

**Testovanie používateľského zážitku pomocou
sledovania pohľadu**

Dokumentácia k dielu

Vedúci tímu: Ing. Róbert Móro

Členovia tímu: Bc. Peter Kiš, Bc. Matej Kucek, Bc. Viktória Lovasová,
Bc. Peter Kušnír, Bc. Marek Šulek, Bc. Šimon Valíček, Bc. Martin Svrček

Akademický rok: 2014/2015

Obsah

1	Úvod.....	1
2	Globálne ciele.....	1
2.1	Prepojenie cez API	1
2.2	Štatistiky a vizualizácie	2
3	Celkový pohľad na systém	4
4	Moduly systému	6
4.1	Modul API.....	6
4.2	Knižnica na komunikáciu systému Gaze-tracking s externými aplikáciami	11
4.2.1	Analýza problematiky	11
4.2.2	Návrh riešenia	11
4.2.3	Implementácia základnej komunikácie	12
4.2.4	Testovanie v jazyku javascript	13
4.3	Agregácie.....	13
4.3.1	Implementácia	13

1 Úvod

Použitelnosť softvéru je pomerne rozšírenou oblasťou, pri ktorej sa využívajú rôzne techniky na jej overenie. V súčasnosti je jednou z najzaujímavejších sledovanie pohľadu pri práci s daným softvérom alebo aplikáciou. V kontexte sledovania pohľadu existuje priestor pre veľké uplatnenie tejto metódy aj v bežnom živote. Napríklad pri ovládaní elektronických zariadení pohľadom.

My sa v našom projekte chceme zamerať na tieto základné oblasti:

- *Analýza použiteľnosti softvéru* – V tomto kontexte chceme využiť eye-tracker na sledovanie pohľadu používateľa pri použití určitého softvéru a následným vyhodnocovaním získaných dát. Tento scenár sa dá využiť v oblasti UX firiem zaoberajúcich sa použiteľnosťou alebo na výskumné účely ako časť overenia pre rôzne články a iné projekty.
- *Prispôsobovanie obsahu na webe* – Táto časť súvisí s prepojením nástroja eye-tracker s určitými aplikáciami (aj webovými) a následným reagovaním na jednotlivé udalosti zachytené sledovaním pohľadu. Klasickým príkladom prispôsobovania môže byť generovanie odporúčaní.
- *Identifikácia problematických miest v danej doméne* – Klasickým využitím je v tomto smere identifikácia nezrozumiteľných častí textu v rôznych vzdelávacích systémoch a podobne.

V rámci tejto kapitoly spíšeme základné ciele, ktoré budú určovať naše smerovanie počas práce na projekte. A následne opíšeme pohľad na architektúru systému obohatenú o naše nové moduly.

2 Globálne ciele

V rámci práce na tomto projekte sme si stanovili dva základné ciele, ktoré chceme dosiahnuť. Obidva ciele vzišli z pomerne podrobnej analýzy. Snažili sme si predstaviť, čo by bolo potrebné pre klasického používateľa nášho systému. Avšak neostali sme len pri tom, a preto sme si dohodli stretnutie s „našimi zákazníkmi“, ktorými sú profesori a študenti používajúci sledovače pohľadu pri overovaní ich projektov.

2.1 Prepojenie cez API

Prvou a základnou úlohou alebo cieľom je prepojenie existujúceho systému s ľubovoľnými externými aplikáciami cez API, ktoré chceme vytvoriť.

- **Externá aplikácia**
 - V rámci tejto časti sa konkrétne budeme snažiť vytvoriť API na prepojenie s výučbovým systémom ALEF, ktorý sa využíva na našej fakulte. Avšak celkové riešenie bude všeobecne použiteľné na rôzne typy aplikácií.
 - Tento systém sme si vybrali hlavne preto, že je reálne využívaný na našej fakulte na viacerých kurzoch alebo predmetoch. Takisto je tu možnosť, že by sa mohli sledovače pohľadu využívať priamo na cvičeniach pri práci so systémom ALEF.
- **Čo bude potrebné spraviť ?**

- V rámci tohto cieľu je potrebné mať možnosť vytvorenia projektov, sedení a pridávania používateľov do systému.
- Následne sa budú počas používania systému ALEF dopredu prepočítavať agregácie anotovaných dát, ktoré budú pri používaní systému vytvárané (napríklad čas, počet pozretí v čase pre danú oblasť).
- Na základe zozbieraných dát a informácií sa budú následne vykonávať rôzne prispôsobenia priamo v systéme ALEF, alebo sa budú generovať upozornenia. Využitie teda vidíme hlavne v zlepšení práce so systémom ako celkom a s tým súvisiacim zlepšením učenia sa.
- **Prenosnosť systému (portability)**
 - Pri vývoji API však chceme dbať na to, aby bolo čo najuniverzálnejšie v kontexte jeho prenositeľnosti na iné aplikácie ako je ALEF. Toto je dôležitý bod, pretože bude mať veľký vplyv na vývoj systému.
 - Celá biznis myšlienka za týmto spočíva v tom, že ak niekto bude chcieť používať náš systém pre jeho vlastnú aplikáciu, budeme schopný poskytnúť mu naše API bez potreby vykonávania nejakých veľkých úprav.

Ciele pre zimný semester v kontexte prepojenia cez API:

- Vytvorenie základnej infraštruktúry
 - Našou snahou v zimnom semestri je v rámci API umožnenie vytvárania projektov, sedení, používateľov oblastí záujmu a agregácií.
 - V kontexte agregácií chceme umožniť samotné počítanie agregácií nazískanými dátami.
- Vytvorenie základnej javascript knižnice
 - Takisto chceme aby bola na konci zimného semestra možná základná komunikácia medzi externou aplikáciou a gaze-tracking softvérom.

2.2 Štatistiky a vizualizácie

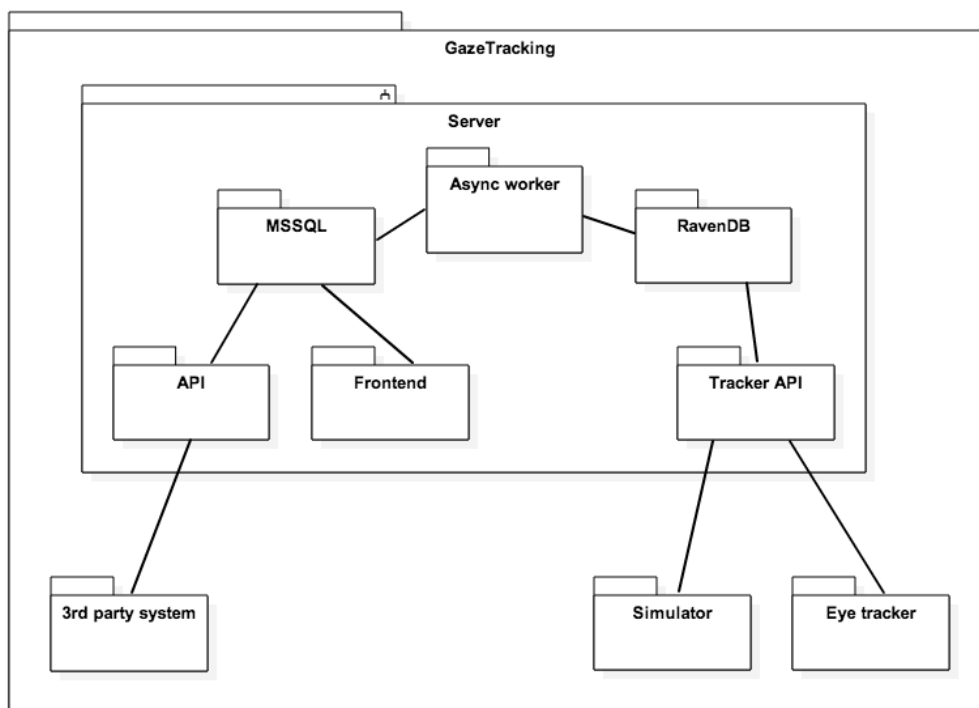
V rámci minuloročného projektu sa podarilo vytvoriť základnú infraštruktúru na sledovanie pohľadu a ukladanie dát z takéhoto pozorovania. Našou úlohou bude využiť získané dáta a nejakým spôsobom ich spracovať a vizualizovať a hlavne ich týmto spôsob spraviť užitočné.

- **Význam spracovania dát**
 - Prvým významom takto spracovaných dát je poskytnutie rozumného pohľadu na dáta aj pre ľudí, ktorý nie sú výskumníci a chceli by vidieť nejaký výstup zo sledovania pohľadu.
 - Druhým využitím je umožnenie získania zaujímavých vizualizácií, ktoré si môžu výskumníci vložiť do práce a zdokumentovať tak overenie práce.
 - Ďalším využitím je pomoc učiteľom pri tvorbe učebných textov, alebo pri získaní informácií o tom, čo je pre používateľov problémová oblasť, a čo je potrebné rozobrať bližšie.
- **Spôsob spracovania dát**
 - Štatistiky
 - Štatistiky predstavujú základný prehľad o projektoch, sedeniach, používateľoch a ich aktivite v systéme, ktorá bola zaznamenaná sledovaním pohľadu.

- Zaznamenávali by sa takto napríklad počty fixácií pohľadu alebo trvanie v čase pre danú oblasť záujmu.
- Vizualizácie
 - Vizualizácie predstavujú rôzne grafy a teplotnémapy zaznamenávajúce poradie fixácií, ale aj pomery časov sledovania jednotlivých oblastí.
 - V tomto kontexte bude potrebné zaznamenávanie zmeny pohľadu v čase (pomocou videa alebo obrázkov).

Ciele pre zimný semester v kontexte vizualizácie nemáme stanovené.

3 Celkový pohľad na systém



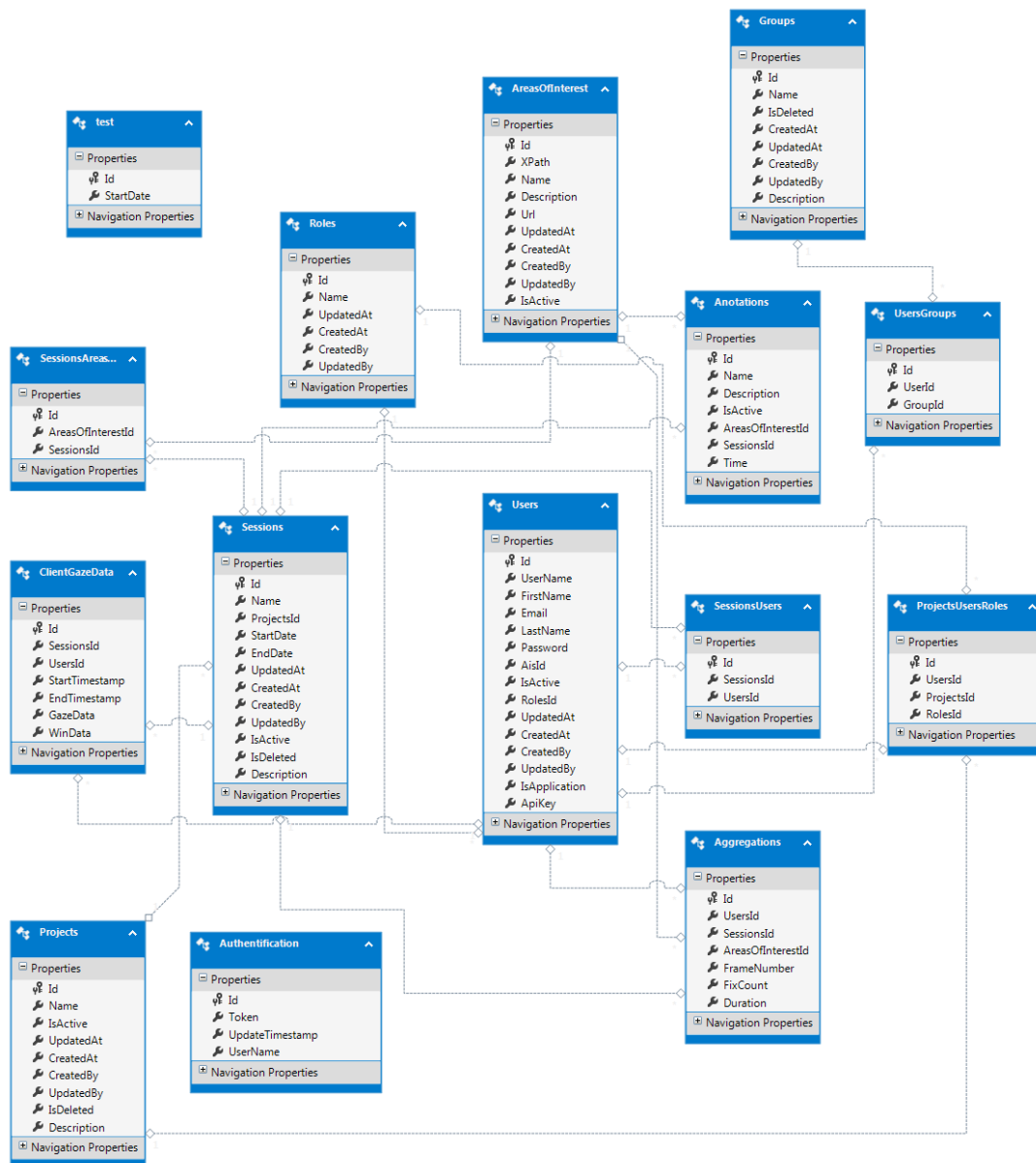
Obrázok 1 – Základná architektúra systému gaze-tracking

Služba je navrhnutá ako server-klient riešenie. Z pohľadu toku dát je jedným z hlavných komponentov celého systému NOSQL databáza RavenDB. Do tejto sa ukladajú jednotlivé súradnice pohľadu sledovaného používateľa. Zdrojom údajov je senzor na sledovanie pohľadu, z ktorého sú dáta čítané prostredníctvom desktopovej klientskej aplikácie. Táto aplikácia následne komunikuje s rozšírením vo webovom prehliadači, ktorý jej poskytuje informácie o elementoch DOM stromu webovej stránky, ktoré sa nachádzajú na aktuálnych súradniciach pohľadu. Klientska aplikácia následne súradnicové dáta doplnené o identifikáciu elementov odosiela prostredníctvom HTTP socketu na server. Na serveri je za týmto účelom implementované proprietárne API rozhranie slúžiace na komunikáciu s klientskou aplikáciou, z ktorého sa prijaté dáta ukladajú do RavenDB. Okrem RavenDB riešenie zahŕňa navyše relačnú databázu, do ktorej sa ukladajú agregované dáta, vypočítané za pomoci asynchrónnych workerov z dát uložených v RavenDB. Tento prístup bol zvolený za cieľom zvýšenia priepustnosti systému pri vyberaní štatistických dát o sedeniach z databázy. V relačnej databáze sa takisto nachádzajú administratívne údaje ohľadom projektov sledovania pohľadu, konkrétnych sedení a sledovaných používateľov. Pre účely manažmentu administratívnych údajov slúži frontend aplikácia.

Novými komponentmi v systéme sú API modul pre komunikáciu s aplikáciami tretích strán. Demonstratívne bude implementovaná komunikácia medzi službou a aplikáciou Alef. Okrem toho bude frontend aplikácia rozšírená o vizualizáciu štatistických dát agregovaných v relačnej databáze.

Dátový model

Centrálnymi entitami dátového modelu sú projekt, sedenie a používateľ. Používateľ je základným prvkom pri prístupe k systému. Zastupuje aplikácie tretích strán a takisto jednotlivých používateľov vstupujúcich do procesu sledovania pohľadu. V prípade aplikácií tretích strán je autentifikácia vykonávaná na základe poskytnutého API kľúča. Takýto používateľ môže následne vytvárať projekty, sedenia, a ďalších používateľov, ktorých pohľad bude v rámci jednotlivých sedení sledovaný. Entita projekt slúži na uchovávanie základných údajov o projektoch sledovania pohľadu. V rámci jedného projektu je spravovaných viacero nezávislých sedení, do ktorých sú pridávaní používatelia.



Obrázok2 – Dátový model systému GazeTracking

Klientská aplikácia

V rámci projektu z minulého roku sa podarilo vytvoriť desktopový klient, ktorý je prvou inštanciou, s ktorou sa používateľ dostáva do styku. Nachádza sa teda na prvej vrstve architektúry, najbližšie k používateľovi.

Jeho úlohou je komunikácia so zariadením na sledovanie pohľadu. Aby takáto komunikácia mohla byť naviazaná, je potrebná synchronizácia so zariadením, ktorú taktiež tento desktopový klient zabezpečuje.

V rámci komunikácie zaznamenáva, kde sa daný používateľ pozeral a následne tieto informácie vo forme dát odosiela na serverovú časť systému.

Rozšírenie do prehliadača

V rámci práce minuloročného tímu sa podarilo vytvoriť rozšírenia pre webové prehliadače GoogleChrome a MozillaFirefox. Samotné rozšírenie sa delí na administrátorskú a klientsku časť (rozšírenie pre Firefox obsahuje obe tieto časti v jednom rozšírení, zatiaľ čo Chrome rozšírenia sú oddelené). Hlavnou funkciou rozšírenia je obohacovanie dát klientskej aplikácie, ktoré získa zo zariadenia na sledovanie pohľadu. Pre každú súradnicu pohľadu je v rozšírení určený identifikátor HTML elementu, na ktorý sa používateľ pozerá, pričom sú tieto údaje spárované s dátami prijatými z klientskej aplikácie. Druhou funkciou rozšírenia je možnosť definovania oblastí záujmu pre jednotlivé stránky. Myšlienkou je špecifikovanie oblastí webovej stránky na základe identifikujúceho elementu, samotnej adresy stránky a doplnujúcimi informáciami ako sú názov a popis.

WCF (Windows Communication Foundation)

WCF predstavuje súbor služieb v rámci .NET, ktorý umožňuje vytváranie aplikácií orientovaných na služby. Minulý rok vytvoril predchádzajúci tím metódy pre správu dopytov na WCF službu v súbore *Viewtracking.wcf*. Táto služba má na starosti zachytávanie dopytov a následné odpovede, pričom komunikuje prostredníctvom protokolu HTTPS. Samotné odpovede alebo dáta v nich obsiahnuté sa posielajú vo formáte JSON.

Simulátor

Tím z minulého roka spravil simulátor, ktorý predstavuje náhradu za eyetracker pričom zaznamenáva súradnice myši a ukladá ich do databázy. Predpokladá sa, že pozícia myši predstavuje používateľov pohľad, pričom sa berie do úvahy aj určitý šum v okolí myši a nie jej presné súradnice.

4 Moduly systému

V tejto kapitole sú popísané jednotlivé moduly systému na sledovanie pohľadu vzhľadom na ciele, ktoré sme si stanovili. Každý z týchto modulov predstavuje oblasť, ktorú aktuálne riešime, alebo chceme riešiť v blízkej budúcnosti.

4.1 Modul API

Modul API slúži na komunikáciu aplikácii tretích strán so službou gaze-tracking. Prostredníctvom jednotlivých volaní API modulu je možné spravovať projekty, sedenia,

agregácií, oblastí záujmu a používateľov prístupujúcej aplikácie. Aplikácia sa musí pri každom prístupe k API modulu autentifikovať za pomoci API kľúča, ktorý je odosielaný v tvare reťazca. API modul je implementovaný za použitia technológie WCF. Prijíma parametre prostredníctvom HTTP POST požiadaviek a odpovedá vo forme JSON reťazcov. Špecifikácia jednotlivých API volaní je uvedená nižšie.

Volania GazeTracking API

- **Manažment Projektov**
 - **CreateProject (Vytvoriť projekt)**
 - **požiadavka:** POST application/x-www-form-urlencoded
 - **url:** /api/project/create
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - name:String(notnull) - názov vytváraného projektu
 - description:String - popis vytváraného projektu
 - **odpoveď:**
 - application/json
 - id:integer - identifikátor vytvoreného projektu
 - **HTTP stavové kódy:**
 - 201 - projekt bol vytvorený
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
 - **ListProjects (Vypísanie všetkých projektov)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/project/list
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - active:Boolean(notnull) - vypísať len aktívne projekty
 - **odpoveď:**
 - application/json
 - projects - pole nájdených projektov
 - id:integer - id projektu
 - name:String - názov projektu
 - isActive:boolean - stav projektu
 - **HTTP stavové kódy:**
 - 200 - všetko v poriadku
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
 - **GetProjectById (Vypísať podrobnosti projektu)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/project/getbyid
 - **parametre:**

- apiKey:String(notnull) - autentifikačný token
 - id:integer(notnull) - id hľadaného projektu
 - **odpoveď:**
 - application/json
 - id:integer - id projektu
 - name:String - názov projektu
 - isActive:boolean - stav projektu
 - description:String - popis projektu
 - createdAt:integer - čas vytvorenia projektu v unix formáte
 - updatedAt:integer - čas poslednej zmeny projektu v unix formáte
 - **HTTP stavové kódy:**
 - 200 - všetko v poriadku
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
- **Manažment sedení**
 - **CreateSession(Vytvoriť sedenie)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/session/create
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - projectId:integer(notnull) - id nadradeného projektu
 - **odpoveď:**
 - application/json
 - id:integer - id vytvoreného sedenia
 - **HTTP stavové kódy:**
 - 201 - sedenie bolo vytvorené
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
 - **ListSession(Vypísať existujúce sedenia)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/session/list
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - active:boolean(notnull) - vypísať len aktívne sedenia
 - **odpoveď:**
 - application/json
 - sessions - pole sedení
 - id - identifikátor sedenia
 - name - názov sedenia
 - isActive - stav sedenia
 - **HTTP stavové kódy:**

- 200 - všetko v poriadku
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
- **GetSessionById(Vypísať podrobnosti sedenia)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/session/getbyid
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - id:integer(notnull) - identifikátor sedenia
 - **odpoveď:**
 - application/json
 - id - identifikátor sedenia
 - projectId - id nadriadeného projektu
 - name - názov sedenia
 - isActive - stav sedenia
 - createdAt:integer - čas vytvorenia sedenia v unix formáte
 - updatedAt:integer - čas poslednej zmeny sedenia v unix formáte
 - **HTTP stavové kódy:**
 - 200 - všetko v poriadku
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
- **Manažment používateľov**
 - **CreateUser (Vytvoriť používateľa)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/user/create
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - username:String(notnull) - používateľské meno
 - firstname:String(notnull) - krstné meno používateľa
 - lastname:String(notnull) - priezvisko používateľa
 - email:String - email používateľa
 - **odpoveď:**
 - application/json
 - id:integer - id vytvoreného používateľa
 - **HTTP stavové kódy:**
 - 201 - používateľ bol vytvorený
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
 - **ListUsers (Vypísať existujúcich používateľov)**
 - **požiadavka:**POST application/x-www-form-urlencoded
 - **url:**/api/user/list

- **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - active:boolean(notnull) - vypísať len aktívnych používateľov
 - sessionId:integer - identifikátor sedenia
- **odpoveď:**
 - application/json
 - users - pole používateľov
 - id:integer - identifikátor používateľa
 - username:String - používateľské meno
 - isActive:Boolean - stav používateľa
- **HTTP stavové kódy:**
 - 200 - všetko v poriadku
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní
- **GetUserById (Vypísať podrobnosti používateľa)**
 - **požiadavka:POST** application/x-www-form-urlencoded
 - **url:**/api/user/getbyid
 - **parametre:**
 - apiKey:String(notnull) - autentifikačný token
 - id:integer(notnull) - identifikátor používateľa
 - **odpoveď:**
 - application/json
 - id:integer - identifikátor používateľa
 - username:String - používateľské meno
 - firstname:String - krstné meno používateľa
 - lastname:String - priezvisko používateľa
 - email:String - email používateľa
 - isActive:Boolean - stav používateľa
 - createdAt:integer - čas vytvorenia používateľa v unix formáte
 - updatedAt:integer - čas poslednej zmeny používateľa v unix formáte
 - **HTTP stavové kódy:**
 - 200 - všetko v poriadku
 - 400 - nesprávny formát požiadavky
 - 401 - nesprávny api kľúč
 - 500 - chyba pri spracovaní

Podobne sme definovali aj manažment pre oblasti záujmu – na zadefinovanie oblasti, ktorú chce používateľ vyhodnocovať.

4.2 Knížnica na komunikáciu systému Gaze-tracking s externými aplikáciami

Jedným z našich hlavných cieľov je vytvorenie API na prepojenie systému na sledovanie pohľadu (Gaze) s inými externými aplikáciami (v našom projekte konkrétne so systémom ALEF). V tomto kontexte je potrebné, aby ALEF vedel nejakým spôsobom komunikovať so systémom Gaze.

V prvom návrhu sme chceli, aby sa dáta posielali klientskej aplikácii, čo sa nám zdalo rozumnejšie ako tieto dáta poselať na server, keďže tu by mohol byť problém s veľkým množstvom posielených dát. Tento návrh vyšiel z prvotnej analýzy systému a jeho častí.

Následne sa však zorganizovalo stretnutie so zákazníkmi, na ktorom sme diskutovali aj túto tému. Práve na tomto stretnutí padol návrh, že dáta nemusíme vôbec nikdeposielať, ale stačí ak ich budeme spracovať priamo v klientskej aplikácii bez nutnosti posielania dát na server. Toto riešenie chceme implementovať prostredníctvom javascript funkcií, čím sa umožní okamžitá odozva systému a súčasne získame komponent, ktorý sa bude dať ľahko použiť aj v iných aplikáciách. Tieto aplikácie tak budú rozšírené o možnosť sledovania pohľadu. Táto myšlienka nás zaujala natoľko, že sme sa rozhodli ju využiť aj pri našom probléme.

4.2.1 Analýza problematiky

Využitie jazyka javascript pri našom probléme súviselo hlavne s počúvaním udalostí, ktoré by posielal systém Gaze pri rôznych udalostiach. V rámci tejto oblasti sme mali na výber viaceré možnosti ako implementovať podobnú komunikáciu aplikácie a nášho systému.

V súčasnosti výučbový systém ALEF využíva na komunikáciu prostredníctvom posielania udalostí knižnicu backboneJS. Výhodou tejto knižnice je prehľadné spracovanie práce s dátami a ich zobrazovaním na stránke. Z tohto dôvodu sme sa rozhodli práve pre toto riešenie.

Vykonalí sme podrobnú analýzu knižnice backboneJS a následne sme sa pustili do implementácie základnej komunikácie medzi systémom ALEF a Gaze. V ďalšej fáze sme chceli využiť túto komunikáciu na nejaké základné zásahy do stránky.

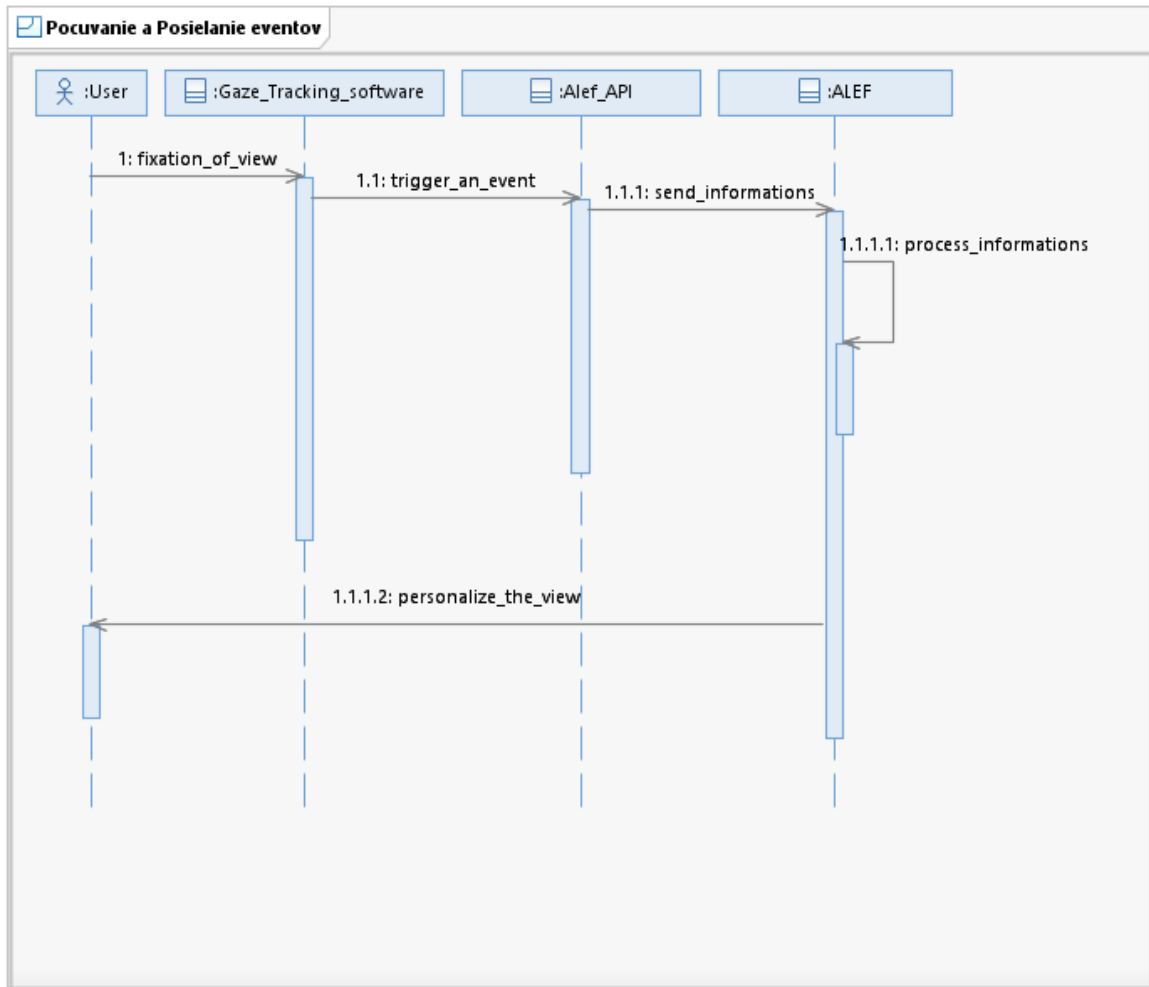
V tejto fáze sme si však pri jednej z konzultácií uvedomili, že využitie backboneJS nebude ideálne riešenie z hľadiska jeho prenositeľnosti. Zistili sme, že ak chceme, aby sme mohli rýchlo prispôsobiť riešenie iným aplikáciám bez potreby zahŕňania ďalších knižníc alebo gemov, tak bude najlepšie, ak využijeme iba čistý javascript.

4.2.2 Návrh riešenia

Celkové riešenie sme navrhli ako jednoduché posielanie a prijímanie udalostí (Obrázok 3). Základný prípad použitia je nasledovný:

- Systém Gaze zaznamená nejakú udalosť počas sledovania pohľadu (napr. fixácia pohľadu v jednej z oblastí záujmu).

- Pošle prislúchajúcu udalosť (event) prostredníctvom volania javascript funkcie trigger.
- Vytvorené API bude počúvať a zachytávať podobné udalosti.
- Následne podľa typu udalosti a z hľadiska kontextu, v ktorom sa nachádza, sa môžu vykonať rôzne akcie (napríklad upozornenia alebo prispôbenia obsahu).



Obrázok 3– Sekvenčný diagram základnej komunikácie

4.2.3 Implementácia základnej komunikácie

Na základe návrhu sme implementovali základnú komunikáciu medzi oboma systémami. Na strane systému Gaze sa iba vždy pri identifikácii fixácie pohľadu na určitom elemente pošle udalosť obsahujúcu XPath s cestou k danému elementu, na ktorý sa pozeráme. Na strane systému ALEF bol vytvorená trieda, ktorá bude počúvať na rovnakú udalosť `gaze_event`.

```

class @GazeTrackingEventListener
  constructor: () ->
    $('body').bind 'gaze_event', (e) =>getElement(e)
  getElement = (e) ->
    element = e.target
  
```

```
alert("Toto poslal " + element.id)
```

V kóde je vidieť, že sme sa snažili, aby sa volala samostatná funkcia (*getElement*). Tento prístup je lepší, ako priame vykonanie funkcie hlavne z hľadiska lepšej prehľadnejšej práce s jednotlivými udalosťami. V tomto príklade sa iba získa element a vypíše sa jeho *ID*. Celá implementácia je v špeciálnom jazyku coffescript, ktorý vychádza priamo z jazyka javascript.

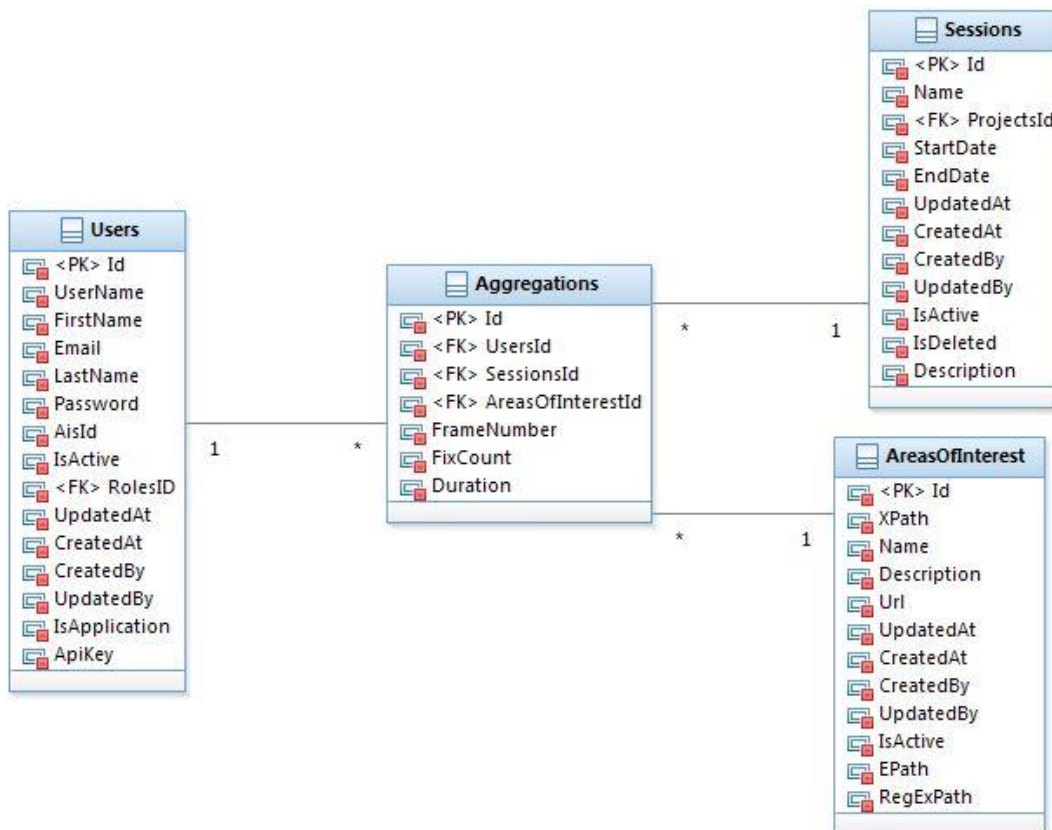
4.2.4 Testovanie v jazyku javascript

V aktuálnom stave projektu sme ešte nevykonávali žiadne testy, keďže nemáme hotovú žiadnu zložitejšiu logiku, ktorú by sa oplatilo testovať. Avšak vykonali sme podrobnú analýzu možných nástrojov na testovanie, z ktorých sme si vybrali nástroj JASMINE. Samotná analýza tohto nástroja sa nachádza v *prílohe A*.

4.3 Agregácie

4.3.1 Implementácia

MSSQL databáza bola rozšírená o novú tabuľku Aggregations, v ktorej budú ukladané priebežné agregované dáta pre konkrétne sedenie (SessionsId), probanta (UsersId) a záujmovú oblasť (AreasOfInterestId). Agregované dáta predstavujú počet fixácií pohľadom na konkrétny AOI (FixCount) a čas trvania týchto fixácií (Duration). Nakoľko sa počas jedného sedenia budú vytvárať agregácie v 1 minútových časových intervaloch, poradie týchto agregácií bude uložené vo (FrameNumber) (pozri Obrázok 4)



Obrázok 4 – Dátový diagram agregácií

Obrázok 4 – Datový diagram znázorňujúci tabuľku Aggregations a k nej pridružené tabuľky.

Z dátového modelu bola vygenerovaná trieda Aggregations, slúžiaca pre potreby ukladania a načítania agregáčnych objekt z MSSQL databázy a pre potreby kontextu Entity frameworku. Pre prácu s agregáčnymi bola vytvorená trieda AggregationDto (Data transfer object). Všetky operácie nad agregáčnymi objektmi sú obsiahnuté v triede AggregationsRepository, implementujúcej rozhranie IAggregationRepository. Trieda momentálne obsahuje nasledovné dopytovacie metódy, ktoré bude možné kedykoľvek v prípade potreby rozšíriť o ďalšie:

- **GetAggregationById (int Id)**
 - vráti agregáčny objekt so zadaným Id
- **AddAggregation (AggregationDto aggregation)**
 - uloží agregáčny objekt do databázy
- **GetAggregationsForSession (int sessionId)**
 - vráti všetky agregáčne objekty pre dané sedenie
- **GetAggregationsForSessionAndUser (int sessionId, int userId)**
 - vráti všetky agregáčne objekty daného probanta pre dané sedenie
- **GetAOITimeStatistic (int AOIId, int sessionId)**
 - vráti pre zadanú AOI a pre zadané sedenie celkové hodnoty počtu a trvania fixácií všetkých probantov počas celého trvania sedenia.
- **GetAOITimeStatistic (int AOIId, int sessionId, int startFrame, int endFrame)**
 - vráti pre zadanú AOI a pre zadané sedenie celkové hodnoty počtu a trvania fixácií všetkých probantov počas trvania sedenia v časovom rozsahu startTime a endTime.

Pre potreby vytvárania agregácií boli triedy AreaOfInterestRepository, AnotationRepository, SessionRepository a k nim pridružené rozhrania rozšírené o nové metódy dopytujúce sa do databázy MSSQL.

AreaOfInterestRepository:

- **GetSessionAreaOfInterestId (int sessionId)**
 - vráti ID všetkých aktívnych oblastí záujmu pre zadané sedenie

AnotationRepository:

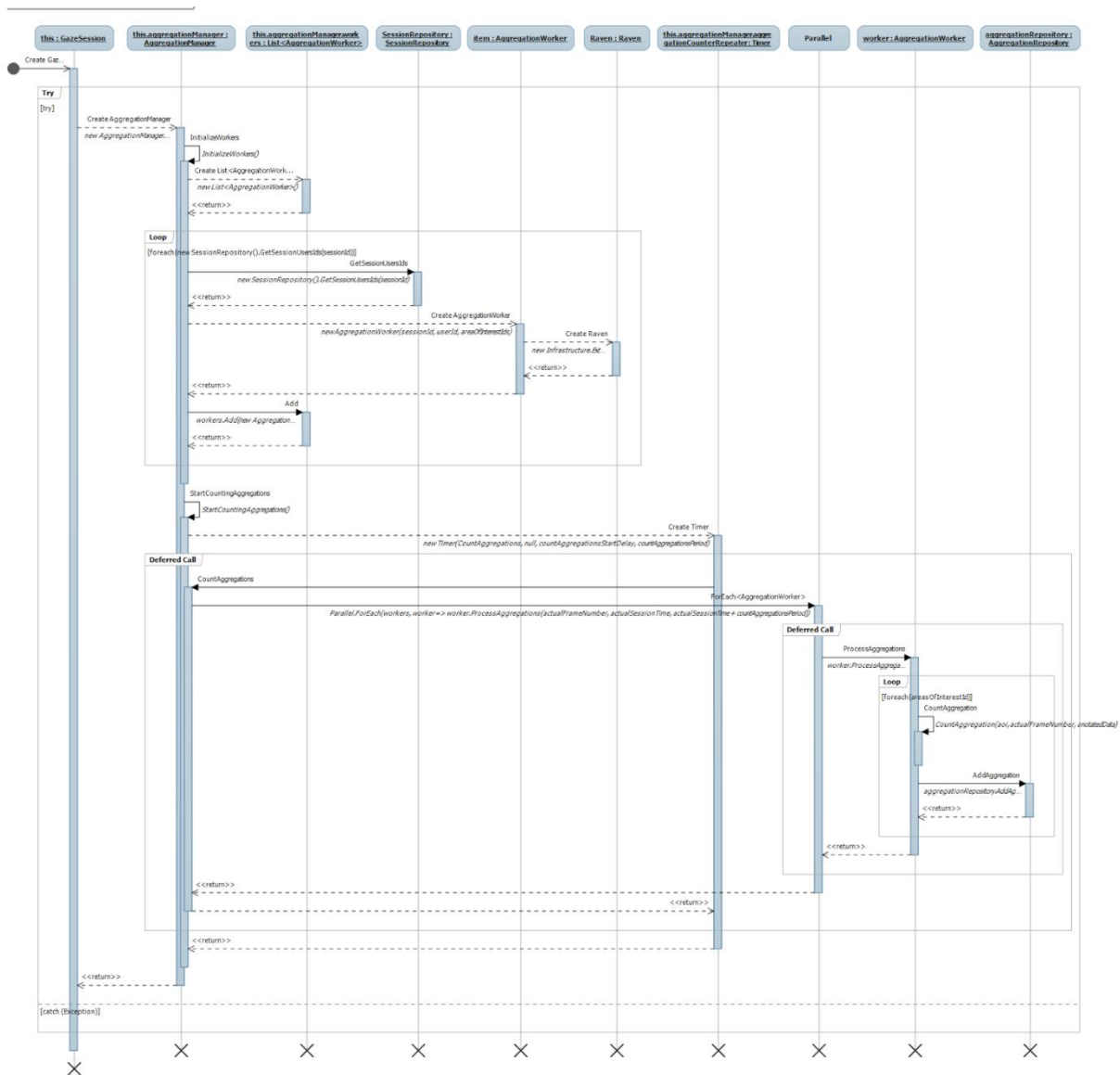
- **GetAnotationsForSessionAndAOI (int sessionId, int AOIId)**
 - vráti všetky anotačné objekty pre danú oblasť záujmu a dané sedenie

SessionRepository:

- **GetSessionUsersIDs (int id)**
 - vráti ID všetkých probantov pre dané sedenie

Proces vytváranie agregácií začína v triede GazeSession, ktorá v konštruktore vytvára okrem inštancie AnotationManager (stará sa o anotovanie prúdu dát) aj nami vytvorenú inštanciu triedy AggregationManager, ktorej konštruktor je naplnený ID aktuálneho sedenia a aktuálneho času. Tento konštruktor má na starosti vytvorenie kolekcie inšancií novovytvorenej triedy AggregationWorker, ktorých počet je rovný počtu probantov daného sedenia. Takto sa pri vytváraní agregáčného workera do jeho konštruktora predá ID sedenia, ID probanta a kolekcia ID oblastí záujmu pre dané sedenie. Následne AggregationManager spustí cyklický časovač, ktorý po uplynutí 1 minúty začína v 1-minútových intervaloch volať metódu CountAggregation, ktorá asynchrónne volá metódy ProcessAggregations všetkých agregáčnych workerov. Každý agregáčny worker načíta z RavenDB databázy anotované dáta

vzniknuté za poslednú minútu pre svoje sedenie a probanta. Následne pre každú oblasť záujmu vypočíta zo získaných dát agregácie a uloží ich do MS SQL databázy. Po skončení sedenia AggregationManager nechá posledný raz zbehnúť spočítanie agregácií a ukončí proces.



Obrázok 5 – Sekvenčný diagram pre vytváranie agregácií