia na spracovanie pohľadu

Vytvorili sme vlastnú knižnicu na komunikáciu so zariadeniami na spracovanie pohľadu. Na nasledovnom zjednodušenom UML diagrame tried môžeme vidieť jej približnú štruktúru.



Obrázok 17 - hlavné triedy z našej knižnice

Vývojári pri použití knižnice pracujú hlavne s triedami Tracker, TrackerBrowser a GazeData. Pomocou triedy TrackerBrowser je možné nájsť Trackeri (zariadenia) a pripojiť sa na ne. Potom vývojár pracuje s triedou Tracker a nemusí ho zaujímať, o aké konkrétne zariadenie v skutočnosti ide, či už skutočné, alebo náš simulátor. Ďalšou dôležitou triedou je trieda GazeData, ktorá obsahuje samotné dáta, ktoré zariadenie vytvára. Trieda má nasledovné atribúty:

- **public long timeStamp** - čas, kedy boli dáta vytvorené, vo formáte DateTime.Now.Ticks
- **public PointTwoD leftGazePoint2D** - 2d súradnice bodu, na ktorý sa pozerá ľavé oko používateľa
- **public PointTwoD rightGazePoint2D** - 2d súradnice bodu, na ktorý sa pozerá pravé oko používateľa
- **public PointTwoD leftGazePoint2Dsmooth** - vyhľadené súradnice pre ľavé oko
- **public PointTwoD rightGazePoint2Dsmooth** - vyhľadené súradnice pre pravé oko
- **public PointThreeD leftGazePoint3D** - 3D súradnice pohľadu pre ľavé oko, v prípade TheEyeTribe je Z súradnica vždy 0, keďže túto funkcionalitu nepodporuje

- **public PointThreeD rightGazePoint3**D - 3D súradnice pohľadu pre pravé oko, v prípade TheEyeTribe je Z súradnica vždy 0, keďže túto funkcionalitu nepodporuje

- **public int leftValidity** - hodnota v intervale 0-4, podľa toho, ako dobre je vidieť ľavé oko - 0 je najlepšie, pri 4 oko nevidí

- **public int rightValidity** - hodnota v intervale 0-4, podľa toho, ako dobre je vidieť pravé oko - 0 je najlepšie, pri 4 oko nevidí

- **public PointThreeD leftEyePosition3D** - 3D súradnice ľavého oka v priestore

- **public PointThreeD rightEyePosition3D** - 3D súradnice pravého oka v priestore

- **public PointThreeD leftEyePosition3DRelative** - 3D súradnice ľavého oka v priestore, relatívne k bounding box, ktorú počíta Tobii tracker. Bližšie informácie v Tobii dokumentácií.

- **public PointThreeD rightEyePosition3DRelative** - 3D súradnice pravého oka v priestore, relatívne k bounding box, ktorú počíta Tobii tracker. Bližšie informácie v Tobii dokumentácií.

- **public double leftPupilSize** - veľkosť zreničky ľavého oka

- **public double rightPupilSize** - veľkosť zreničky pravého oka

HTTP server

HTTP server sa používa pre komunikáciu s rozšíreniami pre prehliadač. Po prijatí HTTP GET požiadavky server odošle rozšíreniu ako JSON triedu DataForAddon, v ktorej sa posielajú súradnice pohľadu používateľa, na základe ktorých rozšírenie doplní do triedu XPath elementu, na ktorý sa používateľ pozeral a URL danej stránky.

11.6 Simulátor

Môže sa stať, že niektorí používatelia nášho systému nevlastnia zariadenie pre sledovanie pohľadu. Z toho dôvodu sme tiež vytvorili aplikáciu, ktorá dokáže simulovať pohľad používateľa zo vstupného zariadenia, ktoré vlastní každý - myš.

11.6.1 Výber simulátora pre “sledovanie pohľadu”

Dôležitým používateľským príbehom je zvolenie výber simulátora pre “sledovanie pohľadu”, nakoľko používateľ nemusí vlastniť zariadenie určené na sledovanie pohľadu. Po spustení aplikácie sa zobrazí používateľské rozhranie a súčasne sa spúšťa broadcast rozosielanie informácií o simulátore. Používateľské rozhranie je bližšie popísané v prílohe dokumentu, konkrétne v príručke pre simulátor. Odosielané informácie slúžia na identifikáciu simulátora v klientskej aplikácií. Výber simulátora pre “sledovanie pohľadu”, je možný cez klientku aplikáciu po prijatí správ s informáciami o simulátore. To, či bol

simulátor vybraný je možné overiť tlačidlom “Test connection”. Celá komunikácia prebieha na porte, ktorý si je tiež možné zvoliť prostredníctvom simulátora.

11.6.2 Simulovanie pohľadu

Po výbere simulátora pre “sledovanie pohľadu” ostáva používateľovi simulovať pohľad, čo je tiež dôležitý používateľský príbeh. Pre simulovanie pohľadu je nutné mať zvolenú myš ako zdroj dát *GazePoint2D*. Ak je simulátor zvolený ako zariadenia pre sledovanie pohľadu, stačí spustiť “sledovanie pohľadu” pomocou tlačidla “Start tracking”. Aplikácia sa zminimalizuje medzi schované ikony štart panelu a začne odosielať dáta klientskej aplikácií. Zastavenie simulácie je možné pomocou tlačidla “Stop tracking”.

Štruktúra dát odosielaných simulátorom

Dáta simulácie, ktoré odosiela simulátor majú podobnú štruktúru ako dáta, ktoré odosiela zariadenie od spoločnosti Tobii. Dáta sú však prefiltrované od zbytočných údajov, ktoré nemali význam pre účely simulácie a zberu dát simulovaného pohľadu. Dáta, ktoré simulátor odosiela je možné vidieť a manuálne nastavovať v používateľskom rozhraní simulátora.

12 Čo sme nestihli

V backlogu nášho podporného systému TFS nám ostalo niekoľko používateľských príbehov, ktorým sme sa nevenovali v žiadnom šprinte. Ide hlavne o príbehy s malou prioritou, teda príbehy, ktoré sme nepovažovali za podstatné.

Import exportovaných dát z Tobii Studio

Cieľom bolo umožniť importovanie údajov zo sledovania, ktoré je možné exportovať v aplikácií Tobii Studio. Výmenným súborom je v tomto prípade .xls súbor.

Filtrovanie projektov

Zobrazovanie projektov je momentálne implementované ako zobrazenie tabuľky, kde jednotlivé riadky reprezentujú napr. projekty. Tabuľku je možné usporiadať podľa stĺpcov (Id, Name, ...) a tak aj docieliť pomerne jednoduché vyhľadanie konkrétneho projektu. Pri väčšom počte položiek sa tento prístup môže stať menej praktickým a preto sme identifikovali ako užitočnú funkcionalitu možnosť záznamy filtrovať podľa zvolených parametrov (Id, Name). Filtrovanie sa neviaže nutne len na zoznam projektov.

Úprava vyhľadávania používateľov

Pri pridávaní používateľov k sedeniu, projektu, alebo do skupiny je vhodné upraviť vyhľadávanie a manipuláciu používateľov tak, aby bola práca efektívnejšia. Príkladom je mazanie viacerých používateľov naraz, alebo presmerovanie po ukončení pridávania používateľov na správnu záložku (Users) v rámci detailov.

Ukladanie kalibrácie

Pre možnosť overenie presnosti zaznamenaných dát zo sedenia sme diskutovali o potrebe uloženia kalibrácie zariadenia pre daného používateľa.

Správa zariadení

Výskumník, ktorý vykonáva test môže sledovať, či test prebieha v poriadku - dáta zo zariadení chodia v pravidelných očakávaných intervaloch, kvalita sledovania je na dobrej úrovni a podobne, či zariadenia správne fungujú a sú správne kalibrované. Táto funkcionalita umožní výskumníkovi identifikovať niektoré problémy pri testovaní a včas na ne reagovať.

Export

Získané dáta zo sledovania, ale aj agregované dáta je vhodné niekedy aj exportovať a využiť ich v inej aplikácií, alebo pri dokumentovaní výsledkov sedenia.

Zaznamenávanie videa zo sedenia

Dáta zo sledovania by bolo vhodné zlúčiť s videozáznamom z obrazovky používateľa, čo by umožnilo detailnejšie vyhodnocovanie niektorých špecifických situácií pri hromadnom sledovaní, napríklad aktivita používateľa pred, počas a po vytvorení automatickej anotácie, alebo celkového správania pri sledovaní malého počtu používateľov.

Monitorovanie vyťaženia serveru

Kontrola stavu serveru môže včas identifikovať vznikajúci problém. Počas sedenia a najmä pri sledovaní viacerých používateľov je vhodné sledovať vyťaženie serveru. Sledované parametre ako vyťaženie procesoru, operačnej pamäte, voľného miesta na disku alebo vyťaženie siete sa zobrazia v rozhraní gazeadmin.

13 Čo sme sa naučili

Počas práce na tímovom projekte sme získali znalosti softvérového aj hardvérového charakteru. Síce sme mali možnosť pracovať v malých tímoch, práca na projekte takéhoto rozmeru nám umožnilo získať nový pohľad na tímovú spoluprácu.

V zimnom semestri sme sa naučili pracovať s verziovacím nástrojom TFS a počas letného semestra sme našu prácu s TFS neustále zlepšovali.

Počas práce na projekte sme mali možnosť získať nové poznatky súvisiace s rôznymi technológiami. Osvojili sme si tvorbu rozšírení do prehliadačov Mozilla Firefox a Google Chrome, tvorbu stránok využitím ASP.NET a architektonickým štýlom MVC, vytváranie Windows služieb a mnoho ďalších.

Získali sme nové skúsenosti s manažmentom operačného systému Windows Server. Dôkladne sme sa oboznámili s NoSQL databázou RavenDB, ktorú sme v našom projekte aj použili.

Nakoľko sme mali k dispozícií oproti zimnému semestru ďalšie zariadenie na sledovanie pohľadu, využili sme znalosti ohľadom refaktoringu pri rozširovaní podpory klienta pre toto zariadenie.

V rámci konferencie IIT.SRC sme na jednej strane získali praktické skúsenosti ohľadom prezentovania väčšieho projektu a na druhej strane sme mali možnosť lepšie spoznať druhé zariadenie, ktoré sme mali k dispozícií. Prišli sme na dôležité zistenia, ktoré sa týkali jeho nedostatkov, ktoré sme prezentovali na stretnutí UX skupiny fakulty.

Vďaka tímovému projektu sme získali mnoho užitočných poznatkov a nazbierali sme dôležité skúsenosti pre budúcu prácu.

PRÍLOHA A - Výstupy analýz

A.1 ACDC – 1. šprint

A.1.1 Formát dát zo zariadenia

Čo obsahuje packet získaný zo zariadenia:

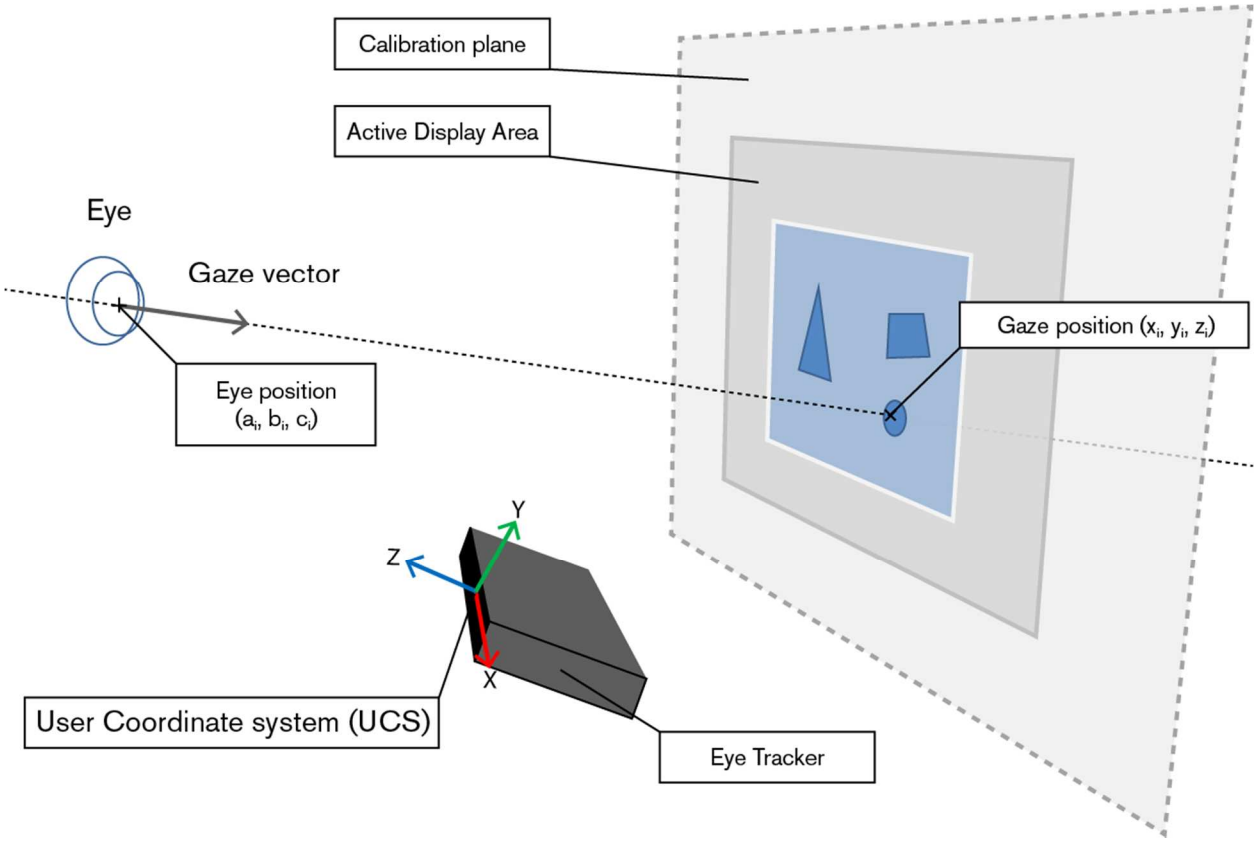
- Timestamp
- Polohu očí (UCS)
- Relatívnu polohu očí
- 3D bod pohľadu
- 2D bod pohľadu
- Validačný kód
- Priemer zreničky

Timestamp:

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

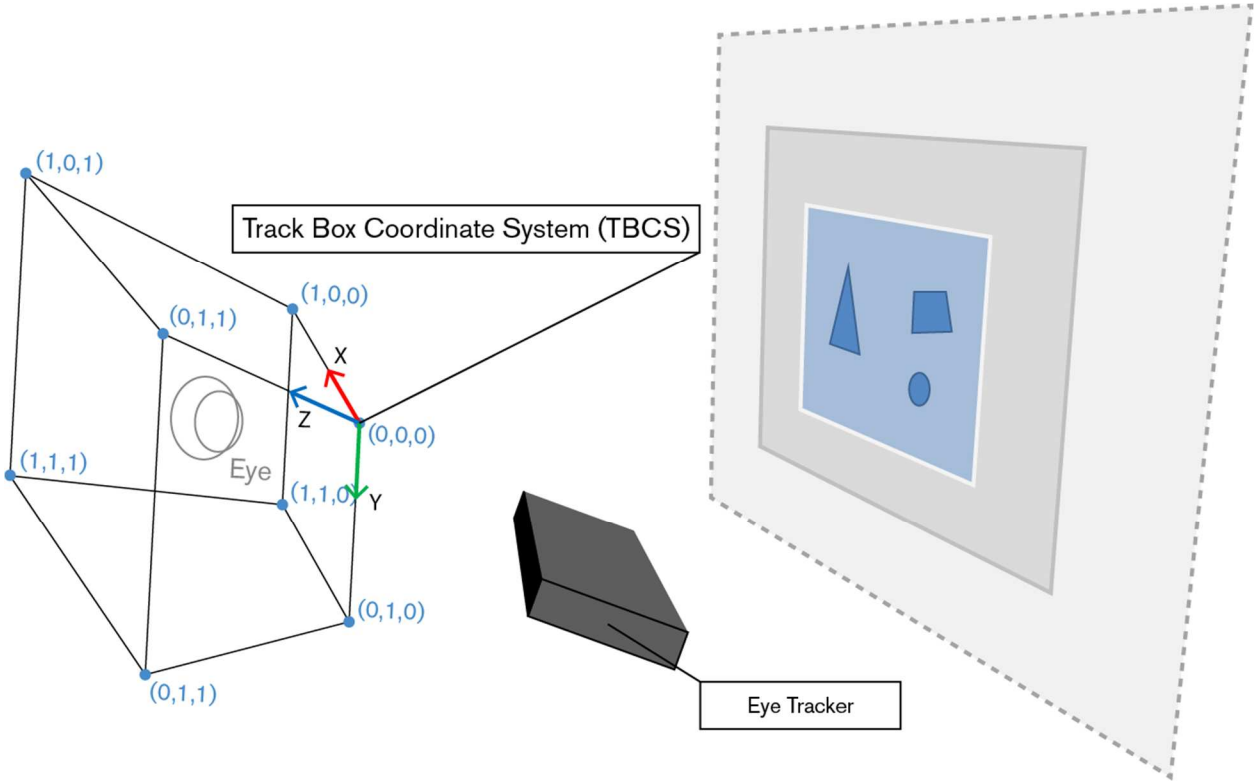
Polohu očí:

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



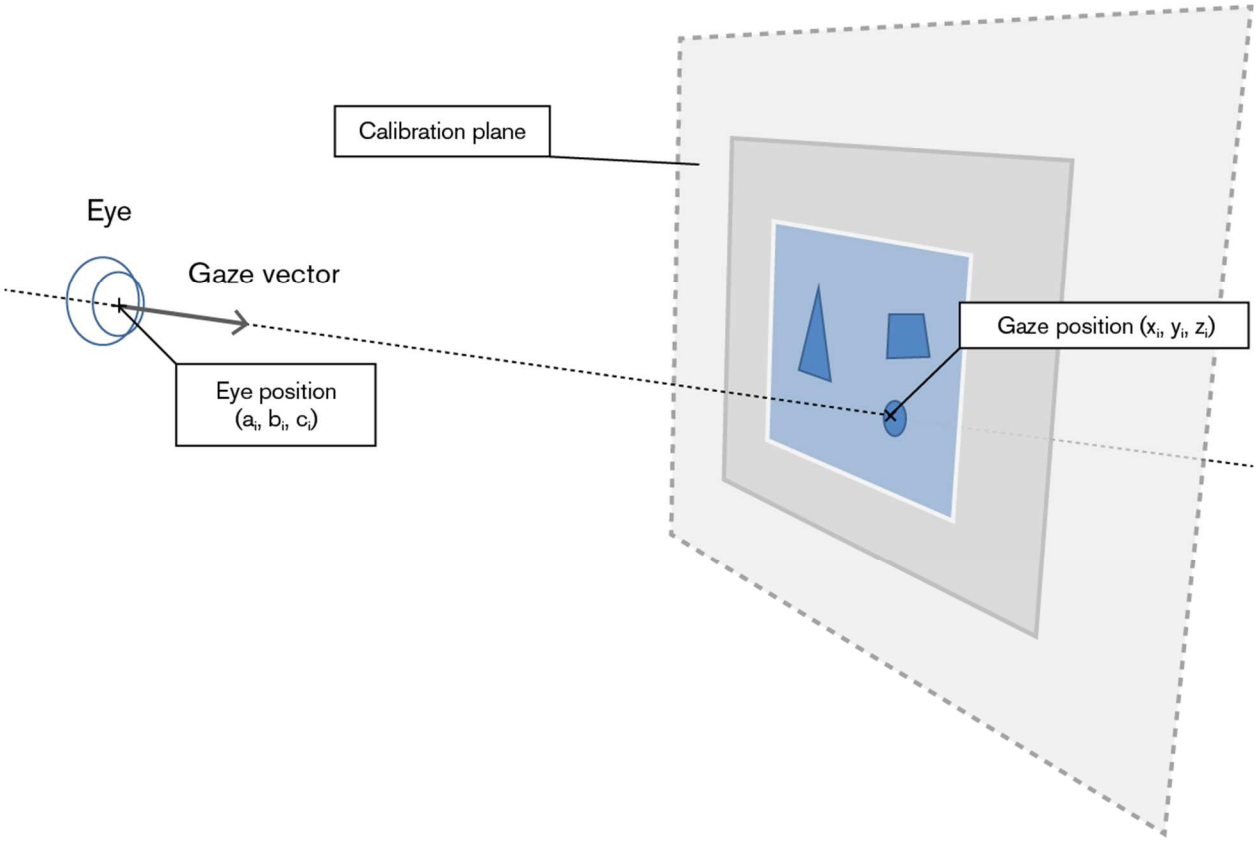
Relatívna poloha očí:

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.



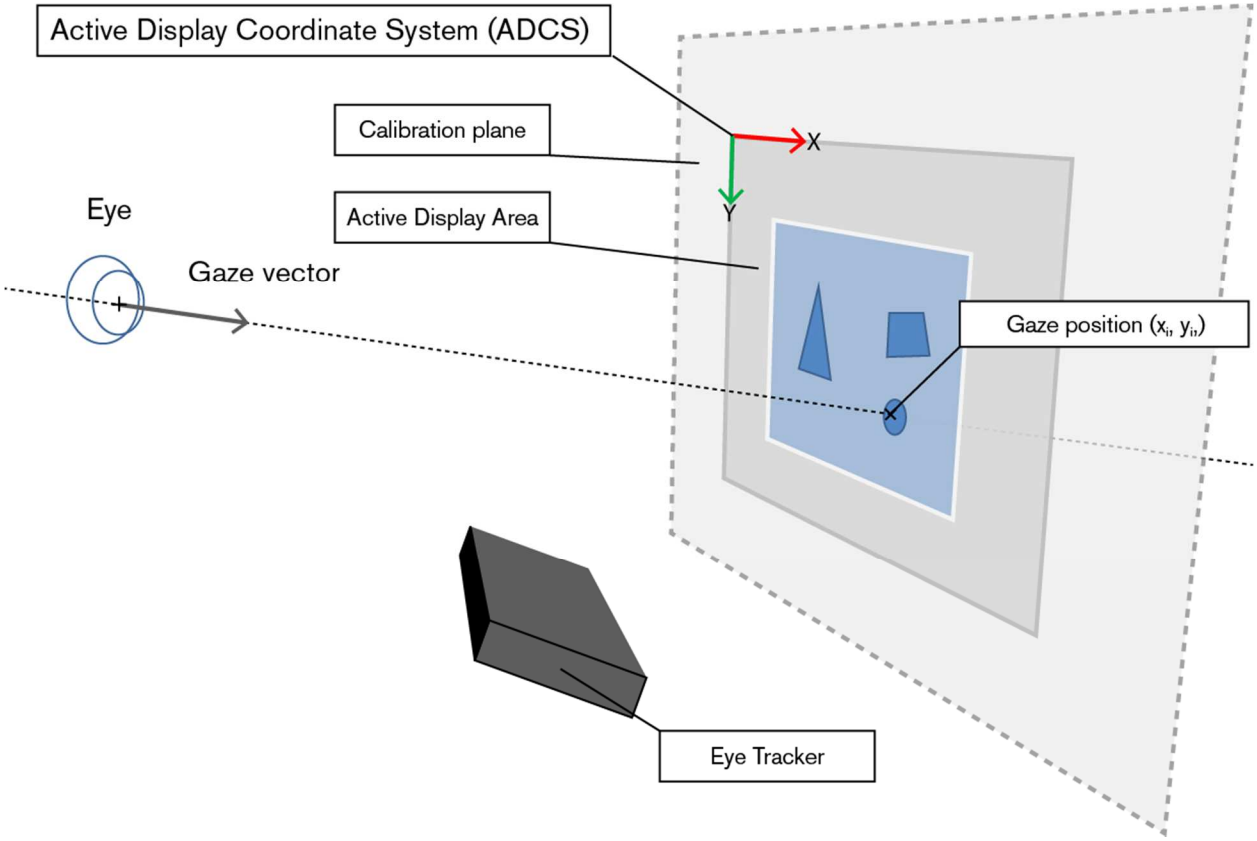
3D bod pohľadu:

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.



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 podobnejši e ľavé oko
-//-	1	N/A	N/A	N/A	Našlo jedno oko. Pravdepod obne ľavé.	N/A
-//-	2	N/A	N/A	Našlo jedno oko. Nevie ktoré.	N/A	N/A
-//-	3	N/A	Našlo jedno oko. Pravdepod obne pravé.	N/A	N/A	N/A
-//-	4	Našlo jedno oko. Najpravde podobnejši e pravé.	N/A	N/A	N/A	Nenašlo ani jedno oko

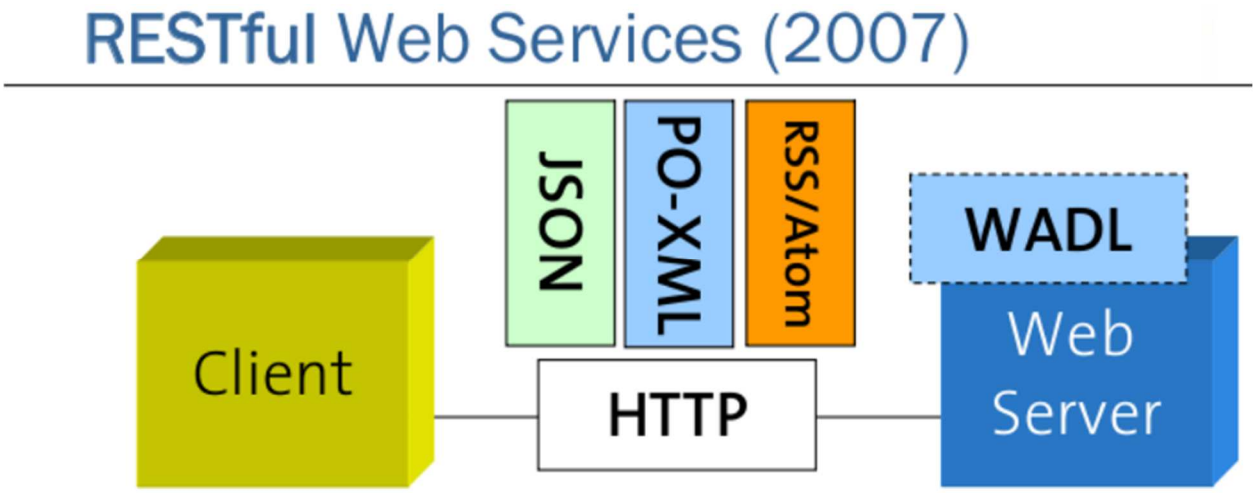
Priemer zreničiek:

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.



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

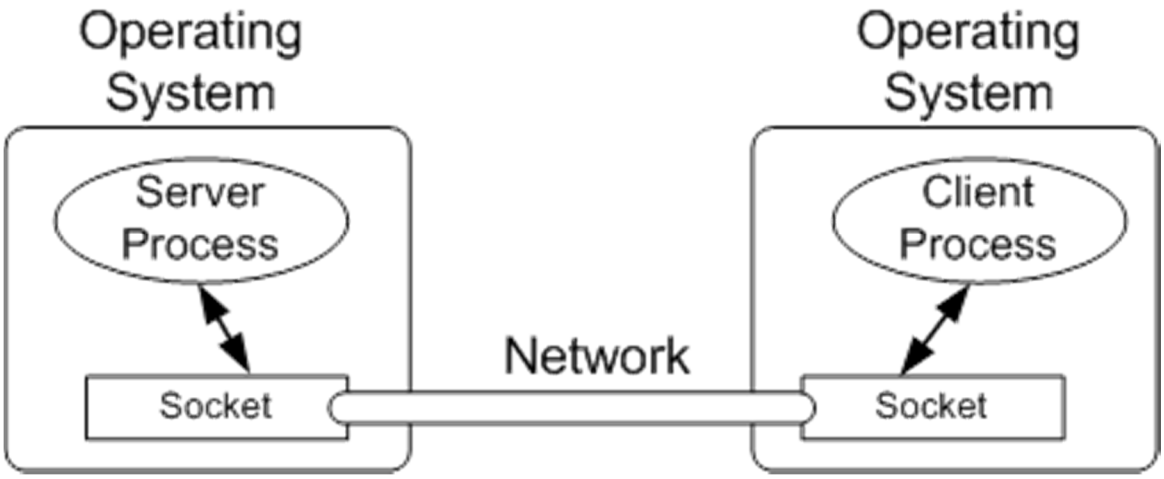
výhody:

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

nevýhody

1. viazaný na http
2. veľké množstvo objektov
3. správa URI namespacov 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 nastavia sa porty na klienstkej 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 ma 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 získať 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 informácii o okne - pozíciu, informácie o viditeľnosti atď.

Return: štruktúra window placement, popísana 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 = Mashal.sizeof(wp);

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

flag 0x0004

WPF_ASYNCHWINDOWPLACEMEN - 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 blokovaný

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

- 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 zobraziť prvýkrát

ptMinPosition - súradnice ľavého horného bodu okna, keď je minimalizované

ptMaxPoisition - 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 maximaluzejeme 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
}
```

PRÍLOHA B - Výstupy analýz z letného semestra

B.1 TheEyeTribe Tracker

Tento dokument opisuje moje dojmy z nového eye trackera, ktorý máme na škole od TheEyeTribe. Tracker je opisovaný hlavne z pohľadu tvorenia aplikácií preňho a jeho porovnania so starším trackerom od Tobii. Tracker bude opísaný z týchto hľadísk:

- Pripojenie na tracker
- Kalibrácia
- Posielané dáta
- Zachytávanie pohľadu - presnosť a dojmy

B.1.1 Pripojenie na tracker

Pre pripojenie na nový tracker sa využíva serverová aplikácia, ktorá komunikuje so zariadením a treba ju spustiť vždy, keď chceme zariadenie používať. Naša aplikácia potom komunikuje s týmto serverom. So serverom môžeme komunikovať buď použitím dllky poskytnutej v SDK, prípadne môžeme komunikáciu spracovávať manuálne spracovávaním JSONov, ktoré posiela server.

Server si vytvára log súbory, ktoré obsahujú informácie napr o pripojení na zariadeni po zapnutí aplikácie, ktorá chce zariadenie používať, prípadne informáciu o odpojení/spadnutí pripojenia po jej skončení.

B.1.2 Kalibrácia

V rámci SDK je aj jednoduchá aplikácia pre kalibráciu trackera, ktorá následne umožňuje zariadenie použiť napr. pre pohyb kurzora myši. Kalibrácia sa síce najprv spustí automaticky, no je ju možné potom v UI nastaviť.

Čo ma zaskočilo bol počet kalibračných bodov, kde my používame len 5, čo bolo odôvodnené dostatočnou presnosťou kalibrácie a minimalizovaním otravovania používateľa. V tejto aplikácii je ale možné nastaviť 9, 12 alebo až 16 kalibračných bodov. Zdá sa mi to trochu prehnané, ale možno je pre to nejaké odôvodnenie, ktoré som zatiaľ nenašiel.

V ich verzii kalibrácii sa taktiež kalibračný kruh nepohybuje po obrazovke, ale priamo objaví na danom mieste, čo môže výrazne šetriť čas. Taktiež kalibračný kruh úplne v našej implementácii úplne zmizne pri kalibrovaní daného bodu, no v tejto implementácii sa len trochu zmenší. Mne osobne príde ich spôsob asi trochu lepší, občas sa mi totiž stane, že po zmiznutí kalibračného kruhu trochu stratím sústredenie na daný bod.

Rozdiel je aj pri vyhodnocovaní výsledkov kalibrácie, kde Tobii štandardne zobrazuje odchýlky používateľovho pohľadu od kalibračných bodov. Tu ale kalibráciu vyriešili inak - jednoduchým slovným popisom, kde pri najlepšej kalibrácii aplikácia jednoducho vypíše perfect a ohodnotí kalibráciu hviezdičkami. Po preskúmaní zdrojového kódu jednej sample aplikácie som zistil, že kalibrácia sa ohodnocuje na základe atribútu presnosti AverageErrorDegree. Tento atribút obsahuje hodnoty od 0 po pravdepodobne 1.5, kde v sample aplikácii bola hodnota menšia ako 0.5 označená ako perfect.

Serverová aplikácia si pamäťá, či už bolo zariadenie kalibrované. Sample aplikácia, ktorú som testoval, umožňuje scrollovať pomocou pohľadu. Ak nebolo zariadenie predtým nakalibrované, aplikácia pri spustení oznámi túto skutočnosť a nespustí sa. Skúšal som tento check vypnúť no po zapnutí aplikácia na pohľad nereagovala, čo je ďalší rozdiel oproti Tobii zariadeniu. Tobii zariadenie si v sebe pamätá poslednú kalibráciu, ktorá bola na ňom vykonaná. Nový tracker ale pôsobil dojmom, že bez kalibrácie vôbec nefunguje (aspoň pri tejto sample aplikácií).

B.1.3 Posielané dáta

Komunikácia je založená na základe získavania JSONov zo serverovej aplikácie, ktoré sú dllkou automaticky deserializované, prípadne ich musíme spracovávať manuálne. Dáta obsahujú nasledovné informácie:

- frame
  - int time - timestamp
  - bool fix - máme v tomto frame fixáciu?
  - int state - stav očí, môže obsahovať 5 hodnôt - gaze, eyes, presence, fail, lost - ktoré označujú, ako dobre vidí tracker používateľove oči
  - raw - obsahuje inty x a y pre 2d pozíciu používateľovho pohľadu v pixeloch
  - avg - podobne ako raw ale súradnice sú vyhladzované
- lefteye - informácie o ľavom oku
  - raw - x a y koordináty v pixeloch
  - avg - x a y koordináty v pixeloch, vyhladzované
  - float psize - veľkosť zreničky
  - pcenter - floaty x a y - normalizované súradnice zreničky
- righteye - informácie o pravom oku, rovnaké ako pre ľavé oko

Ako je vidno z tohto popisu, narozdiel od Tobii trackera nemáme k dispozícii napr. 3d koordináty pohľadu a očí. Máme ale už zo zariadenia spracované koordináty do jedného bodu a nemusíme ich priemerovať automaticky. Taktiež je k dispozícii vyhladzovanie, ktoré zabezpečuje, že koordináty budú menej “skákať” a snažiť sa priblížiť k tomu, kam sa skutočne používateľ pozerá.

B.1.4 Zachytávanie pohľadu

Pri testovaní trackera som sa snažil porovnať jeho presnosť so zariadením od Tobii. Na základe iba mojich dojmov môžem povedať, že trackeri majú podobnú presnosť. Tu by trebalo vykonať podrobnejšie testy. Čo som si ale všimol bolo, že na rozdiel od Tobii trackera sa TheEyeTribe tracker oveľa rýchlejšie dokáže zotaviť zo straty používateľových očí. Nerobil som síce presné merania, ale myslím že to vždy stihol do jednej sekundy, kde Tobii tracker dokázal byť stratený aj 5-10 sekúnd.

PRÍLOHA C - Používateľská príručka pre browser addon

Používateľska príručka pre browser addon

Autor: Kristína Mišíková, Lukáš Gregorovič
Dátum: 03.05.2014
Verzia: 1

C.1 Úvod

Príručka popisuje postup ako získať, nainštalovať a používať rozšírenia pre prehlidače Mozilla Firefox a Google Chrome.

C.2 Inštalačná príručka

Inštalácia rozšírenia pre prehliadač je pomerne jednoduchá. V prípade Google Chrome, ale aj Mozilla Firefox je nutné stiahúť požadovanú verziu rozšírenia.

C.2.1 Krok 1 - stiahnutie inštalačného súboru

Súbory potrebné pre inštaláciu rozšírení je možné stiahnuť z webovej stránky:
http://team01-13.ucebne.fiit.stuba.sk/web/Home/Downloads

Podľa zvoleného prehliadača je možné stiahnuť dve verzie:

- Google Chrome verzia
- Firefox verzia

Súbory stiahnite na ľahko prístupne miesto, napríklad na plochu.

C.2.2 Krok 2 - rozbalenie inštalačného súboru

Stiahnutý súbor je nutné rozbaliť použitím komprimačného nástroja, napríklad WinRar. (Pravdepodobne však postačí aj základný nástroj dodávaný s Windows)

Obsah súboru si dočasne umiestnite na plochu. V prípade Firefox verzie súbor obsahuje iba jednu inštalačku s príponou .xpi a Google Chrome verzia obsahuje dve inštalačky (klientsú a administrátorskú verziu) s príponou .crx

C.2.3 Krok 3 - spustenie inštalačného súboru a postup inštalácie

Inštalácia rozšírenia prebieha jednoducho. V prvom rade je nutné otvoriť požadovaný internetový prehliadač (podľa verzie rozšírenia).

Ak sa jedná o firefox, súbor s príponou .xpi stačí uchytením na ploche presunúť do okna prehliadača.

V prípade Google Chrome je nutné otvoriť v prehliadači panel rozšírení (Menu / Nástroje / Rozšírenia). Následne stačí opäť premiestniť požadovaný súbor z plochy do okna prehliadača.

C.3 Používateľská príručka

Hlavná funkcionalita oboch verzii rozšírení je rovnaká, avšak existujú menšie rozdiely v prevedení rozšírení. Niektoré odlišnosti pri používaní rozšírení sú uvedené v ďalšej časti.

C.3.1 Základná funkcionalita

Rozšírenie pre prehliadač plní dve funkcie.

- Hlavnou funkciou je obohacovanie dát zo sledovania pohľadu, ktoré zo zariadenia získava náš desktopový klient. Nazbierané dáta sú odoslané rozšíreniu a to následne pre jednotlivé súradnice pohľadu k dátam priradí identifikátor konkrétneho elementu stránky, na ktorý sa používateľ pozeral.
- Druhou funkcionalitou je umožnenie vytvárania takzvaných oblastí záujmu. Tie sú reprezentované identifikátorom elementu, ktorý tvorí ohraničenie oblasti, url adresou stránky a doplňujúcimi informáciami ako sú názov a popis.

Obohacovanie dát zo sledovania je možné jednoduchým aktivovaním rozšírenia v prehliadači.

Definovnie oblastí záujmu vyžaduje prihlásenie sa cez **gazeadmin** a zvoliť si sedenie, pre ktoré chceme oblasť definovať (Projects -> Edit project / Sessions -> Edit session / Areas Of Interest). V pravej časti záložky sa nachádza ovládacie tlačidlo pre aktiváciu rozšírenia.

Obr. Tlačilo pre aktiváciu a deaktiváciu rozšírenia

Po aktivovaní rozšírenia kliknutím na tlačidlo je možné vytvárať oblasti záujmu (nie na gaze admin stránke). Pozastavenie a aktivovanie rozšírenia pre posledné aktivované sedenie je možné kliknutím na ikonku rozšírenia v prehliadači.

Zmenu sedenia, pre ktoré je vyžadované vytvorenie oblasti záujmu je možné urobiť len cez **gazeadmin** podobne ako pri prvej aktivácií. Rozšírenie dokáže editovať oblasti len pre jedno sedenie súčasne. Aktivovaním editácie pre sedenie sa zmení stav editácie predchádzajúceho sedenia na neaktívny.

C.3.2 Odlišnosti v implementácii rozšírení

Mozilla Firefox

Obe funkcionality rozšírenia sú implementované v jednom rozšírení

Google Chrome

Rozšírenie je rozdelené podľa funkcionality na dve časti

- administrátorská - zabezpečuje definovanie oblastí záujmu
- klientská - umožňuje obohacovanie dát zo sledovania

PRÍLOHA D - Používateľská príručka pre CarrotsClient

Používateľska príručka pre CarrotsClient

Autor: Jakub Daráž
Dátum: 2.5.2014
Verzia: 1

D.1 Úvod

Táto príručka je určená pre našu klientskú aplikáciu CarrotsClient. Aplikácia slúži na získavanie dát zo zariadenia pre sledovanie pohľadu, ich posielanie rozšíreniam pre webový prehliadač na obohacovanie a následné posielanie na server, kde sa ďalej spracúvajú a ukladajú do databázy. Implementovaná je v jazyku C# pomocou Windows Forms.

Dokument obsahuje inštalačnú a používateľskú príručku.

D.2 Inštalačná príručka

V tejto časti je popísané, ako aplikáciu CarrotsClient nainštalovať.

D.2.1 Krok 1 - stiahnutie inštalačného súboru

Inštalačný súbor je možné stiahnuť na adrese: http://team01-13.ucebne.fiit.stuba.sk/web/Home/Downloads

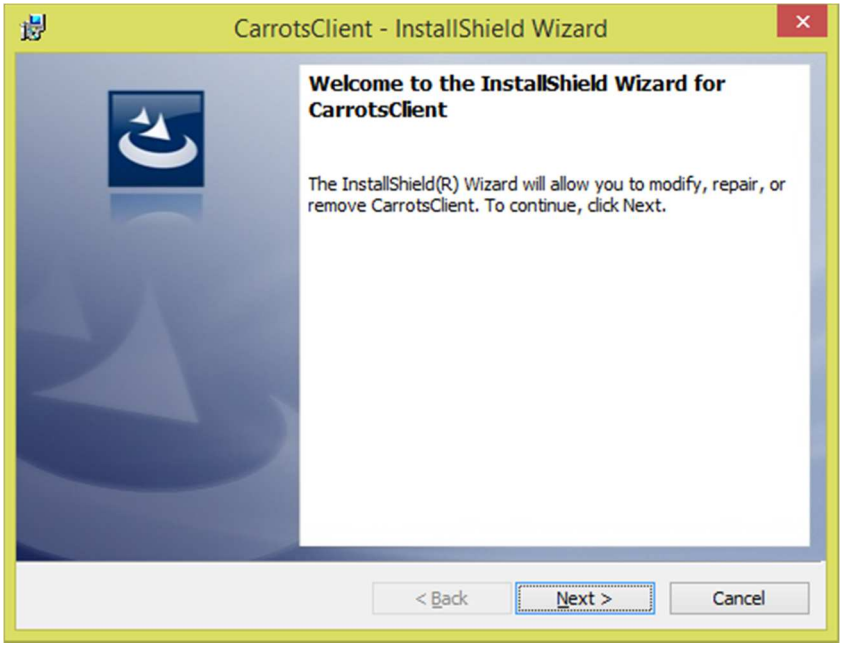
V sekcii Desktop Client. Súbor sa stiahne zabalený vo formáte zip a je ho potrebné rozbaliť.

D.2.2 Krok 2 - rozbalenie inštalačného súboru

Súbor je najprv potrebné rozbaliť pomocou vášho obľubeného komprimačného programu. Použiť je možné napr. 7-zip.

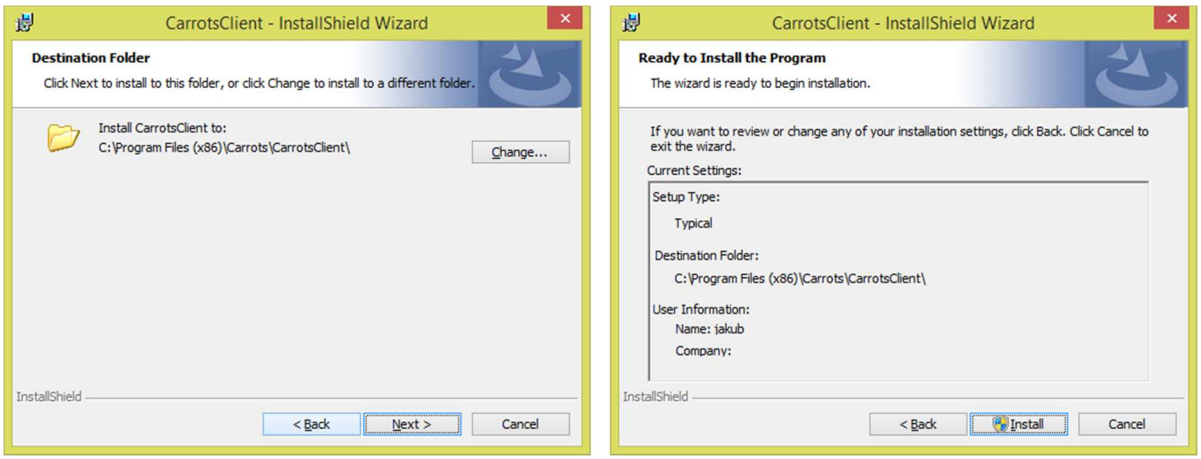
D.2.3 Krok 3 - spustenie inštalačného súboru a postup inštalácie

Po rozbalení inštalačného súboru ho spustíme dvojklikom a zobrazí sa nám nasledovné okno.



Obrázok 1 - uvítacie okno inštalátora

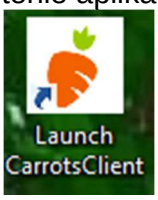
V tomto okne jednoducho klikneme na Next > a pokračujeme ďalej.
V ďalšom okne si môžeme zvoliť adresár, do ktorého sa má aplikácia nainštalovať, kliknutím na Browse. Keď sme spokojný s adresárom, klikneme znova na Next > a následne na Install.



Obrázok 2 - výber inštalačného adresára a spustenie inštalácie

Po prebehnutí inštalácie si na poslednej obrazovke môžeme vybrať, či chceme nainštalovanú aplikáciu aj priamo spustiť a kliknutím na Finish inštaláciu ukončíme.

Inštalátor automaticky vytvorí pre spustenie aplikácie odkaz na pracovnej ploche.



Obrázok 3 - odkaz na spustenie aplikácie na pracovnej ploche

D.3 Používateľská príručka

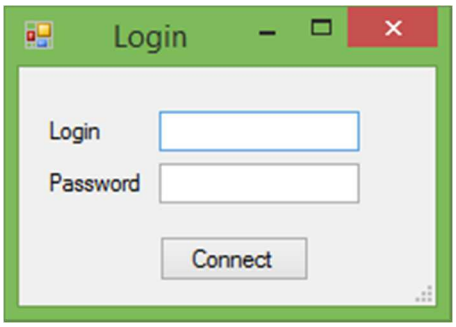
Táto časť opisuje každú obrazovku aplikácie a aké akcie sa v nej dajú vykonať.

D.3.1 Spustenie aplikácie

Aplikáciu je možné jednoducho spustiť pomocou odkazu na pracovnej ploche. Po spustení aplikácie sa zobrazí obrazovka pre prihlásenie.

D.3.2 Obrazovka pre prihlásenie

Do obrazovky treba zadať používateľské meno a heslo a následne kliknúť na tlačidlo connect. Prihlásiť sa je možné pomocou AIS konta, no používateľ sa musí nachádzať v našej databáze. Ak potrebujete vytvoriť konto, môžete nám poslať email na adresu tp1314team1@gmail.com.



Obrázok 4 - obrazovka pre prihlásenia

Po úspešnom prihlásení môžu nastať dva scenáre:

- ak má používateľ pripojené práve jedno zariadenie pre sledovanie pohľadu, automaticky sa aplikácia pripojí na toto zariadenie a spustí sa kalibrácia
- ak má používateľ pripojených viac, alebo žiadne zariadenie pre sledovanie pohľadu, zobrazí sa mu obrazovka pre výber zariadenia, na ktoré sa chce pripojiť

D.3.3 Výber zariadenia pre sledovanie pohľadu

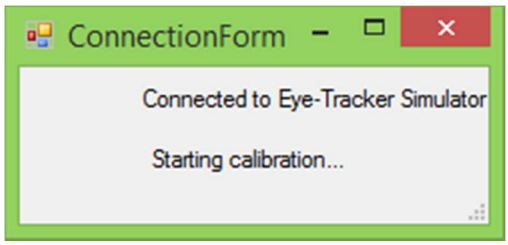
Zariadenia môžeme zobraziť kliknutím na tlačidlo Refresh list. Následne kliknutím môžeme vybrať zariadenie a pripojiť sa naň kliknutím na tlačidlo Connect to tracker.

Obrázok 5 - obrazovka pre výber zariadenia pre sledovanie pohľadu

Po pripojení na zariadenie sa automaticky spustí kalibrácia zariadenia.

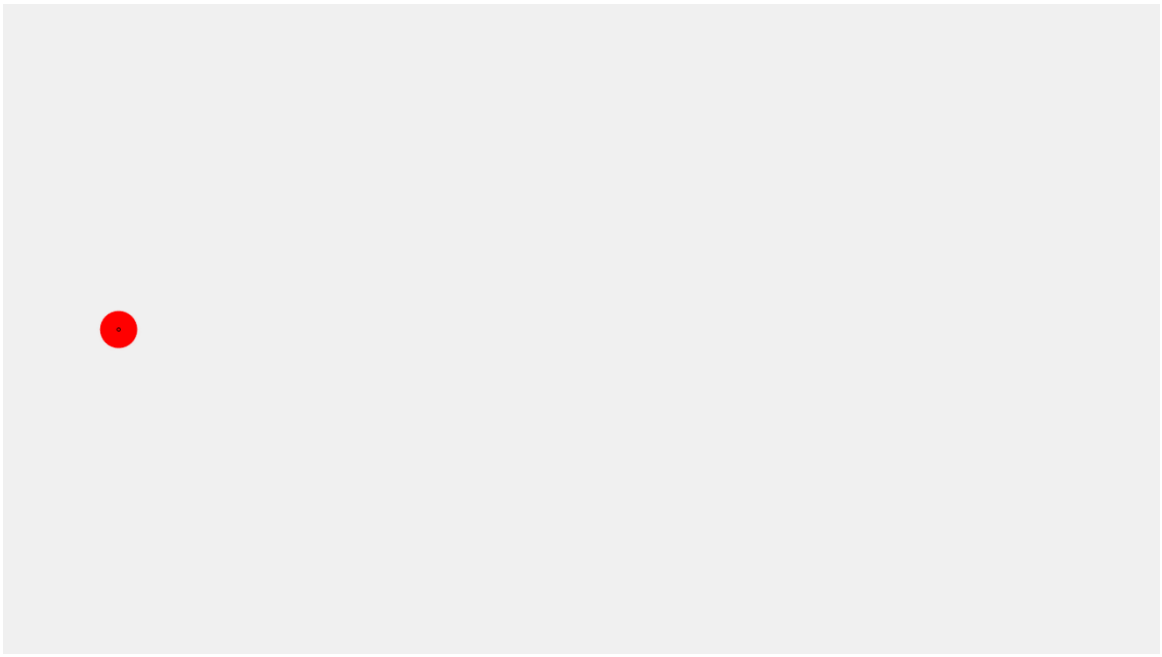
D.3.4 Kalibrácia zariadenia

Po pripojení na zariadenie sa zobrazí oznámenie o spustení kalibrácie.



Obrázok 6 - oznámenie o spustení kalibrácie

Následne sa zapne samotná kalibrácia. Pri kalibrácii musí používateľ sledovať stred červenej guličky, kým sa nenakalibruje 9 bodov kalibrácie.

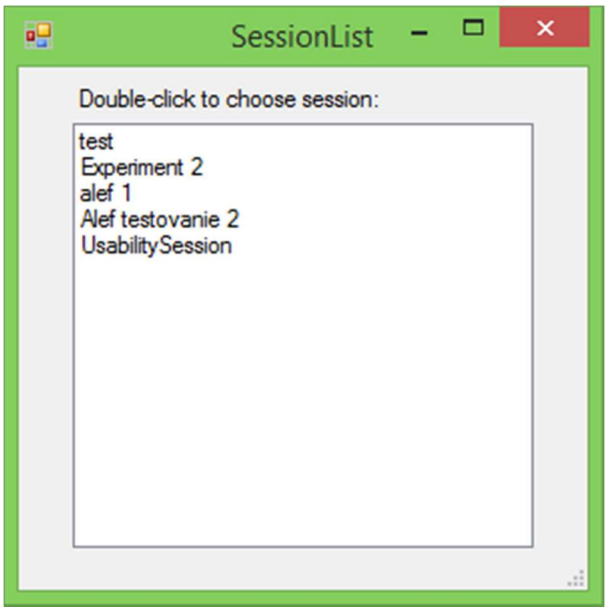


Obrázok 7 - priebeh kalibrácie zariadenia

Po kalibrácii zariadenia sa zobrazí výsledok kalibrácie. Výsledok kalibrácie vyzerá rôzne pre rôzne zariadenia. Vždy ale obsahuje dve tlačidlá - jedno pre potvrdenie kalibrácie a jedno pre opätovné nakalibrovanie. V prípade, že kalibrácia nebola dobrá, môžeme zariadenie nakalibrovať znovu. Po ukončení kalibrácie sa automaticky zobrazí obrazovka pre výber sedenia (session).

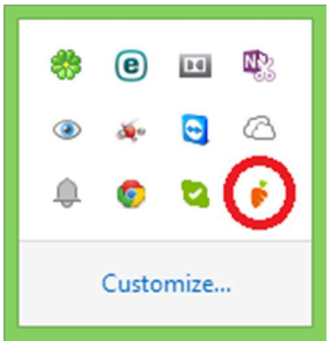
D.3.5 Obrazovka pre výber sedenia

V tejto obrazovke sú jednoducho vypísané aktívne sedenia, na ktoré je používateľ prihlásený. Pre pokračovanie si používateľ musí dvojklikom musí vybrať sedenie, na ktoré sa chce aktuálne prihlásiť.



Obrázok 8 - výber sedenia

Po výbere session sa aplikácia automaticky minimalizuje a začne sledovať používateľov pohľad. Aplikáciu je možné zobraziť pomocou dvojkliku na ikonu mrkvičky v system tray.

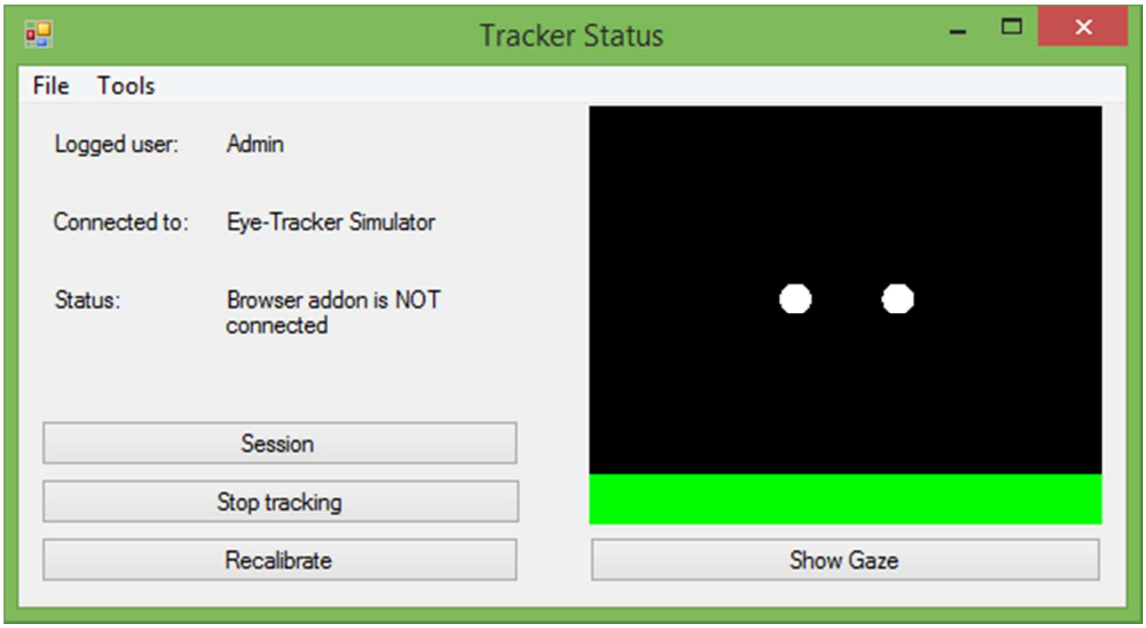


Obrázok 9 - ikona aplikácie v system tray

Po dvojkliku na ikonu sa zobrazí obrazovka so stavom aplikácie

D.3.6 Obrazovka so stavom aplikácie

Obrazovka so stavom aplikácie predstavuje hlavnú obrazovku aplikácie a obsahuje všetky dôležité informácie o stave aplikácie.



Obrázok 10 - obrazovka stavu aplikácie

Obrazovka so stavom aplikácie obsahuje v ľavom hornom rohu rôzne informácie:

- Logged user: prihlásený používateľ - meno zadávané v prvej obrazovke
- Connected to: meno trackera, na ktorý sa používateľ pripojil
- Status: zobrazuje, či je pripojené rozšírenie pre prehliadač

V ľavom dolnom rohu sa nachádza niekoľko tlačidiel:

- Session - tlačidlo pre zmenu sedenia, ak je to nutné
- Start/Stop tracking - tlačidlo pre spustenie/zastavenie zachytávania dát z pohľadu používateľa
- Recalibrate - tlačidlo, pre opätovnú kalibráciu zariadenia, ak je potrebná

V pravej časti sa ďalej nachádza grafické znázornenie umiestnenia používateľových očí v priestore, zobrazené ako biele body v čiernom poli. Pod čiernym poľom sa nachádza farebný pásik, ktorý svojou farbou znázorňuje stav zachytávania pohľadu:

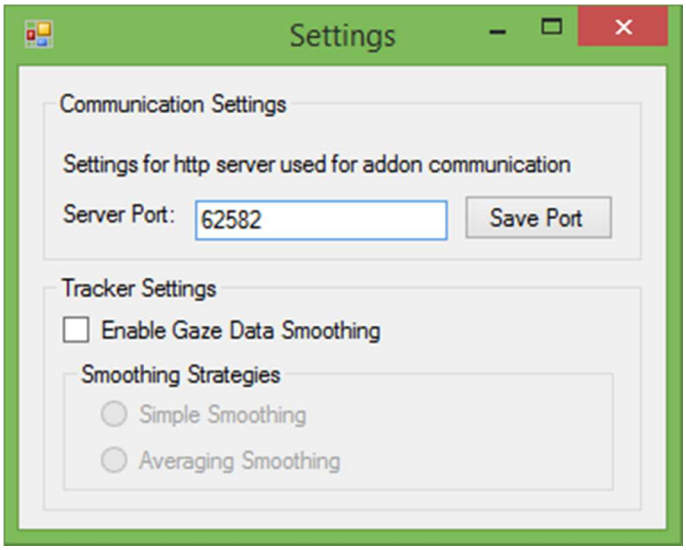
- šedá farba - zachytávanie pohľadu je vypnuté
- červená farba - zariadenie má problém so zachytením očí
- zelená farba - zariadenie v poriadku sleduje oči

Pod touto časťou sa nachádza tlačidlo Show Gaze, ktoré zobrazí formulár podobný kalibrácii. V tomto prípade ale nemá používateľ sledovať červenú guličku, naopak sa červená gulička nachádza tam, kam sa používateľ práve pozerá. Zavrieť sa táto obrazovka dá buď pozretím sa do pravého dolného rohu, alebo stlačením klávesovej skratky ALT+F4.

Okrem týchto častí obsahuje na vrchu okno stavu aplikácie navigačné menu. V položke File je zatiaľ možné iba vypnúť aplikáciu. V položke Tools sa dá kliknutím na Settings prístupiť k nastaveniam klienta, popísaným v nasledovnej časti.

D.3.7 Nastavenia klienta

Nastavenia klienta momentálne obsahujú dve položky: nastavenie HTTP servera, ktorý sa používa na komunikáciu s rozšíreniami pre prehliadač a nastavenie vyhladzovacej stratégie pre pripojené zariadenie pre sledovanie pohľadu.



Obrázok 11 - nastavenie HTTP servera

Pri serveri je možné v prípade potreby zmeniť port. Port je možné zmeniť prepísaním hodnoty Server Port na požadovaný port a stlačením tlačidla Save.

Pre povolenie vyhladzovania dát je potrebné zaškrtnúť Enable Gaze Data Smoothing a následne vybrať stratégiu vyhladzovania z dvoch možných:

- **Simple Smoothing** - berie sa každých 5 gaze data, ktoré prišli zo zariadenia a spraví sa z nich priemer. Tým pádom nám prakticky 5-násobne klesne framerate zariadenia.
- **Averaging Smoothing** - podobne ako pri predchádzajúcej stratégii sa počíta priemer z piatich gaze data, ale je to vždy 5 posledných. Framerate teda zostáva rovnaký. Je **odporúčané** využívať túto stratégiu.

PRÍLOHA E - Používateľská príručka pre Simulátor

Používateľska príručka pre Simulátor

Autor: Martin Janík
Dátum: 17. 5. 2014
Verzia: 1

E.1 Úvod

Príručka je určená pre aplikáciu simulátora pohybu očí. Príručka pozostáva z dvoch častí - inštalačnej a používateľskej príručky.

E.2 Inštalačná príručka

Táto časť uvádza postup inštalácie aplikácie simulátora.

E.2.1 Krok 1 - stiahnutie inštalačného súboru

Stiahneme aktuálnu verziu Simulátora zo stránky projektu: http://team01-13.ucebne.fiit.stuba.sk/web/Home/Downloads

E.2.2 Krok 2 - rozbalenie inštalačného súboru

Súbor rozbalíme pomocou obľubeného archivovacieho nástroja.

E.2.3 Krok 3 - spustenie inštalačného súboru a postup inštalácie

Spustíme inštalačný súbor aplikácie a riadime sa pokynmi sprievodcu inštalácie.

E.3 Používateľská príručka

Táto časť uvádza použitie aplikácie s popisom jej jednotlivých prvkov.

E.3.1 1. EyeTracker Simulator Information

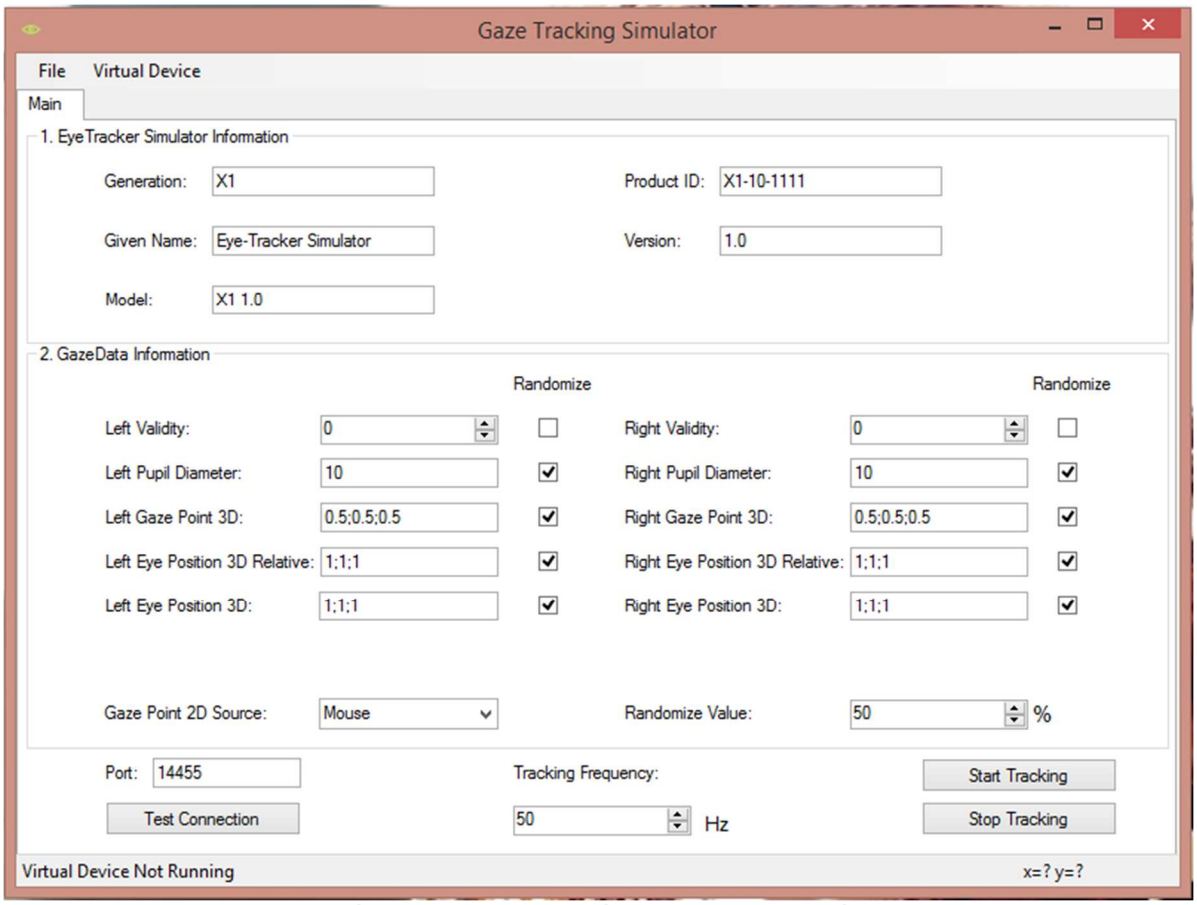
Táto časť nie je povinná. Obsahuje informatívne údaje o simulátore aby ho bolo možné nájsť jednoduchšie pomocou browsera poskytovaného firmou Tobii. Tieto údaje zodpovedajú údajom reálnych zariadení od firmy Tobii.

E.3.2 2. GazeData Information

V tejto časti sa nachádzajú informácie o pohľade pre obe oči, ktoré je možné simulovať. Validity môže nadobúdať hodnoty 0 až 4, pričom 0 predstavuje zaznamenané oko a hodnota 4, nezaznamenané oko. GazePoint 3D určuje polohu očí v priestore v rámci imaginárnej kocky projektovanej zariadením. Najdôležitejší údaj je Gaze Point 2D Source, ktorým vieme nastaviť vstup dát, ktoré určujú kam na obrazovku sa používateľ “pozeral”.

E.3.3 Nastavenia spustenia sledovania

Je možné nastaviť číslo portu, na ktorom má prebiehať komunikácia simulátora s ďalšími aplikáciami. Simulátor po spustení automaticky spúšťa komunikáciu. Broadcastom odosiela informácie z prvej časti simulátora na danom porte aby sa mohli iné aplikácie pripojiť k simulátoru. Tlačidlom “Test connection” môžeme testovať, či je simulátor prepojený s inou aplikáciou a tým pádom, či môžeme spustiť simuláciu sledovania pohľadu. Tlačidlami “Start Tracking” a “Stop Tracking” spúšťame a zastavujeme simuláciu sledovania pohľadu. Na najspodnejšej časti aplikácie sa nachádza riadok o stave simulátora spolu so súradnicami simulujúcimi pohľad na obrazovku.



Obrázok č. 1 - Grafické rozhranie Simulátora

PRÍLOHA F - Používateľská príručka pre webovú aplikáciu GazeAdmin

Používateľska príručka pre webovú aplikáciu GazeAdmin

Autor: Dominika Červeňová, Róbert Kocian, Michal Mészáros
Dátum: 20.5.2014
Verzia: 1

F.1 Úvod

Tento dokument popisuje spôsob práce s webovou aplkáciou GazeAdmin. Aplikácia slúži najmä na administráciu usability experimentov a používateľov v rámci. Dokument obsahuje URL aplikácie, základné informácie o nej a používateľskú príručku.

F.2 Inštalačná príručka

Inštalácia webovej aplikácie Gaze admin je pomerne jednoduchá, stačí zo zdrojového kódu vytvoriť release verziu projektu ViewTracking.Web, a aby fungovala aplikačná vrstva tak tiež projektu ViewTracking.Wcf. Následne upnúť do IIS na serveri obe aplikácie zvlášť a nastaviť v projekte web v jeho webconfigu cestu na svc subor Wcf projektu.

F.2.1 Krok 1 publishovanie ViewTracking.Wcf

Po pravom kliku na projekt sa zobrazí kontextové menu kde sa nacháda publish po nastavaní súboru pre publish stačí skopírovať kód.

F.2.2 Krok 2 nastavanie webconfigu Viewtracking.Wcf

1. **Nastavenie connection stringu na databázu:**

<add name="ModelContainer" connectionString="ms sql metadata" />

2. **Nastavenie enpointov pre raven db v časti client:**

<endpoint address="net.pipe://localhost/raven/GazeDataService.svc"

contract="GazeDataService.IGazeDataService" name="NetNamedPipeBinding_Raven" />

3. **Nastavenie Entity Frameworku v časti configuration:**

<section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />

<entityFramework>

<defaultConnectionFactory

type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

<parameters>

<parameter value="v11.0" />

</parameters>

</defaultConnectionFactory>

<providers>

<provider invariantName="System.Data.SqlClient"

type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

```
        </providers>
 </entityFramework>
 <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
        <dependentAssembly>
        <assemblyIdentity name="EntityFramework"
publicKeyToken="b77a5c561934e089" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
        </dependentAssembly>
        </assemblyBinding>
 </runtime>
```

4. Nastavenie service model časti vo webconfigu:

```
<system.serviceModel>
<client>
     binding="netNamedPipeBinding" bindingConfiguration="NetNamedPipeBinding_Raven"
     contract="GazeDataService.IGazeDataService" name="NetNamedPipeBinding_Raven"
/>
        </client>
        <bindings>
        <basicHttpBinding>
        <binding name="BasicHttpBinding_IGazeDataService" />
        <binding name="BasicHttpsBinding_IGazeDataService">
        <security mode="Transport" />
        </binding>
        </basicHttpBinding>
        <netNamedPipeBinding>
        <binding name="NetNamedPipeBinding_Raven" />
        </netNamedPipeBinding>
        <webHttpBinding>
        <binding name="pokus">
        <security mode="Transport"/>
        </binding>
```

```
<binding name="http" />
        </webHttpBinding>
</bindings>
<services>
<service behaviorConfiguration="serviceBehavior"
name="ViewTracking.Wcf.ViewTrackingService">
<endpoint address=""
binding="webHttpBinding" bindingConfiguration="pokus" behaviorConfiguration="pokus2"
contract="ViewTracking.Wcf.IViewTrackingService" />
<endpoint address=""   binding="webHttpBinding" bindingConfiguration="http"
behaviorConfiguration="pokus2" contract="ViewTracking.Wcf.IViewTrackingService" />

</service>
</services>
<behaviors>
<serviceBehaviors>
<behavior name="serviceBehavior">
<serviceMetadata httpsGetEnabled="true" httpGetEnabled="true"/>
<serviceDebug includeExceptionDetailInFaults="true"/>
</behavior>
</serviceBehaviors>
<endpointBehaviors>
<behavior name="pokus2">
        <webHttp/>
        </behavior>
</endpointBehaviors>
</behaviors>
<serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
</system.serviceModel>
```

5. Upnutie do IIS:

a. pravym klikom na stránku v IIS a klik na Add application

b. nastaviť cestu k súboru

c. nastaviť application pool na .net framework 4.0 classic

d. nastaviť názov, ktorý bude v URL

F.2.3 Krok 3 publishovanie ViewTracking.Wcf

Po pravom kliku na projekt sa zobrazí kontextové menu kde sa nachádza publish po nastavaní súboru pre publish stačí skopírovať kód.

F.2.4 Krok 4 nastavenie web configu ViewTracking.Web

1. nastavenie connection stringu pre WCF službu

Name parameter musí byť rovnaký ako v uvedenom príklade: <add name="WcfConnnection" connectionString="http://team01-13.ucebne.fiit.stuba.sk/wcf/ViewTrackingService.svc/" />

2. nastavenie authentifikačnej chybovej spravy: <authentication mode="Forms"><forms loginUrl="~/account/login" timeout="2880" /></authentication>

3. Upnutie do IIS:

a. pravym klikom na stránku v IIS a klik na Add application

b. nastaviť cestu k súboru

c. nastaviť application pool na .net framework 4.0 classic

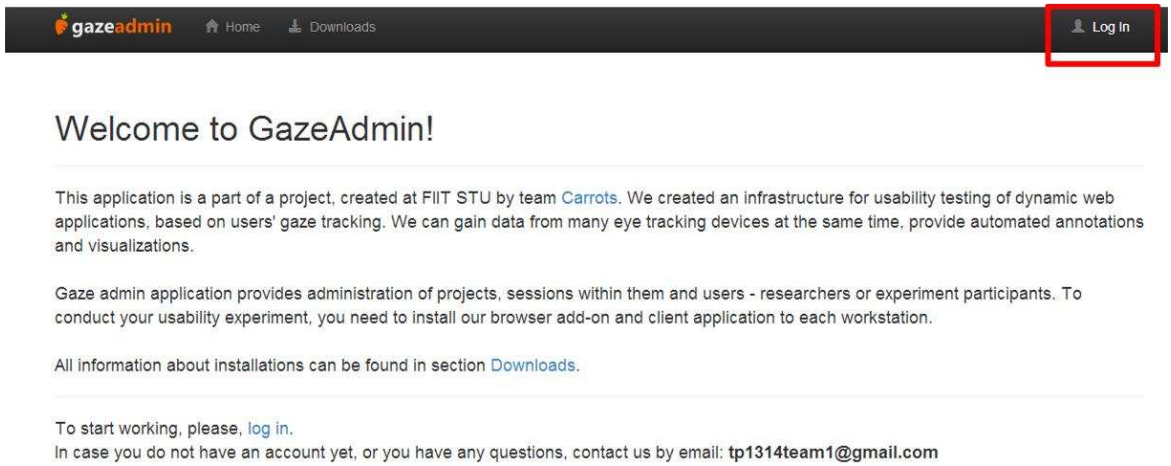
d. nastaviť názov, ktorý bude v URL

F.3 Používateľská príručka

Aplikácia GazeAdmin je dostupná na adrese: http://team01-13.ucebne.fiit.stuba.sk/web. Úvodná stránka obsahuje základné pokyny pre začiatok práce s aplikáciou. Je nutné najprv sa prihlásiť. Ak používateľ doposiaľ nemá konto, je na stránke uvedený kontakt, kam sa môže obrátiť so žiadosťou o jeho vytvorenie. Pre neprihláseného používateľa je dostupná okrem úvodnej aj stránka Downloads, kde sú dostupné na stiahnutie verzie inštalovateľných komponentov našej infraštruktúry:

- rozšírenia pre prehliadače Google Chrome a Mozillu Firefox
- desktopový klient
- simulátor

Ak používateľ má svoj účet, prihlási sa pomocou tlačidla Log in vpravo, v hornom menu (Obrázok 1).



Obrázok 1 - Úvodná stránka s vyznačeným tlačidlom na prihlásenie používateľa

Po prihlásení sa používateľovi sprístupnia ďalšie časti aplikácie prostredníctvom položiek menu. Základné sú administrácia experimentov, administrácia používateľov a zobrazenie štatistík. Všetky tri skupiny popisujeme v nesledujúcich častiach.

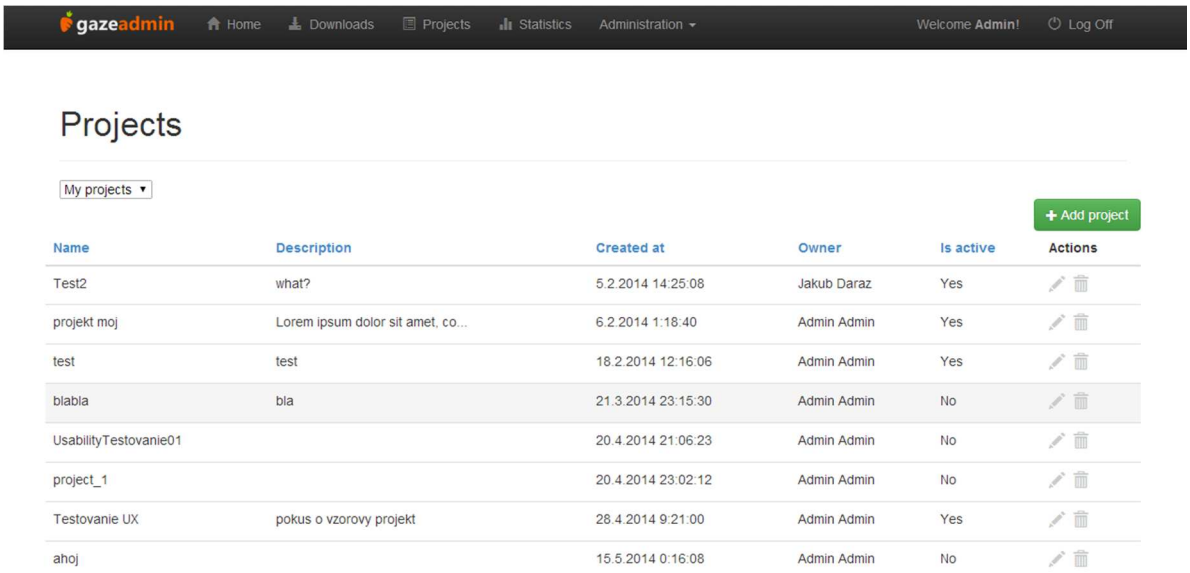
F.3.1 Administrácia experimentov

Na uskutočnenie experimentu je potrebné:

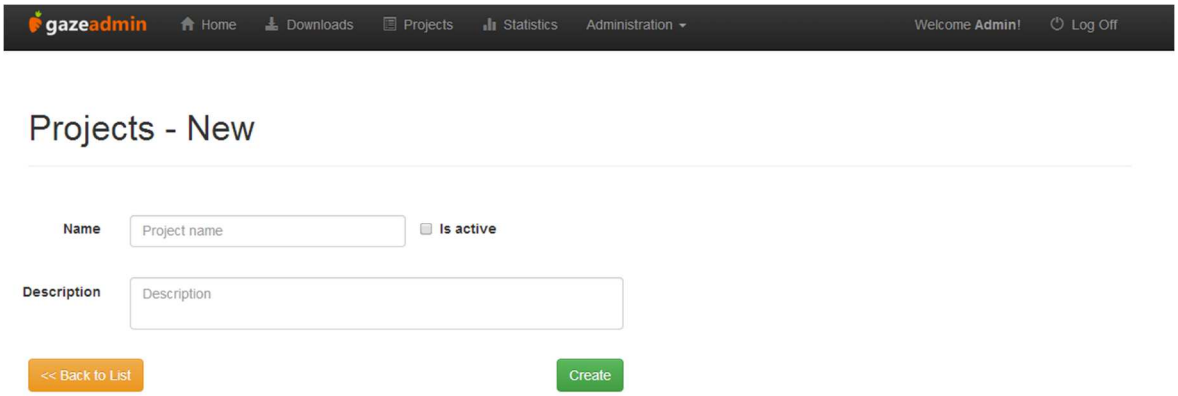
- vytvoriť projekt a pridať k nemu používateľov
- vytvoriť vrámci projektu sedenie/sedenia (sedenie = 1 UX experiment)
- pridať k sedeniu participantov
- určiť oblasti záujmu sledovanej aplikácie

Vytvorenie projektu

Na prístup k projektom slúži v hornom menu položka *Projects*. Používateľ uvidí zoznam všetkých projektov, ktoré vytvoril alebo bol na ne pridelený. V tabuľke v sĺpci *Actions* má možnosť upraviť alebo vymazať projekt a pomocou tlačidla *Add project* môže pridať nový projekt.

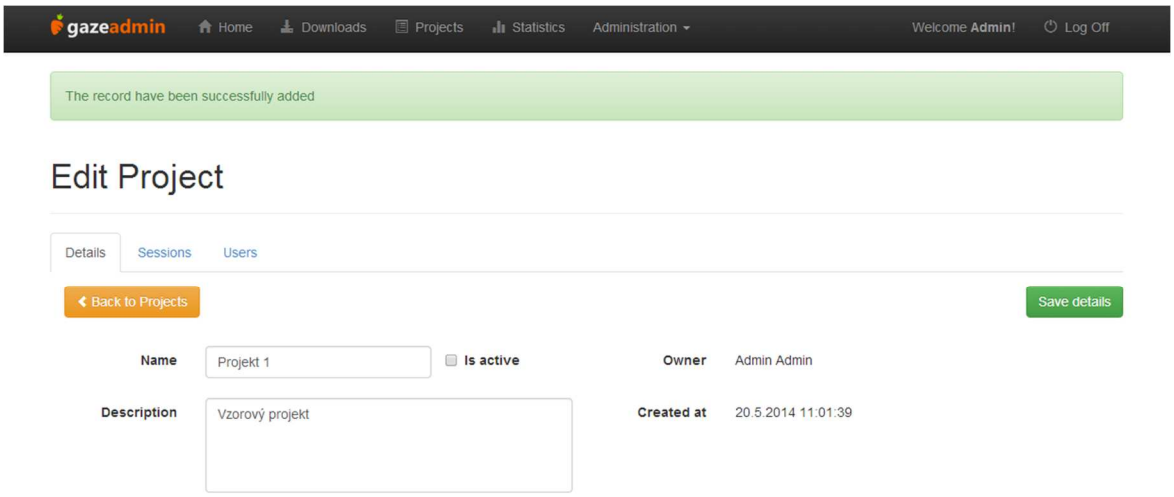


Obrázok 2 - Zoznam projektov, ktorý sa zobrazí po kliku na položku Project horného menu



Obrázok 3 - Vytváranie projektu

Po kliknutí na tlačidlo *Create* sa vytvorí nový projekt, ktorý sa rovno otvorí, pripravený na editovanie.



Obrázok 4 - Editovanie projektu

Stránka *Edit Project* (Obrázok 4) obsahuje tri záložky

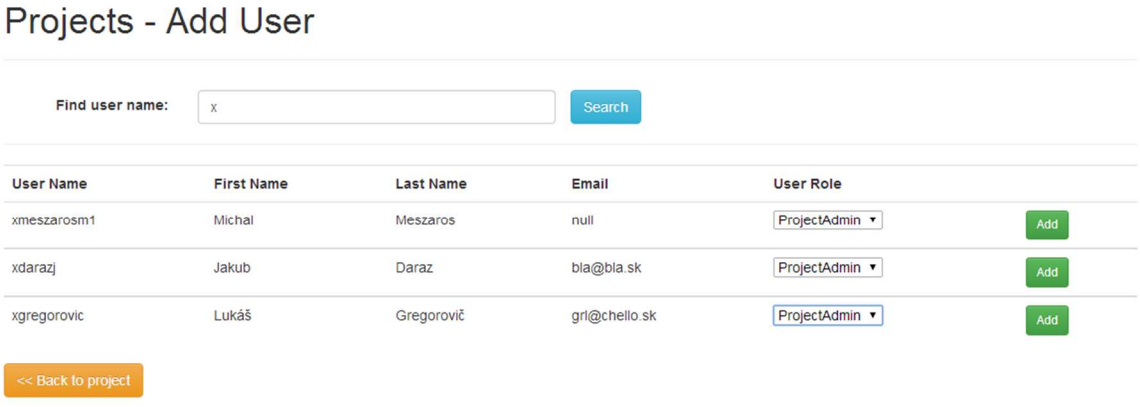
- **Details** - úprava zkladných informácií o projekte, uloženie zaleným tlačidlom Save details
- **Sessions** - možnosť prezerať vytvorené a pridávať nové sedenia (experimenty). Postup je analogický ako pri projekte
- **Users** - možnosť pridať k projektu používateľov s rôznou rolou

Pridávanie používateľov k projektu

V záložke Users používateľ vidí ľudí, pridaných na projekt/sedenie. Pridať ďalších môže pomocou tačila *Add users*. Pridávať k projektu možno len tých používateľov, ktorí sú registrovaní v našej databáze.

Postup: zadať meno > potvrdiť tlačidlom *Search* > zobrazí sa zoznam nájdených používateľov, z ktorých je možné vybrať a pridať ich k projektu alebo sedeniu s vybranou rolou (obrázok 5). Exitujú dva druhy rolí používateľa na projekte:

- ***Project admin*** - môže prezerať a upravovať daný projekt a sedenie v rámci neho
- ***Researcher*** - môže si prezerať dáta o danom projekte, nemôže upravovať

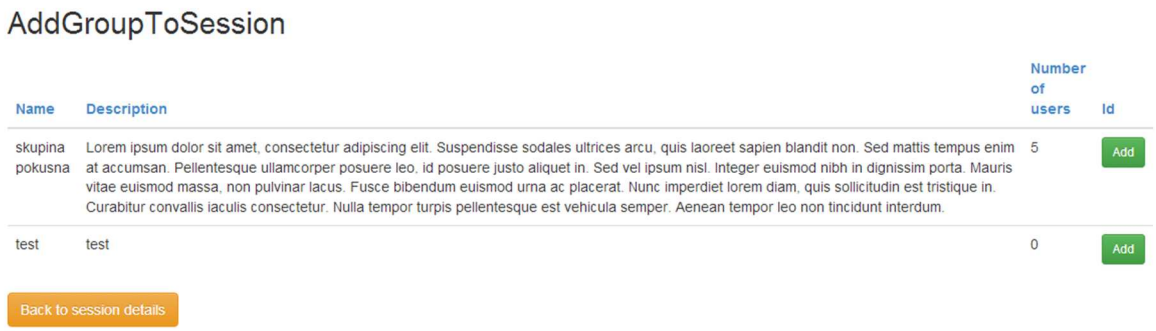


Obrázok 5 - Pridávanie používateľa k projektu

Vytvorenie sedenia v rámci projektu
Pridávanie session (sedenia) funguje podobne, ako vytváranie projektov. Po vytvorení sa podobne ako pri projekte otvorí stránka Edit, kde možno upraviť detaily, pridať používateľov - participantov a definovať oblati záujmu vo svojej aplikácii.

Pridávanie participantov k sedeniu
K sedeniu sa pridávajú len používatelia-participanti, nie je možné nastaviť im rolu. Pridať používateľov k sedeniu v časti Edit session dvoma spôsobmi:

1. ***po jednom*** - pomocou tlačidla *Add users*, podobne ako pri projektoch
2. ***po skupinách*** - pomocou tlačidla *Add group of users* je možné pridať vopred definovanú skupinu používateľov. Pridanie používateľov pomocou skupiny je výhodné najmä v prípade nutnsti pridania viacerých používateľov naraz. Skupina musí byť definovaná vopred (definovanie skupiny je popísané v časti Administrácia používateľov nižšie). Po kliknutí na Add group of users sa zobrazí zoznam vytvorených skupín. Jednoduchým klikom na tlačidlo Add sa pridajú všetci používatelia príslušnej skupiny k sedeniu.



Obrázok 6 - Pridávanie skupiny používateľov k sedeniu

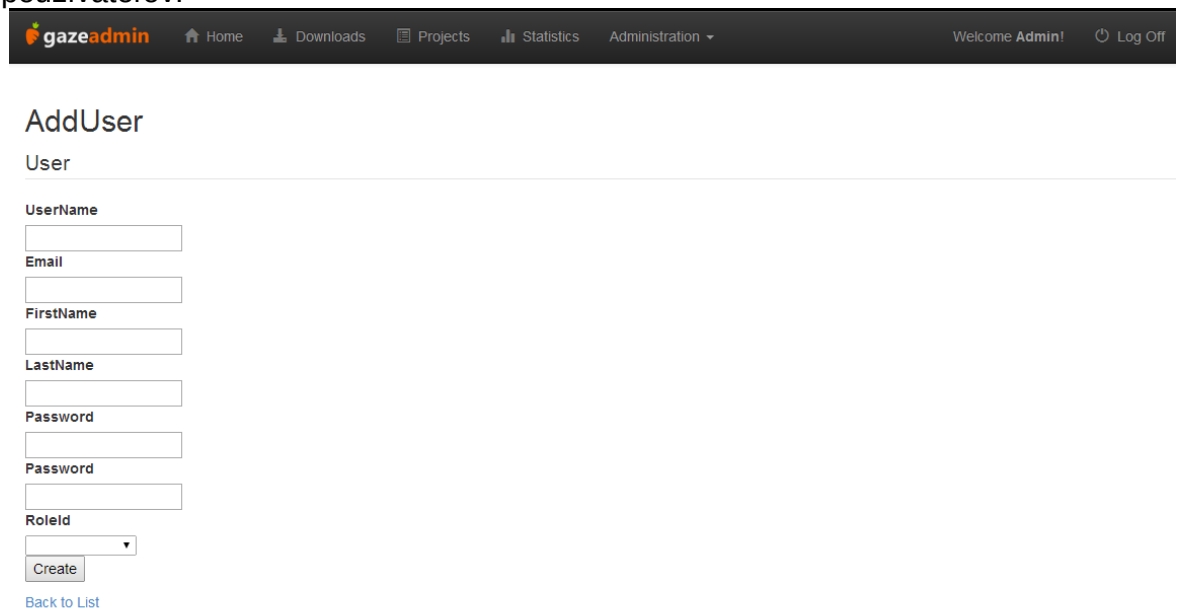
Definovanie oblastí záujmu (areas of interest) k sedeniu
Definovanie a zobrazovanie oblastí záujmu k sedeniu prebieha v časti Edit session v záložke Areas of interest. Na definovanie oblastí je potrebné mať nainštalovaný Addon pre ľubovoľný prehliadač. Samotné definonaie potom prebieha pomocou neho (podrobnejší popis - v príručke pre Addon).

F.3.2 Administrácia používateľov

Gaze Admin aplikácia umožňuje vytvárať používateľské účty a skupiny či importovať používateľov z CSV. Administrácia používateľov sa nachádza v hornom menu v položke Administration.
Vytváranie používateľkých účtov
Používateľský účet je možné vytvoriť v časti Administration > Users kde je zobrazený zoznam používateľov a nad nim sa nachádza odkaz pre vytvorenie používateľov. Všetky položky na obrazovke pre vytváranie používateľov je treba vyplniť, vybrať rolu a následne odoslať.
Používateľ môže mať nasledujúce roly:

- User
- Project admin
- Researcher
- Admin

Používateľ s právami User nemá v gaze admine žiadne práva a môže byť priradený len k sedeniu. Project admin vidí projekty a sedenia. Admin má prístup ku všetkým častiam aplikácie. Researcher môže prezerať projekty, ku ktorým je priradený ale nemôže ich upravovať. Akcie Delete a Edit sa nachádzajú pri každom zázname v tabuľke používateľov.

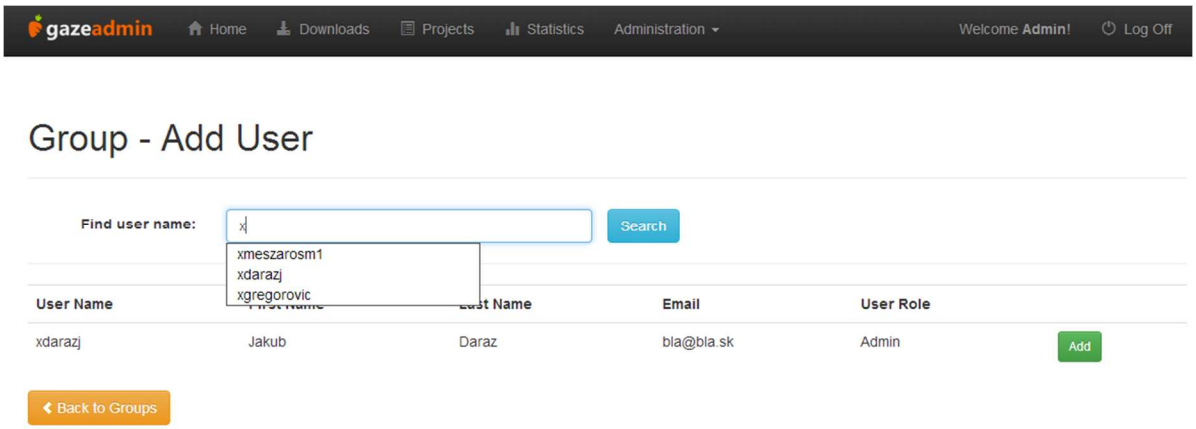


.

Obrázok 7 - Pridávanie používateľov do aplikácie

Vytváranie skupín používateľov

Pre vytvorenie skupiny požívateľov je potrebné ísť do časti Application > User Groups, kde sa nachádza zoznam skupín. Nad zoznamom je odkaz, ktorý zobrazuje formulár pre vytvorenie skupiny. Pre vytvorenie skupiny je potrebné zadať hodnoty len do názvu. Description netreba vypĺňať. Po vytvorení skupiny je možné v tabke nad formulárom pri editovaní skupiny, pridať do skupiny viacero používateľov. Po kliku na tlačidlo + Add users sa zobrazí prvok Autocomplete, kde stačí zadať časť alebo celé používateľské meno, ak zadáte časť tak program automaticky odporučí zvyšok mien, ktoré sa podobajú.



Obrázok 8 - Rozhranie pridávania používateľov

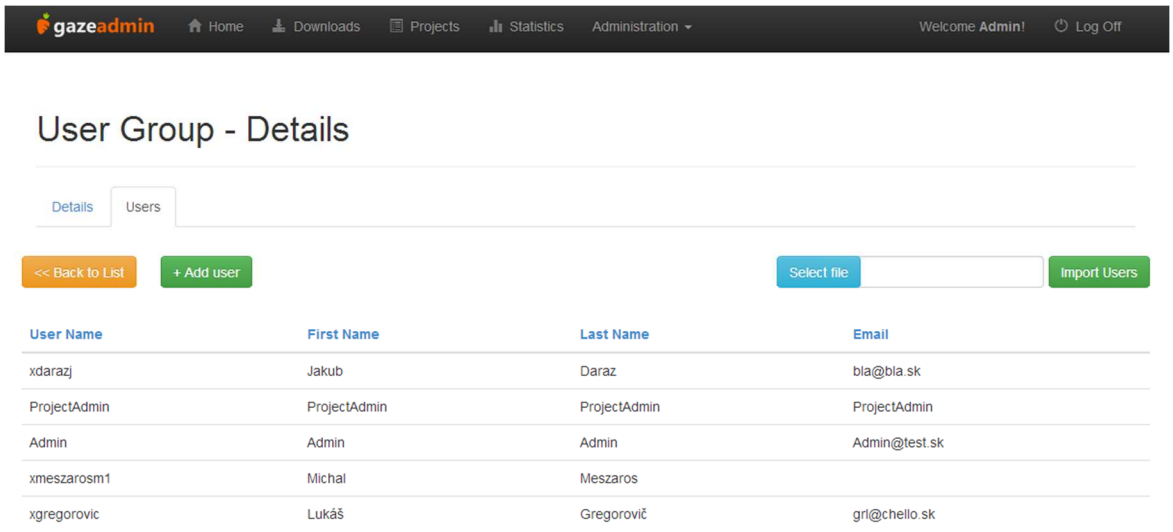
Po vybratí odporučeného používateľa stačí kliknúť na search a v tabuľke sa zobrazí tento používateľ. Po kliku na tlačidlo Add, ktoré sa nachádza v tabuľke bude používateľ pridaný.

Vytváranie aplikácií a priraďovanie API kľúčov

Vytváranie aplikačných účtov sa nachádza v časti Administration > Application user kde je zoznam vytvorených aplikácií. Princíp je rovnaký ako pri vytváraní používateľov, pričom Gaze Admin aplikácia po vyplnení všetkých položiek formulára priradí aplikácii unikátny API kľúč, ktorý je možné si pozrieť v zozname aplikácii(Application user)

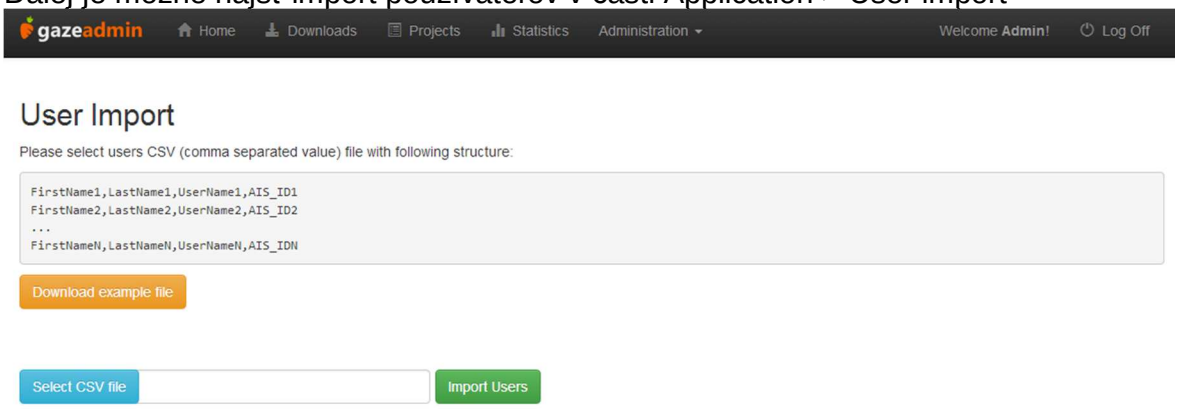
Import používateľov

Import používateľov je možné nájsť na dvoch miestach. V časti Application > User Group kde sa nachádza zoznam skupín pri každom zázname skupiny je možné kliknúť na ikonu detail. Po kliku na túto ikonu sa zobrazí zoznam priradených používateľov a nad zoznamom je možné importovať používateľov, ktorý sa rovno priradia ku skupine. Importovaný súbor musí byť vo formáte CSV.



Obrázok 9 - Zoznam používateľov priradených k skupine

Ďalej je možné nájsť import používateľov v časti Application > User import



Obrázok 10 - Rozhranie pre import používateľov

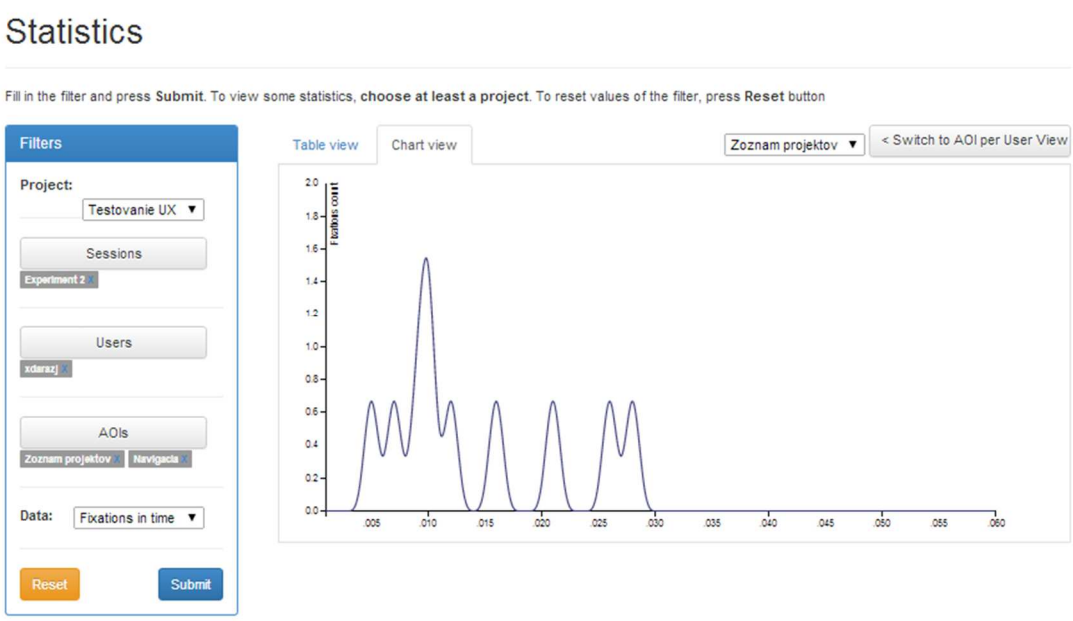
Rozhranie pre import používateľov, ktoré je uvedené na obrázku 9 znázorňuje v prvej časti príklad CSV súborov, ktorý treba nahrať a je možné si stiahnuť príklad CSV. Po nahratí sa zobrazí prehľad nahratých používateľov a tí ktorí sa nenahrali budú v tomto zozname zvýraznení.

F.3.3 Rozhranie štatistík

Gaze Admin umožňuje aj prezeranie dát z ukončených sedení pomocou rozhrania štatistík (položka *Statistics* v hornom menu). Na zobrazenie štatistík je potrebné: vyplniť filter > potvrdiť tlačidlom *Submit*. Cez filter je možné vybrať:

- projekt
- sedenia
- areas of interest
- konkrétnych participantov
- požadovaný typ štatistiky - momentálne poskytujeme zobrazenie troch typov:
    - počet fixacií
    - dĺžka fixácií
    - počet fixácií v čase

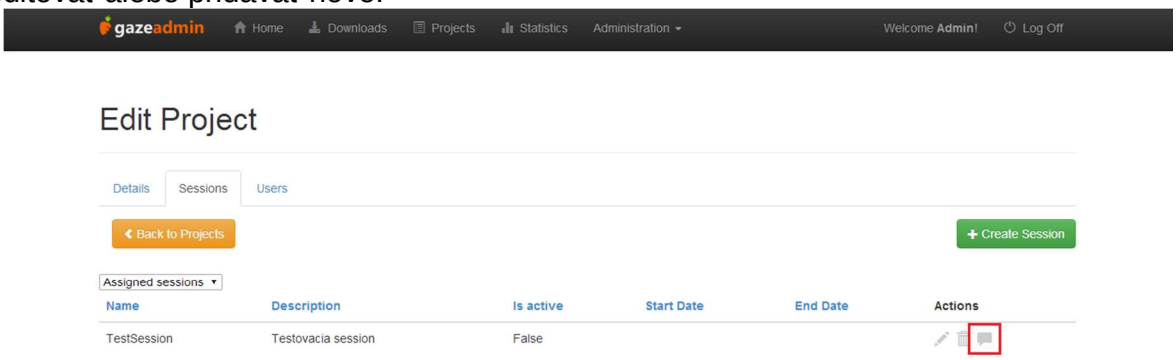
Po odoslaní formulára sa zobrazia vyfiltrované údaje zobrazia v tabuľke. Prepnutím záložky na *Chart view* je možné vidieť grafové zobrazenie.



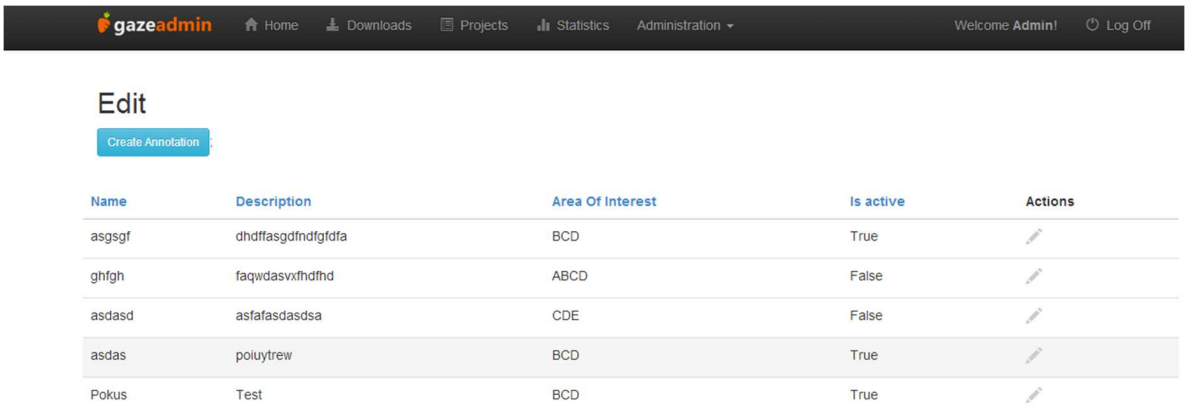
Obrázok 11 - Rozhranie štatistík

F.3.4 Rozhranie pre automatické anotácie

Gaze Admin podporuje aj vytváranie automatických anotácií definovanými Projektovým administrátorom. Aby prístup k definovaniu anotácie bol čo najjednoduchší k rozhraniu na spravovaniu anotácií sa dostaneme po vybratí príslušného projektu a následne v záložke Sessions vyberieme možnosť Annotation v stĺpci Actions (viď obrázok č. 12). V prípade, že používateľ nemá ProjectAdmin práva ale len Researcher práva dostane sa po kliknutí na túto ikonu len do rozhrania kde sú anotácie zobrazené a nemá právo ich editovať alebo pridávať nové.

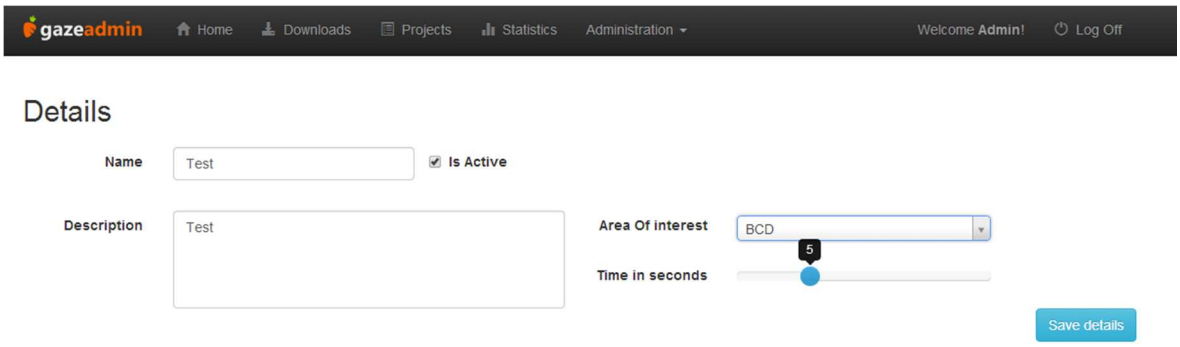


Obrázok 12: Prístup k rozhraniu pre anotácie



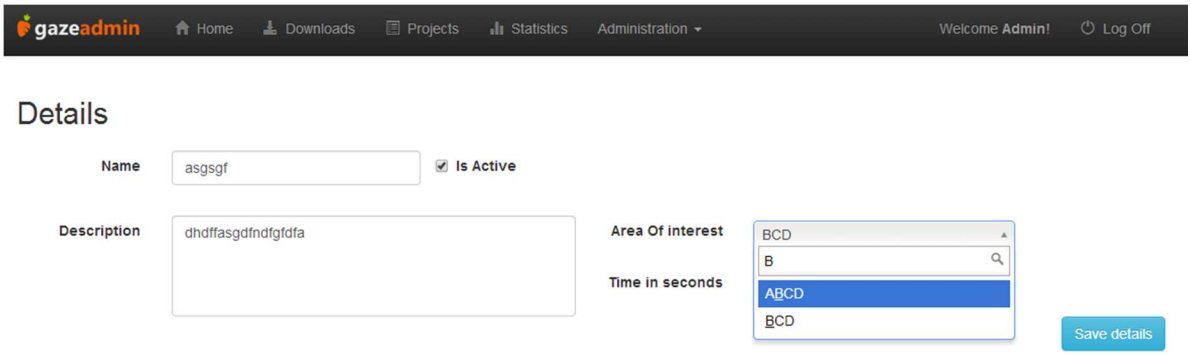
Obrázok 13: Rozhranie pre editáciu anotácií respektíve ich pridávanie

Nakoľko anotácie sú viazané na oblasti záujmu pri jej vytváraní je potrebné aby používateľ vybral oblasť ku ktorej sa majú anotácie vytvárať. Taktiež je potrebné určiť čas po ktorom sa vytvorí anotácia. Defaultná hodnota je určená 1 sekunda pričom granularita pre čas je 1 sekunda.



Obrázok 14: Rozhranie pre vytváranie anotácie

Pre uľahčenie párovania oblasti záujmu na anotáciu je v pri v vyberaní oblasti možné vyhľadávať ich pomocou fultextového vyhľadávania viď obrázok číslo 15.



Obrázok 15: Možnosť fulltextového vyhľadávania pri vyberaní oblasti záujmu