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

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.

Tento dokument sa venuje 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 usability experimentov

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 Opis prototypu

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

3.1 Architektúra

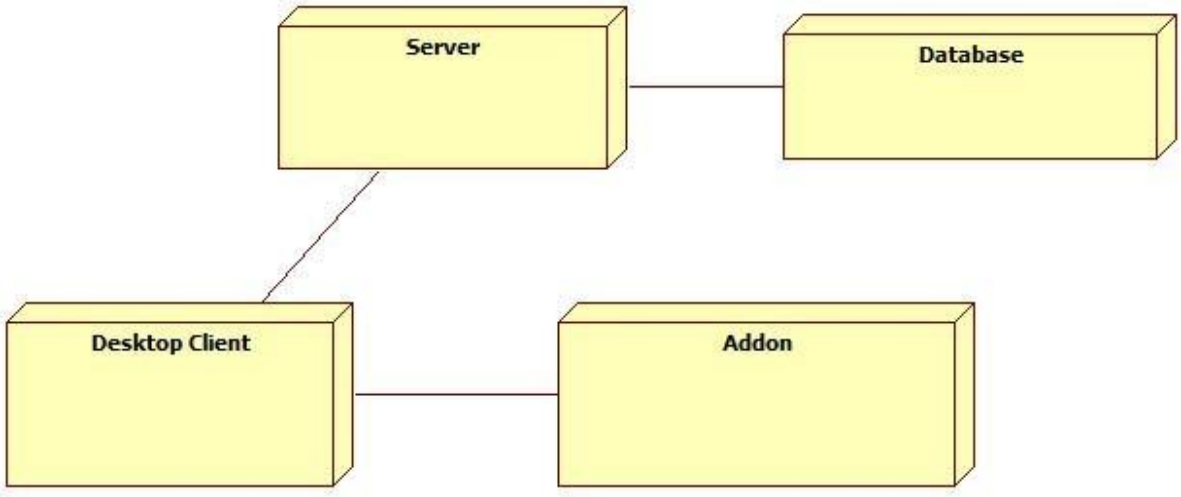
Architektúra nášho prototypu je momentálne rozložená do troch vrstiev.

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.

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

Tretia vsrtva - 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.



Obrázok 1 - Deployment diagram architektúry prototypu

3.2 Funkcionálne požiadavky

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

3.2.1 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 pohladu.

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

3.2.2 Webová aplikácia

Do webovej aplikácie sa možé prihlásiť každý používateľ systému. Roly používateľov však určujú práva a možnosti práce s webovou aplikáciou. "Bežný používateľ" má možnosť si, po prihlasení, *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 vrámci webovej aplikácie.

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

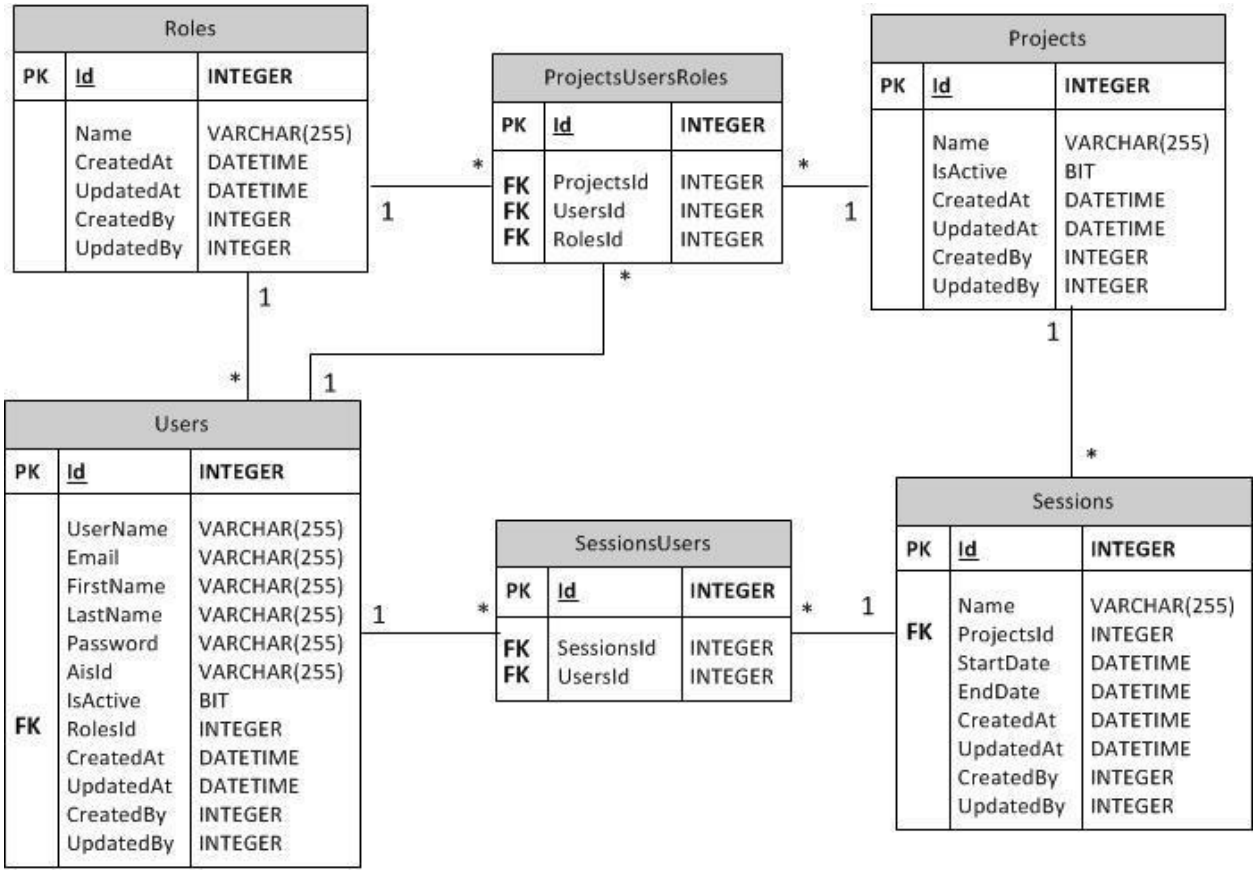
3.2.3 Addon

S addonom pracuje priamo vedúci experimentu a nepriamo "bežní používatelia". Vedúcemu experimentu addon poskytuje možnosť *nastavovať oblasti záujmu* a ich *dolezitost* v rámci webovej stránky, ktorá je cieľom testovania.

"Bežní používatelia" s addonom 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 addon dokáže *namapovať* elementy webovej stránky, ktorá je predmetom experimentu.

3.3 Dátový model

Nasledujúca schéma znázorňuje dátový model nášho prototypu.



Obrázok 2 - Dátový model prototypu

Hlavnou entitou v nej je používateľ (Users), so svojou globálnou rolou. Ten si po prihlásení si 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 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.

4.1 Identifikácia elementu na stránke

4.1.1 Úloha

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

4.1.2 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.

4.1.3 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 + "]";
};
```

4.1.4 Testovanie

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

4.2 Vytvorenie Add-onu pre browser

4.2.1 Úloha

Vytvorenie základného add-onu do prehliadača.

4.2.2 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.

4.3 Kalibrácia zariadenia

4.3.1 User story

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

4.3.2 Úloha

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

4.3.3 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.

4.3.4 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.

4.3.5 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.

4.4 Autentifikácia a autorizácia
4.4.1 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.

4.4.2 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.

4.4.3 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.

4.5 Analýza typov anotácii

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

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

4.5.3 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

4.6 Komunikácia pluginu s desktopovou aplikáciou

4.6.1 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.

4.6.2 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.

4.6.3 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

4.7 Ďalšie úlohy na ktorých sa pracovalo počas šprintu

4.7.1 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.

4.7.2 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

4.7.3 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

5 Beatles - 2. šprint

5.1 Identifikácia elementov na základe súradnice

5.1.1 Úloha

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

5.1.2 Analýza

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

5.1.3 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.

5.2 Používateľ sa chce prihlásiť (autentifikovať)

5.2.1 User story

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

5.2.2 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.

5.2.3 Analýza

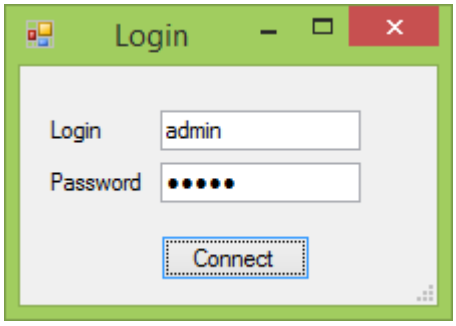
Počas stretnutia tímu bol vytvorený papierový návrh nového používateľského rozhrania. Toto nové používateľské rozhranie zahŕňa formulár pre prihlásenie, formulár pre výber zariadenia, formulár pre kalibráciu a formulár pre zobrazenie stavu zariadenia pre sledovanie pohľadu.

5.2.4 Implementácia

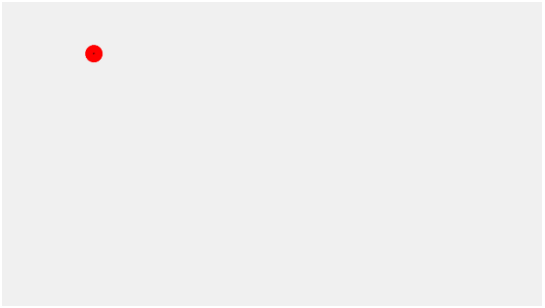
Konkrétne formuláre boli implementované pomocou Windows Forms. Rozhodli sme sa znovupoužiť niektoré komponenty z predchádzajúcej verzie programu. Patrí medzi ne

napr. čierne okno, v ktorom je vidno pozíciu očí používateľa, alebo panely zobrazujúce výsledok kalibrácie.

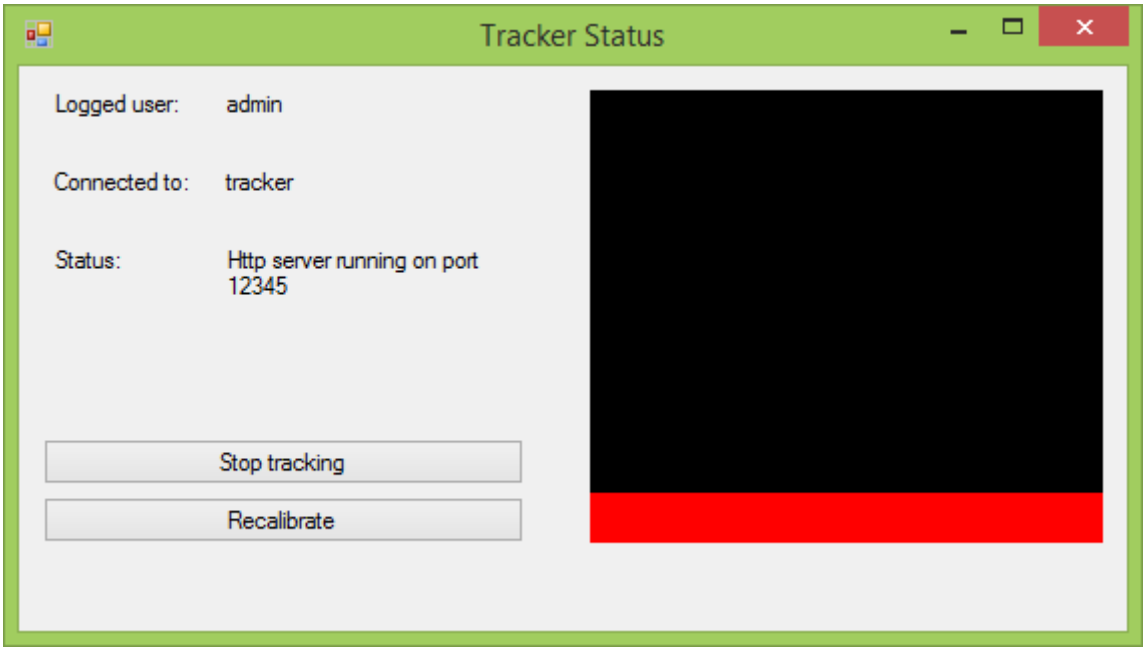
Používateľské rozhranie vyzerá nasledovne:



Obrázok 3 - Formulár pre prihlásenie používateľa



Obrázok 4 - Kalibrácia zariadenia



Obrázok 5 - Formulár zobrazujúci stav programu

5.2.5 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.

5.3 Prepočítavanie súradníc vzhľadom na aktívne okno
5.3.1 User story
Výskumník chce mať prepočítané súradnice zo zariadenia na aktuálne aktívne okno, aby vedel, kam sa presne na okno používateľ pozerá.

5.3.2 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.

5.3.3 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.

5.3.4 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).

5.3.5 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).

5.4 Experimentátor chce vidieť možné oblasti dynamicky pri hoveri

5.4.1 User story

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

5.4.2 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.

5.4.3 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;
}
```

5.4.4 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();
  }
};
```

5.4.5 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.

5.5 Simulácia Eye-trackera z pohybu myši

5.5.1 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.

5.5.2 Úloha

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

5.5.3 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.

5.5.4 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.

5.5.5 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.

5.5.6 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.

5.6 Prihlásenie ako REST
5.6.1 Úloha
Vytvoriť API ktoré bude môcť jednoducho a rýchlo komunikovať s inými aplikáciami

5.6.2 Analýza
Vybrať vhodnú technológiu pre servis ako je rest alebo sockety atd.

5.6.3 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.

5.7 Vytvorenie nového používateľa
5.7.1 User story
Zákazník si želá prostredie pre jednoduché pridávanie používateľov s prehľadnou validáciou zadaných hodnôt.

5.7.2 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.

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

5.8 Zadefinovanie rolí
5.8.1 User story
Požiadavka bola vytvoriť používateľa ktorý bude môcť pristupovať a spravovať všetky moduly webovej aplikácie.

5.8.2 Analýza
Pridávať pomocou SQL kódu do databáze

5.8.3 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.

5.9 Zadefinovanie formy protokolu pre odosielanie dát na server

5.9.1 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.

5.9.2 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.

[1] Odkaz na špecifikáciu JSON - http://tools.ietf.org/html/rfc4627

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

5.10 Odosielanie dát na server

5.10.1 Úloha

Odosielanie zozbieraných dát na server pomocou REST architektúry.

5.10.2 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);
```

5.10.3 Testovanie
Bolo potrebné aby serializácia a deserializácie 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.

5.11 Vytvorenie projektu
5.11.1 User story
Výskumník chce vytvoriť projekt, v rámci ktorého bude vykonávať experimenty.

5.11.2 Úloha
Umožniť v rámci webovej aplikácie vytváranie projektov.

5.11.3 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

5.11.4 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.

5.11.5 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.

5.12 Vytvorenie sedenia v rámci projektu

5.12.1 User story

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

5.12.2 Úloha

Umožniť vo webovej aplikácií výskumníkovi vytvoriť konkrétny experiment – sedenie.

5.12.3 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

5.12.4 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.

5.12.5 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.

5.13 Ďalšie úlohy na ktorých sa pracovalo počas šprintu

5.13.1 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.

5.13.2 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.

5.13.3 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.

5.13.4 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.

6 Coldplay - 3. šprint

6.1 Prihlásenie používateľa cez LDAP

6.1.1 Úloha

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

6.1.2 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.

6.1.3 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.

6.2 Podpora pre usporiadanie, filtrovanie a paginácu dát v tabuľke
6.2.1 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.

6.2.2 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.

6.2.3 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" }))

  ));

}
```

6.3 Komunikácia add-onu s klientom na desktope
6.3.1 User story
Výskumník chce vedieť, kam sa presne v okne pozerá používateľ, aby vedel zistiť, či sa pozerá na zadefinovaný area of interest.

6.3.2 Úloha
Zabezpečiť komunikáciu medzi desktopovým klientom a browser add-onom pre prepočítavanie súradníc.

6.3.3 Analýza
Na stretnutí tímu sme rozmýšlali 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.

6.3.4 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.

6.3.5 Testovanie
Server bol testovaný posielaním GET a POST requestov pomocou Google Chrome aplikácie „Postman – REST Client“. Server úspešne dokázala prijímať a posielať súradnice pohľadu, posielané ako JSON.

6.4 Adaptér pre klienta
6.4.1 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.

6.4.2 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.

6.4.3 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.

6.4.4 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.

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

6.4.6 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*.

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

6.5.3 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.

6.5.4 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.

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

6.5.6 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.

6.6 Manažment používateľov CRUD
6.6.1 User story
Požiadavka na prostredie pre manipuláciu s používateľmi a s prehľadom zapísaných používateľov

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

6.6.3 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.

6.7 Import používateľov z CSV súboru do databázy
6.7.1 Úloha
Vytvoriť metódu pre importovanie používateľov s CSV súboru cez webové rozhranie.

6.7.2 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");
}
```

6.8 Pridanie používateľa k projektu

6.8.1 User story

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

6.8.2 Úloha

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

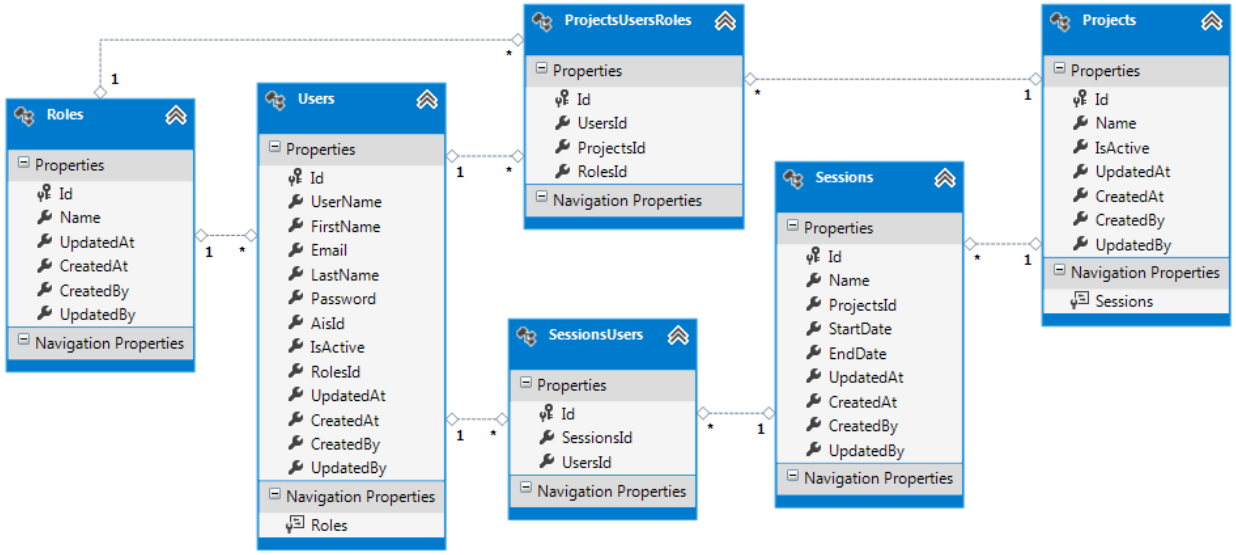
6.8.3 Analýza

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

6.8.4 Implementácia

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



Obrázok 6 - čiastočný fyzický model databázy

K 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.

Po ukončení pridávania používateľov je nutné, aby výskumník uložil projekt a tým odoslal nové údaje do databázy.

6.8.5 Testovanie

Testovanie bolo zamerané na to, aby používateľ, ktorý nemá dostatočné práva pridávať iných, k tejto funkcionalite nemal prístup. Pridávať nového používateľa môže iba projectAdmin. Podobne, ako pri pridávaní projektu a sedenia bolo tiež potrebné testovať správnosť údajov, ich formát a komunikáciu s databázou.

PRÍLOHA A - Výstupy analýz

A.1 ACDC – 1. šprint

A.1.1 Formát dát zo zariadenia

Čo obsahuje packet získaný zo zariadenia:

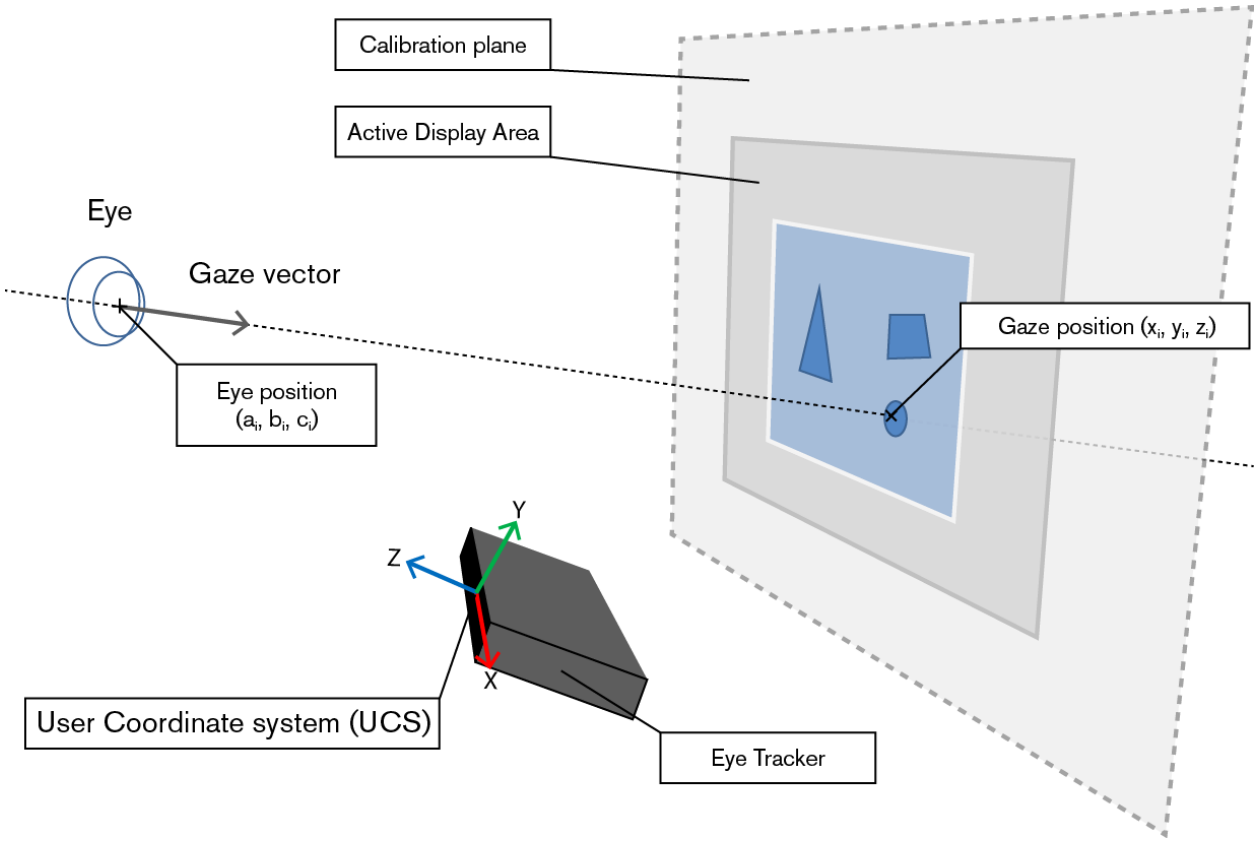
- Timestamp
- Polohu očí (UCS)
- Relatívnu polohu očí
- 3D bod pohľadu
- 2D bod pohľadu
- Validačný kód
- Priemer zreničky

Timestamp:

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

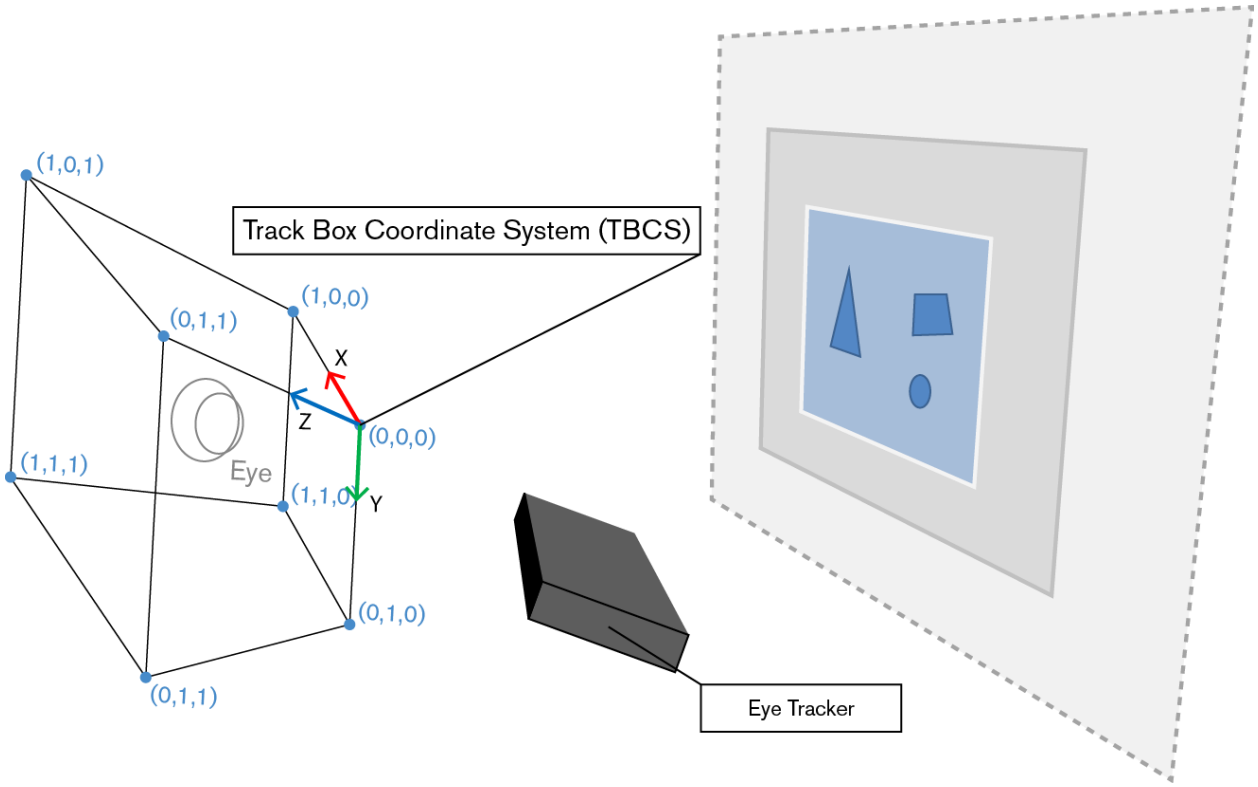
Polohu očí:

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



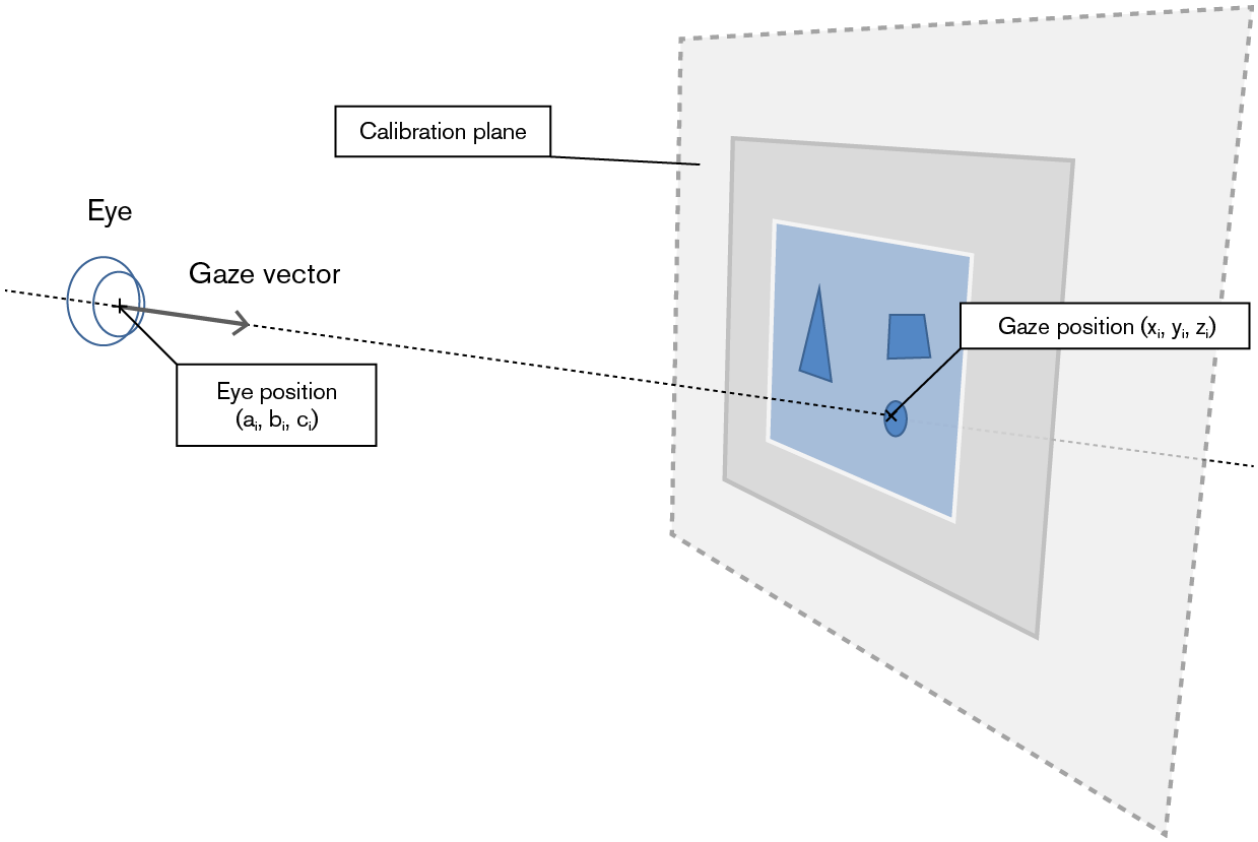
Relatívna poloha očí:

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.



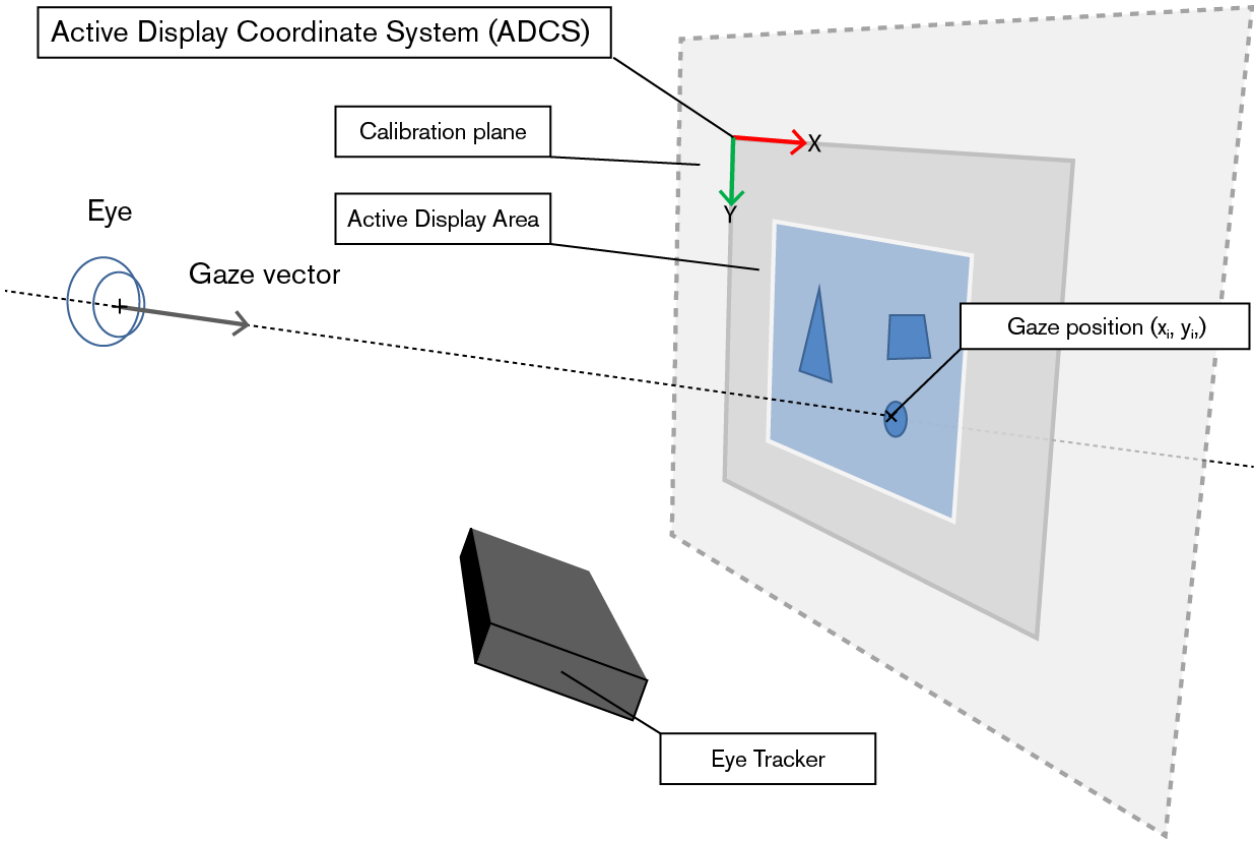
3D bod pohľadu:

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

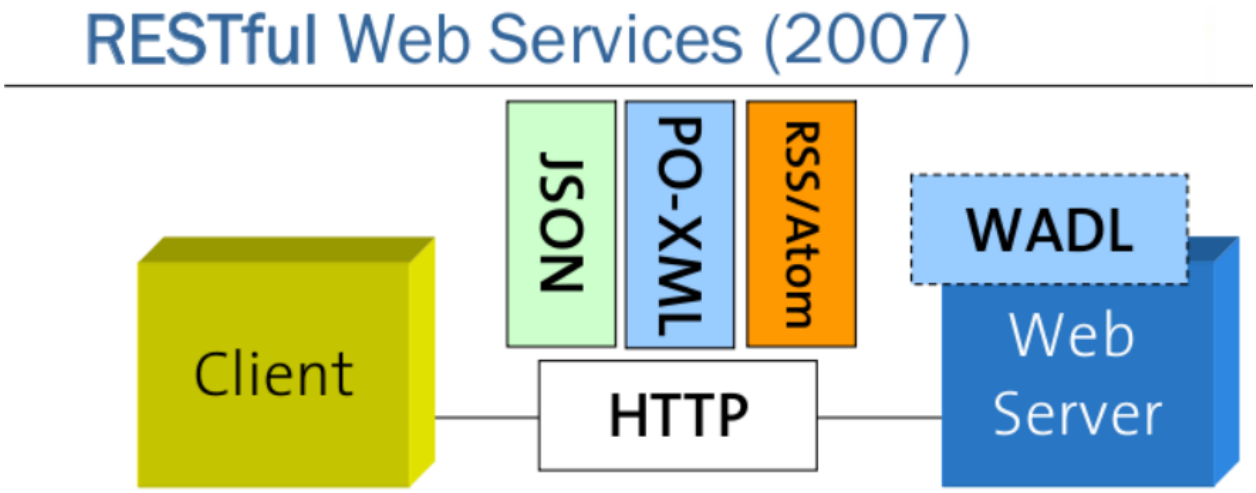
Priemer zreničiek:

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

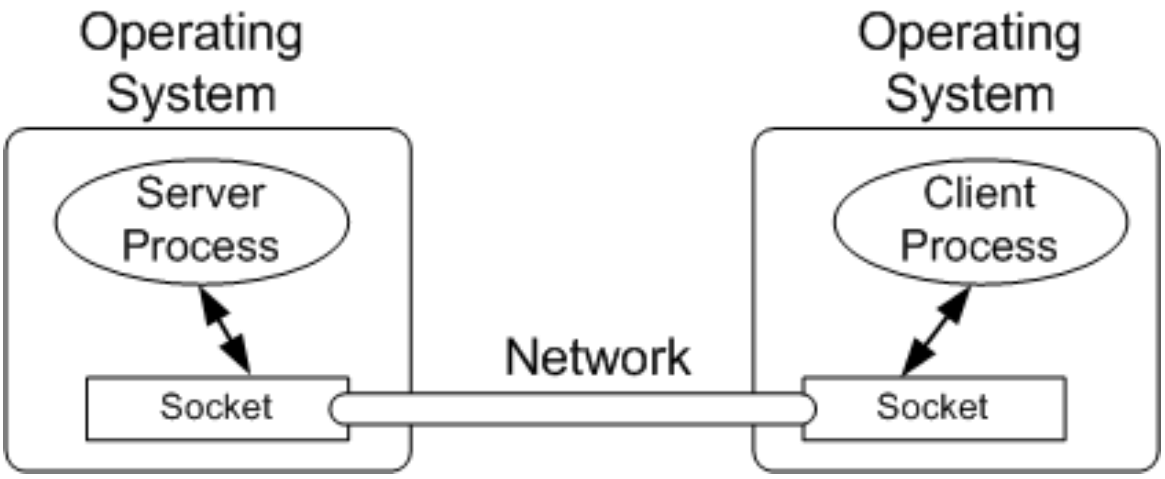
výhody:

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
}
```