Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

FIITKinect

(Projektová dokumentácia)

Tím: TeamToo
Vedúci: Ing. Vanda Benešová, PhD
Kontakt: tim2_fiit@googlegroups.com
Ak. rok: 2012/2013

Autori: Bc. Ján Antala
Bc. Martin Čertek
Bc. Jakub Gondár
Bc. Ondrej Grman
Bc. Silvia Hudačinová
Bc. Michal Igaz
Bc. Richard Sámela

1 Úvod

Tento dokument slúži ako špecifikácia požiadaviek pre softvérový produkt „MaeToo“. Projekt je vypracovávaný tímom 2 v rámci predmetu tímový projekt 1 a 2.

1.1 Význam dokumentu

Životný cyklus každého softvérového produktu začína analýzou a návrhom. Je to esenciálne pre správne smerovanie vývoja softvérového produktu.

1.2 Prehľad systému

Softvér sa vyvíja ako decentralizovaná aplikácia so serverom a modulmi. Moduly spolu navzájom komunikujú cez štandardizovaný protokol, čo umožňuje modulom aby boli implementované v rôznych jazykoch a navonok ukazovali iba svoje rozhrania.

Je to aplikácia, ktorá má ako hlavnú úlohu sprostredkovať ovládanie výstupného zariadenia Kinectom. Každé vstupné a každé výstupné zariadenie má vlastný modul, ktorý má na starosti správu tohto zariadenia. Moduly vstupných zariadení majú možnosť správy eventov (v tomto prípade gest, reči a iných) výstupné zariadenia spravujú akcie. Ústredná časť aplikácie, inak nazývaná aj core aplikácia má za úlohu prepájať eventy s akciami, tieto prepojenie uchováva. Nad touto aplikáciou je aj grafické užívateľské rozhranie pre pohodlnejšiu prácu a konfigurovatelnosť systému.

1.3 Perspektíva produktu

V súčasnosti sa už na trhu vyskytuje viacero produktov, ktoré využívajú Kinect ako primárne vstupné zariadenie na ovládanie výstupných zariadení telom. Tento produkt však ponúkne značne inovatívne vylepšenie k súčasným produktom ako napríklad ovládanie výstupných zariadení smartfónom.

Produkt má veľký potenciál stať sa primeranou a hlavne lacnejšou alternatívou k zabehnutým riešeniam.

1.4 Ciele

Tento softvérový produkt je vyvíjaný hlavne za účelom poskytnutia plnohodnotného systému na ovládanie výstupných zariadení pomocou Kinectu alebo Androidu. Určený je hlavne bežným užívateľom ako možnosť ovládania rôznych zariadení (televízor, rádio, osvetlenie, atď) mobilným zariadením, gestom prípadne zvukom. Uplatnenie preň nájdu aj výskumníci, ktorí budú mať rozšírený a plný prístup k všetkým funkciám softvérového produktu. Pre určenie optimálnych spojení udalostí (events) a akcií (actions) im budú poskytnuté nástroje na vytváranie, zmenu a mazanie gest (gestá teľa alebo tzv. „touch“ gestá na mobilnom

zariadení) či zvuku. Cieľom je aj vytvoriť transparentnú organizáciu experimentov, čo sa docieli poskytnutím možnosti ich exportu a importu.

1.5 Ohraničenia

Vychádzajúc z návrhu riešenia aplikácie, možno identifikovať obmedzenia:

- Riešenie vyžaduje prostredie operačného systému Windows pre správnu funkcionality rozpoznávania gest pomocou Kinectu
- Nutnosť sprevádzkovania knižnice OpenCV 2.3 z dôvodu využívanie tejto knižnice pri časti aplikácie pracujúcej so senzorom Kinect
- Pre zaznamenávanie gest a zvukových povelov je pre túto časť funkcionality nevyhnutné pripojenie a správne fungovanie senzoru Kinect
- Android aplikácia na rozoznávanie dotykových gest je kompatibilná s verziou operačného systému Android 2.2 a vyššou
- Pre rozpoznávanie reči na Android zariadení je požadované nainštalovanie balíčku Speech Recognition
- Nakonfigurovaný server podporujúci Node.js a databázové riešenie Redis
- Pre správne fungovanie a spoluprácu zariadení je nutné, aby tieto boli vzájomne zosieťované

1.6 Referencie

1. Projektová dokumentácia tímu Art Quintet, č. 11

2 Opisy príbehov

2.1 Analýza

2.1.1 Špecifikácia požiadaviek

V rámci systému sú vyčlenené dve typické roly používateľov – bežný používateľ a výskumník.

Bežný užívateľ je človek, ktorý reálne používa náš softvérový produkt. Nie sú a ani nemôžu byť naňho kladené iné ako minimálne požiadavky na prácu s počítačom. Funkcionalita aplikácie je preňho zúžená tak, aby nemusel riešiť pokročilé nastavenia a ďalšiu funkcionalitu.

Výskumník je človek, ktorý softvérový produkt môže využívať k jeho plnému potenciálnu. Má k dispozícii mnoho možností ako produkt upraviť a vyhľadáva optimálne možnosti, ktoré má využívať bežný užívateľ.

Títo používatelia typicky vykonávajú v systéme nasledujúce operácie.

2.1.1.1 Príbehy používateľov

2.1.1.1.1 Bežný používateľ

Ako používateľ chcem mať v grafickom rozhraní možnosť zobrazenia jednotlivých dvojíc gest a akcií pre ich lepšie napodobenie.

Ako používateľ chcem mať k dispozícii vizuálne zobrazenie eventov pre lepšiu prácu s programom.

Ako používateľ chcem mať k dispozícii spätnú väzbu o zachytávaní mojich povelov aby som vedel, že zariadenie, ktoré používam pracuje ako má.

2.1.1.1.2 Výskumník

Ako výskumník chcem mať v rámci užívateľského prostredia možnosť výberu eventu a akcie cez „select box“ aby som mohol ovládať program jednoducho.

Ako výskumník chcem mať k dispozícii plne konfigurovateľné užívateľské prostredie odlišné od jednoduchého menu bežného užívateľa.

Ako výskumník chcem mať k dispozícii možnosť spustenia užívateľského rozhrania aby som mohol v praxi odskúšať svoje experimenty.

Ako výskumník chcem mať k dispozícii nástroj na spájanie eventov s akciami aby som si mohol systém prispôsobiť svojim potrebám.

Ako výskumník chcem mať k dispozícii nástroj na spájanie eventov s akciami aby som mohol skúmať optimálne sady event – akcia.

Ako výskumník chcem mať možnosť vytvoriť nový experiment, aby som si vedel rozdeliť svoju prácu.

Ako výskumník chcem mať možnosť zrušiť existujúci experiment.

Ako výskumník, chcem mať možnosť úpravy vytvorených experimentov nasledujúcim spôsobom:

1. Pridávanie dostupných gest
2. Odoberanie pridaných gest
3. Pridávanie dostupných akcií
4. Odoberanie pridaných akcií

Ako výskumník chcem vedieť vytvárať nové gestá aby som mohol určiť optimálnu sadu gest pre bežného používateľa.

Ako výskumník chcem vedieť upravovať prípadne mazať už vytvorené gestá aby som zbytočne nezaberal pamäť počítača.

Ako výskumník chcem mať možnosť vytvoriť nové „dotykové gestá“ aby bolo možné systém rozširovať.

Ako výskumník chcem vedieť upravovať prípadne mazať „dotykové gestá“ aby som zbytočne nezaberal pamäť počítača.

Ako výskumník chcem vedieť pridávať akcie tak, aby bolo možné systém naďalej rozširovať.

Ako výskumník chcem vedieť upravovať prípadne mazať akcie, ktoré nie sú naďalej aktuálne.

Ako výskumník chcem mať možnosť exportovania prípadne importovania experimentov.

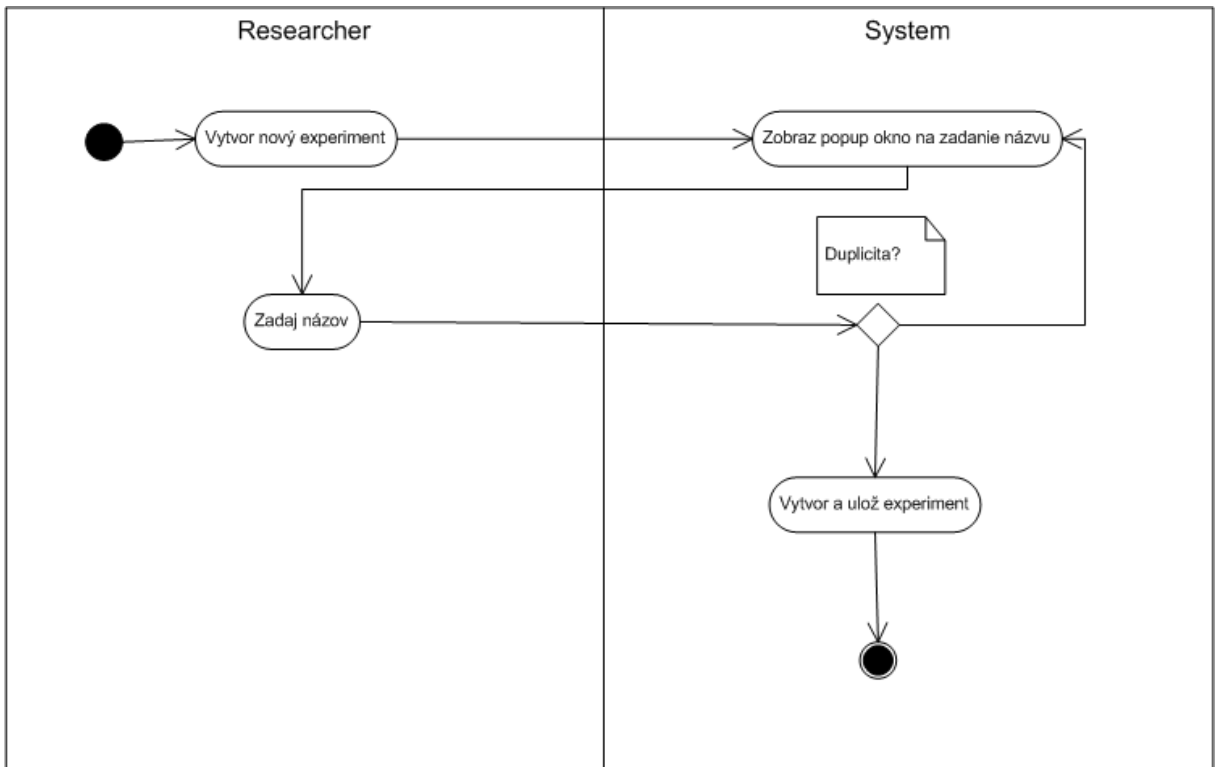
3 Funkčná špecifikácia produktu

Funkcionalita aplikácie sa priamo odvodzuje od príbehov používateľov. Produkt rozlišuje nasledujúce funkcionality:

1. Správa experimentov
2. Správa „touch“ gest
3. Správa gest
4. Správa akcií
5. Mapovanie
6. Zobrazenie uložených gest

3.1.1 Správa experimentov

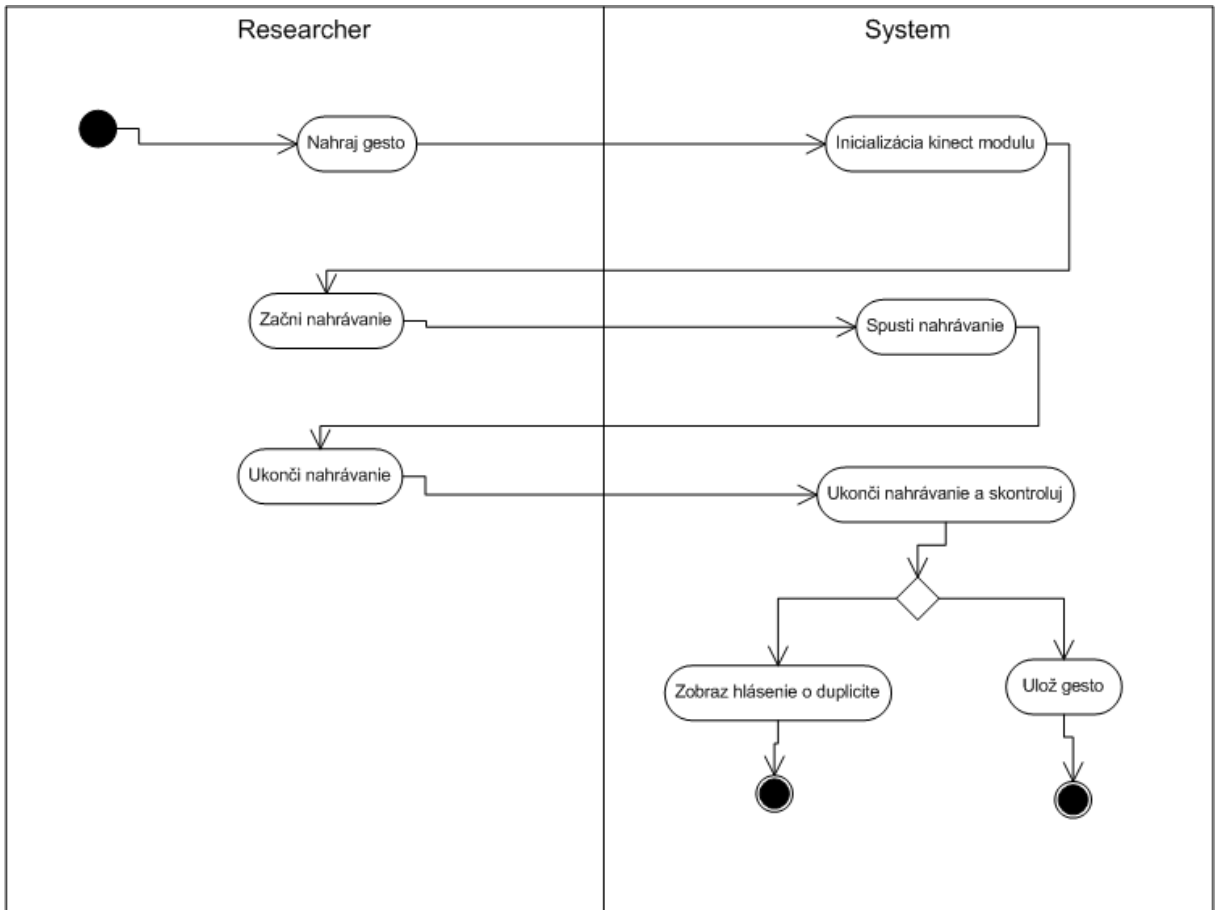
K tejto funkcionalite má prístup výskumník (na obrázku Researcher). Do správy experimentov patrí vytvorenie nového experimentu a zmazanie starého experimentu. Táto funkcionalita je podstatná najmä z toho dôvodu, že poskytuje niečo ako základ. Ďalšie podstatné funkcie softvérového produktu ako pridávanie gest a akcií, prípadne ich vzájomné mapovanie prebieha v rámci experimentu, ktorý vykonáva výskumník.



Obr. č. 1: Vytváranie nového experimentu

3.1.2 Správa „touch“ gest

Táto funkcionalita zahŕňa používanie mobilného zariadenia so systémom Android ako vstupného zariadenia. Patrí sem vytváranie, úprava a zmazanie „touch“ gesta. Túto možnosť môže používať výskumník aj bežný užívateľ. Bežný užívateľ však využíva iba časť na ovládanie, konfiguračnú má plne k dispozícii iba výskumník.

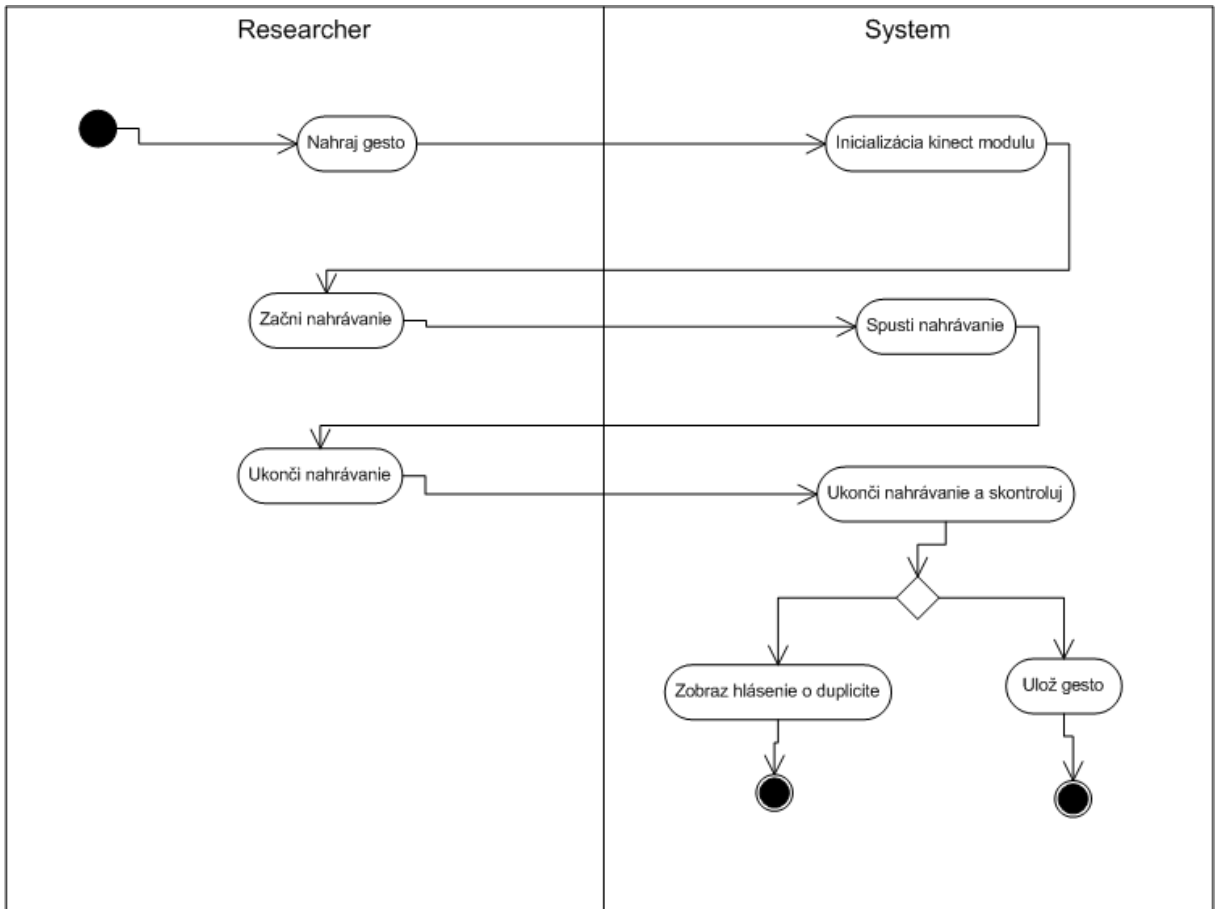


Obr. č. 2: Vytváranie nového „touch“ gesta

3.1.3 Správa gest

Táto funkcionalita zahŕňa používanie zariadenia Kinect ako vstupné zariadenie systému. Zahŕňa najmä funkcionalitu vytvárania, upravovania a mazania gest. Túto funkcionalitu môže využívať iba výskumník. Používa ju na vyhľadávanie optimálnych gest pre bežného užívateľa. Nepotrebné alebo nepodarené gestá maže.
Požiadavky na gesto:

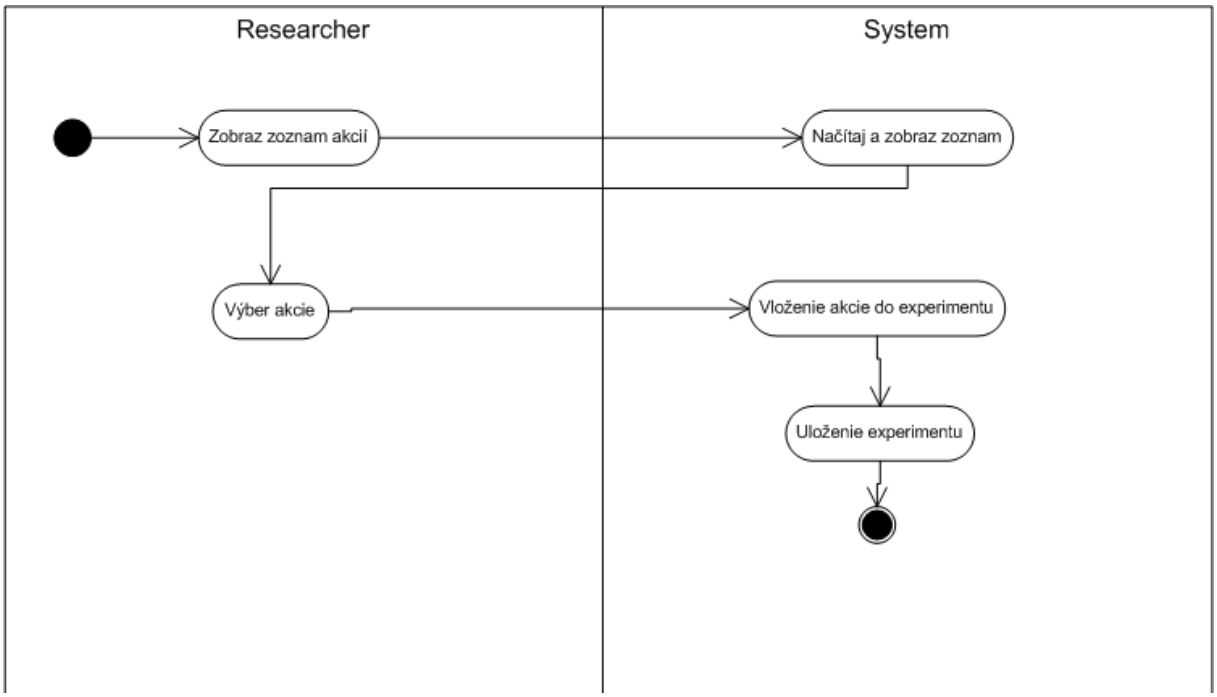
- Jednoduché
- Intuitívne
- Ľahko napodobniteľné
- Neduplicitné



Obr. č. 3: Vytváranie nového gesta

3.1.4 Správa akcií

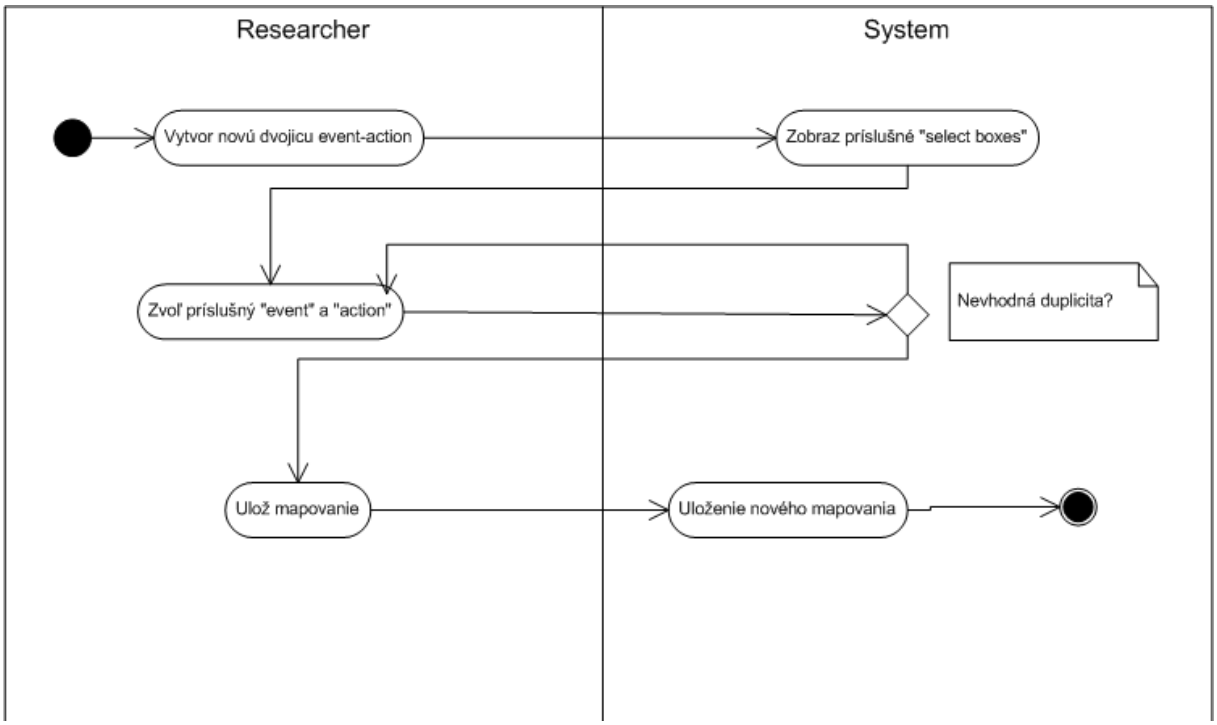
Táto funkcionalita zahŕňa problematiku výstupných zariadení. Zastrešuje hlavne povely, ktoré sa používajú v rámci systému tak, aby vykonali požadované úkony. S touto časťou narába výskumník. Jej nutným predpokladom je funkcionalita správy experimentov. Akcie sa dajú pridávať jedine do experimentu. Táto funkcionalita zahŕňa pridávanie nových akcií alebo doberanie pôvodných.



Obr. č. 4: Pridanie novej akcie do experimentu

3.1.5 Mapovanie

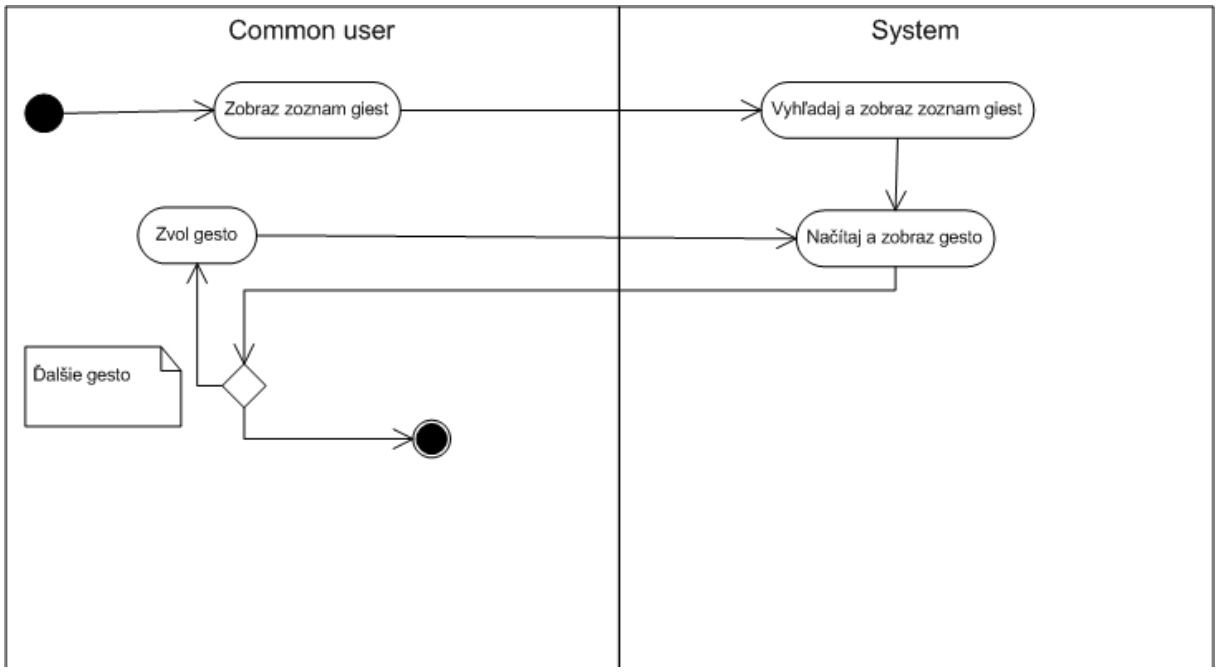
Táto funkcionalita zahŕňa problematiku prepájania event (udalosť) a action (akcia). Je to základný nástroj práce výskumníka. Sem patrí najmä funkcionalita pridávania, zmeny a odobratia mapovania. Náleží výskumníkovi.



Obr. č. 5: Vytváranie nového mapovania

3.1.6 Zobrazenie uložených gest

Táto funkcionalita zahŕňa zobrazenie dostupných gest a ich prepojení s akciami. Táto funkcionalita je dostupná bežnému užívateľovi aj výskumníkovi. Funguje ako bežné zobrazovanie. Slúži hlavne pre vizualizáciu gest.



Obr. č. 6: Zobrazenie uložených gest

4 Analýza

4.1.1 Analýza QT

Qt je jeden z dvoch najpoužívanejších multiplatformových frameworkov, ktorý je rozšírenej miere používaný pri vytváraní softvérových aplikácií s GUI. V týchto prípadoch býva Qt označovaný aj ako widget toolkit. Bol vytvorený v roku 1999

Primárne je Qt určený pre programovanie v C++, pričom na rozšírenie C++ používa viacero makier a aj špeciálny kompilátor – Meta Object Copmiler (moc).

Použitie Qt je rozšírené aj na ďalších platformách, ako desktopových, tak aj mobilných. Ako knihovňa existuje pre programovacie jazyky : Python (PyQt),Ruby (QtRuby), C, Perl, Pascal, C#, Java (Jambi).

Vyznačuje sa dobrou podporou jazykových lokalizácií pre aplikáciu. Ponúka viacero funkcionalít nesúvisiacich s GUI ako : SQL prácu s databázami, parsovanie XML súborov, podporu viacerých vlákien, sieťovú podporu.

Je distribuovaný pod licenciou GNU Lesser General Public License, je voľne dostupný a poskytovaný ako open-source. Podporuje viacero prekladačov ako GCC C++, ako aj prekladače produktov MS Visual Studio.

Vývoj Qt je realizovaný ako nekomerčný open-source pri zapojení komunity programátorov a za podpory spoločností ako Nokia a Digia. Popri voľnej verzií je predávaný so špeciálnou podporou a funkcionalitou aj ako komerčné riešenie.

Výhody	Nevýhody
Multiplatformovosť	Netypický prístup vyžadujúci pochopenie – signals , slots
Využívanie natívneho vzhľadu operačného systému	

4.1.2 Windows Forms (WinForms)

Windows Forms je názov pre nové aplikačné programové rozhranie (API), ktoré je súčasťou .NET Frameworku. Toto API ponúka prístup k natívnym prvkom Windows rozhrania, ktoré obaľuje a beží v takzvanom „riadenom (managed)“ kóde. Môže sa zdať, že sa jedná o náhradu MFC, neponúka funkcionalitu podobnú paradigme Model-View-Controller. Windows Forms na svoj beh potrebujú nainštalovaný .NET framework, ktorý je plne implementovaný iba pre platformu Windows.

Windows Forms používajú udalosťami riadenú (event-driven) architektúru, ktorú poskytuje .NET Framework. Jazyk, ktorý sa používa na implementáciu Windows Forms je najčastejšie C#. S použitím integrovaného prostredia Visual Studio sa dajú vytvárať vo Windows Forms moderné a bohaté aplikácie. Visual studio má integrovaný pokročilý dizajnér formulárov Window Forms. V spolupráci s ostatnými komponentmi .NET Frameworku a jazykom C# sa dajú vytvárať aplikácie kvalitne, rýchlo a efektívne.

Windows Forms vedia pracovať aj s jazykom C++, ale Microsoft Visual Studio nepodporuje IntelliSense pri takomto vývoji, čo nepriamo naznačuje, že použitie C++ s Windows Forms nie je príliš vhodné.

Výhody	Nevýhody
Kvalitný nástroj pre tvorbu GUI, jednoduchý na použitie a programovanie	Iba Windows platforma s .NET Framework
Manažovaný kód ponúka nevídané možnosti debugovania a podpory pri vývoji (IntelliSense)	Nepracuje dobre s C++, navrhnuté pre C#

4.1.3 Zhodnotenie a výber technológie GUI pre aplikáciu

V našom projekte pracujeme s knižnicou FIITKinect, ktorá bola vyvinutá v rámci minuloročného Tímového projektu. Táto knižnica je potrebná pri rozpoznávaní gest zo zariadenia Kinect. Knižnica je implementovaná v C++, čo nám kladie obmedzenia pri výbere.

Aby sa zabezpečila integrácia, tak aj grafické rozhranie musí byť postavené na jazyku C++. Minuloročná aplikácia používa knižnice QT GUI.

Po zvážení viacerých faktov z analýzy grafických rozhraní sme sa rozhodli, že prototyp budeme ďalej dopracovávať v rozhraní QT GUI na platforme C++.

4.1.4 Analýza databázových prostredí

4.1.4.1 Relačné – SQL (MySQL, PostgreSQL)

- musí byť definovaná pevná štruktúra tabuliek a väzieb medzi nimi na začiatku, ktorá sa napĺňa

- prijaté údaje je nutné previesť do formy podporovanej databázou – SQL jazyka

+ dáta sú uložené raz a je možné zobrazovať ich rôznymi spôsobmi podľa rôznych filtrov a sortov

+ dá sa vytvoriť transakcia a keď nezbehne, niektorá časť je možné proces vrátiť

4.1.4.2 NoSQL (Redis)

- nie je možné dáta prechádzať podľa atribútov

- dáta treba nakopírovať viackrát kvôli rôznym náhľadom (vznikajú duplicity)

- nedá sa vytvoriť transakcia, v ktorej by bolo možné podľa rozhodnutia programátora vrátiť zmeny

+ veľmi rýchla, pretože beží v operačnej pamäti

+ možnosť vkladať a získavať dáta v JSON formáte, ktorý je použitý ako komunikačný protokol v rámci celej aplikácie (nie je nutné dáta prevádzať na SQL dopyty)

4.1.4.3 Výber databázy

Po zhodnotení pomeru počet uchovávaných údajov/ rýchlosť a na základe výhod, ktoré poskytuje, bola vybraná NoSQL varianta v podobe Redis.

4.1.4.4 Redis

- open-source nosql key-value databáza
- slúži na ukladanie dát na server obsahujúca kľúče (keys) na string-y, hash-e, set-y, list-y a sorted set-y
- umožňuje jednoduché transakcie riešiace konflikty prístupu rôznych klientov
- beží v operačnej pamäti počítača => rýchlosť
- k dispozícii je 15 nezávislých databáz, ktoré sa prepínajú príkazom *SELECT <number>*
- dáta sú ukladané na disk počítača- tri možné typy (po každej zmene, v nejakých časových intervaloch, každú sekundu)

4.1.5 Dotykove gesta na platforme Android

4.1.5.1 Gesto

- rukou nakreslený objekt na displej mobilného zariadenia
- môže mať jednoduchý ťah alebo môže byť vyskladané z viacerých ťahov
- je závislé od času
- je reprezentované ako sekvencia bodov na displeji zariadenia
- na jeho rozpoznanie je možné použiť knižnicu gest - tá môže byť vlastná, s vlastnými gestami alebo pôvodná, ktorá je poskytovaná priamo platformou
- rozpoznávanie prebieha na špeciálnom type grafického rozhrania
- gesta je možné pridávať do vlastnej knižnice gest a tu je možné uložiť na diskový priestor zariadenia prípadne odoslať po sieti
- je možné reprezentovať aj graficky v podobe bitmapy

4.1.5.2 Reprezentácia

Samotné gesto je reprezentované bodmi, ktoré v sebe zahŕňajú x,y súradnice a údaj o časovej známke. Jednotlivé gestá je možné pridávať do vlastných knižníc pre rozpoznávanie

gest. Dané knižnice je možné vo formáte súboru XML ďalej ukladať, pripadne inak spracovávať.

4.1.5.3 Porovnávací algoritmus

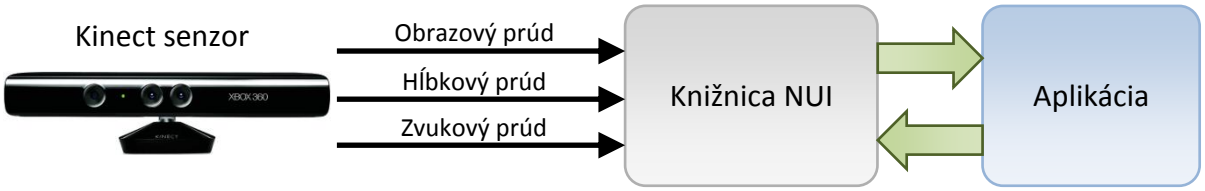
Framework pre prácu s gestami na platforme Android ponuka triedu, ktorá obsahuje algoritmy pre konkrétne porovnanie a rozpoznanie gest. Rozpoznanie gest sa vykonáva spracovaním gesta nasledovnými pomocnými algoritmami:

- počítanie obdĺžnikových okrajov gesta
- dĺžka gesta
- pohyb
- rotácia
- škálovanie
- počítanie Euklidových alebo Kosínusových vzdialeností medzi dvomi gestami

4.1.6 Kinect pre Windows – architektúra

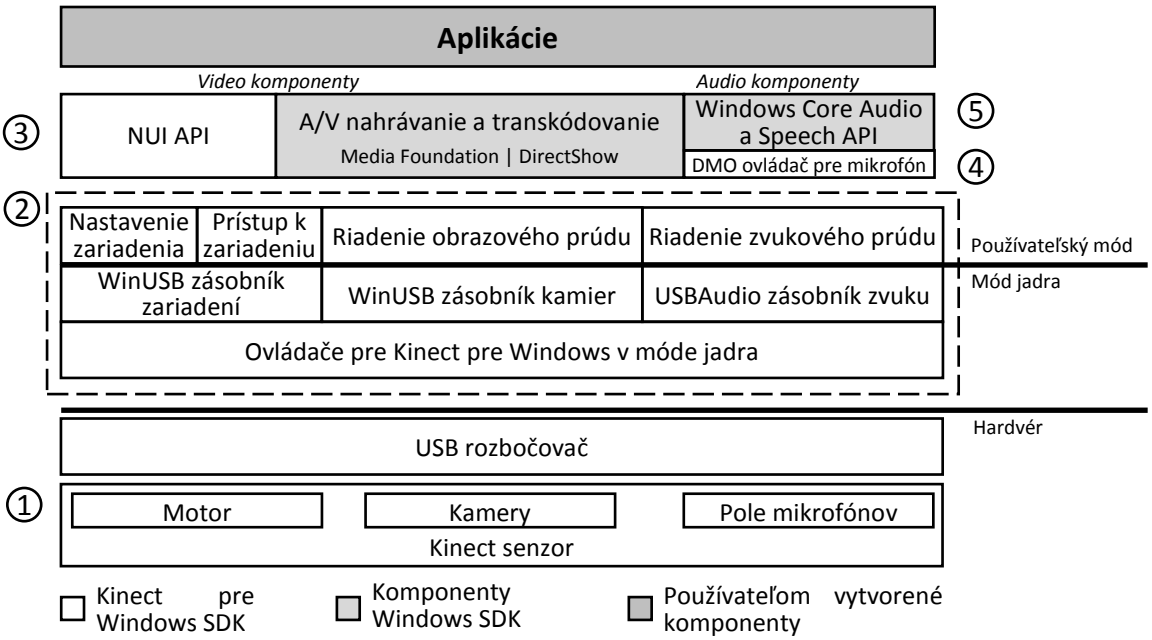
Táto podkapitola bola prebratá z analýzy architektúry Kinect tímu Art Quintet (č. 11) z akademického roku 2011/2012.

Beta SDK poskytuje sofistikovanú softvérovú knižnicu a nástroje, ktoré pomáhajú programátorom použiť bohatú formu prirodzených vstupov získaných zo zariadenia Kinect na vnímanie a reakciu na udalosti reálneho sveta. Kinect spolu so softvérovou knižnicou interagujú s aplikáciou, ako na Obr. č. 7.



Obr. č. 7: Interakcia hardvéru a softvéru s aplikáciou.

Komponenty beta SDK sú zobrazené na Obr. č 8.



Obr. č 8: Architektúra beta SDK.

Komponenty beta SDK z Obr. č 8 zahŕňajú:

1. **Hardvér zariadenia Kinect**

   Hardvérové komponenty, zahŕňajúc senzor Kinect a USB rozbočovač, cez ktorý sa pripája k počítaču.

2. **Ovládače Microsoft Kinect**

   Windows 7 ovládače pre senzor Kinect, ktoré sú nainštalované spolu s beta SDK. Ovládače Microsoft Kinect podporujú:

   - pole mikrofónov senzora Kinect, ku ktorému je možné pristúpiť cez štandardné zvukové API vo Windows,
   - prenos obrazu a hĺbkových dát pomocou prúdov,
   - funkcie na enumeráciu zariadení, ktoré umožňujú aplikácii použiť viac ako jeden senzor Kinect pripojený k počítaču.

3. **NUI API**

   Množina rozhraní pre tvorbu aplikácií, ktorá získava údaje z obrazových senzorov a riadi zariadenie Kinect.

4. **KinectAudio DMO**

   Kinect DMO rozširuje podporu poľa mikrofónov vo Windows 7 pre funkcionalitu lokalizácie zdroja zvuku.

5. **Štandardné API Windows 7**

   Zvukové, rečové a multimediálne API vo Windows 7.

5 Návrh

5.1.1 Používateľské rozhranie

Používateľské rozhranie je vytvárané podľa definovaných požiadaviek zadaných pri špecifikácií požiadaviek. Z uvedeného vyplýva, že vytvorené používateľské rozhranie sa má vyznačovať jednoduchosťou použitia, intuitívnosťou pre používateľov, z radov používateľov, ktorí pracujú s aplikáciou na štandardnej úrovni.

Pri návrhu na základe požiadaviek bol vypracovaný koncept 2 používateľských rozhraní plniacich vzhľadom na funkcionalitu programu rozdielnu náplň. Je navrhnuté rozhranie na prácu s vytváraním, nastavovaním a akoukoľvek prácou s experimentom spočívajúcim v konfigurácii kombinácií ovládania pomocou gest a zvukov.

Druhým rozhraním je jednoduché, informačné rozhranie pre používateľa z radov bežných používateľov aplikácie, ktorých náplňou nie je práca s nastavovaním experimentov. Toto rozhranie slúži pre sprostredkovanie minimálnej množiny informácii potrebných pre interagujúceho používateľa, ktorý nemá za úlohu pracovať s experimentmi, ale iba používať aplikáciu pre svoj úžitok.

Grafické rozhranie určené na prácu s nastavovaním experimentov pozostáva z viacerých obrazoviek, slúžiacich na prácu s parametrami ako – správa povelov – gestá zo senzora Kinect, zvukové povely, dotykové gestá z mobilných zariadení, správa vykonateľných akcií a obrazovky slúžiacej na mapovanie akcií vykonaných po zaznamenaní povelu od zariadení.

Jednotlivé obrazovky sú podrobnejšie opísané v nasledujúcich častiach tejto kapitoly. Návrh rozhrania nie je vytvorený ako verná kópia skutočnosti, predstavuje ich schematické zaznamenanie pohľadu na rozmiestnenie jednotlivých ovládacích a informačných prvkov na obrazovke. Konkrétne grafické prevedenie – tvary a farby je realizované priamo pri implementácií podľa štandardných grafických tém.

5.1.1.1 Aplikácia Experimentátora

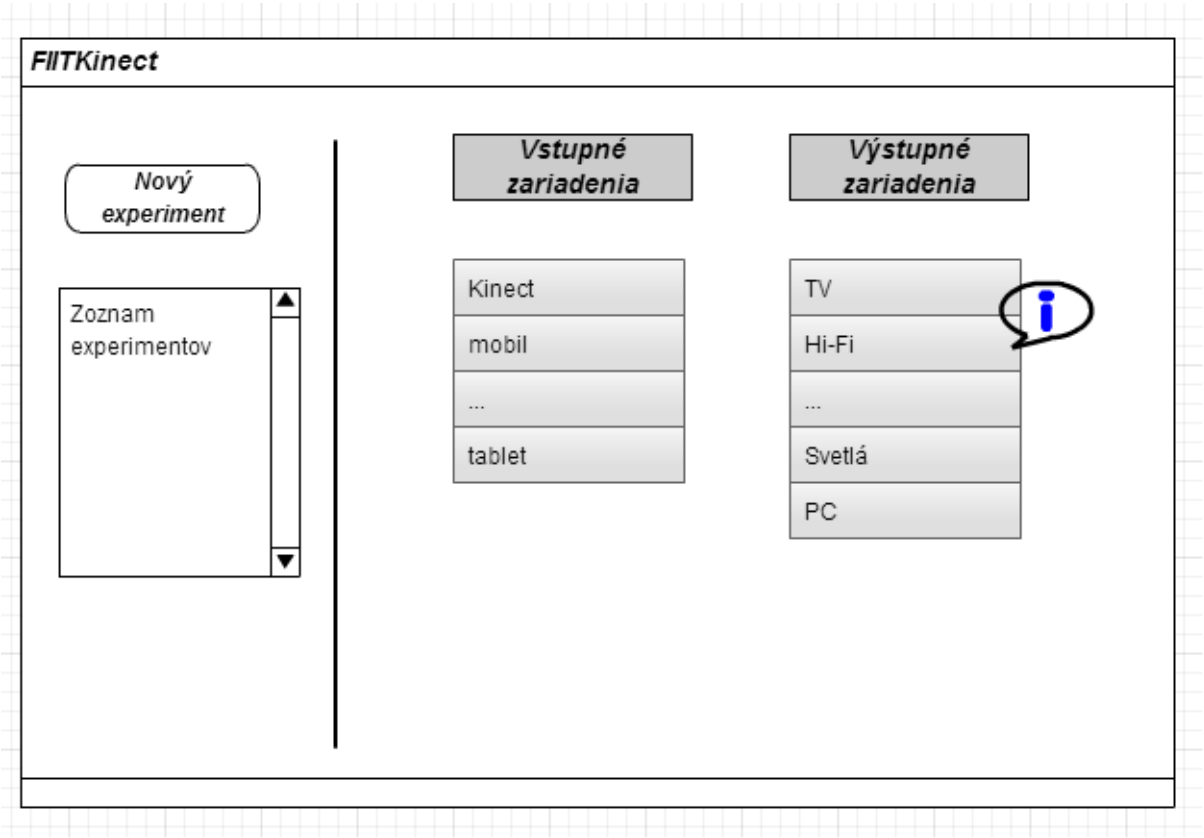
5.1.1.1.1 Práca s experimentmi

Obrazovka na obrázku Obr. č. 9 ja zameraná na prácu s experimentmi. Predstavuje základný pohľad pre experimentátora pri spustení aplikácie. Obsahuje zoznam existujúcich experimentov, ktoré sa v aplikácií nachádzajú. V ľavom hornom rohu sa nachádza tlačidlo umožňujúce vytvoriť alebo pridať (importovať) experiment do katalógu experimentov.

Na pravej strane obrazovky sa nachádza informačné pole zobrazujúce v dvoch stĺpcoch množinu zariadení umožňujúcich získavanie informácií od používateľa (gestá, reč) a množina výstupných zariadení, na ktorých budú realizované výstupné akcie.

Zariadenie, ktoré sú pripojené do systému a sú aktívne sú graficky rozlíšené pre jednoduchšie identifikovanie používateľom.

Pri každom vstupnom a výstupnom zariadení možno zobraziť doplnkovú množinu informácií pri označení konkrétneho zariadenia.



Obr. č. 9: Hlavná obrazovka Experimentátorskej aplikácie.

5.1.1.1.2 Práca s experimentom

Po vytvorení/zvolení experimentu na prácu sa zobrazí používateľovi obrazovka na obrázku Obr. č. 10, ktorá ponúka možnosti na konfigurovanie parametrov experimentu. Pre každý experiment je zobrazená pomocou záložiek – tabov v hornej časti obrazovky množina k nemu patriacich povelov, akcií ako aj priradeného mapovania. V pravom hornom rohu sa nachádza tlačidlo na spustenie experimentu s nastavenými parametrami, čo vyvolá spustenie aplikácie určenej pre bežnú interakciu bez možnosti nastavovania experimentu.

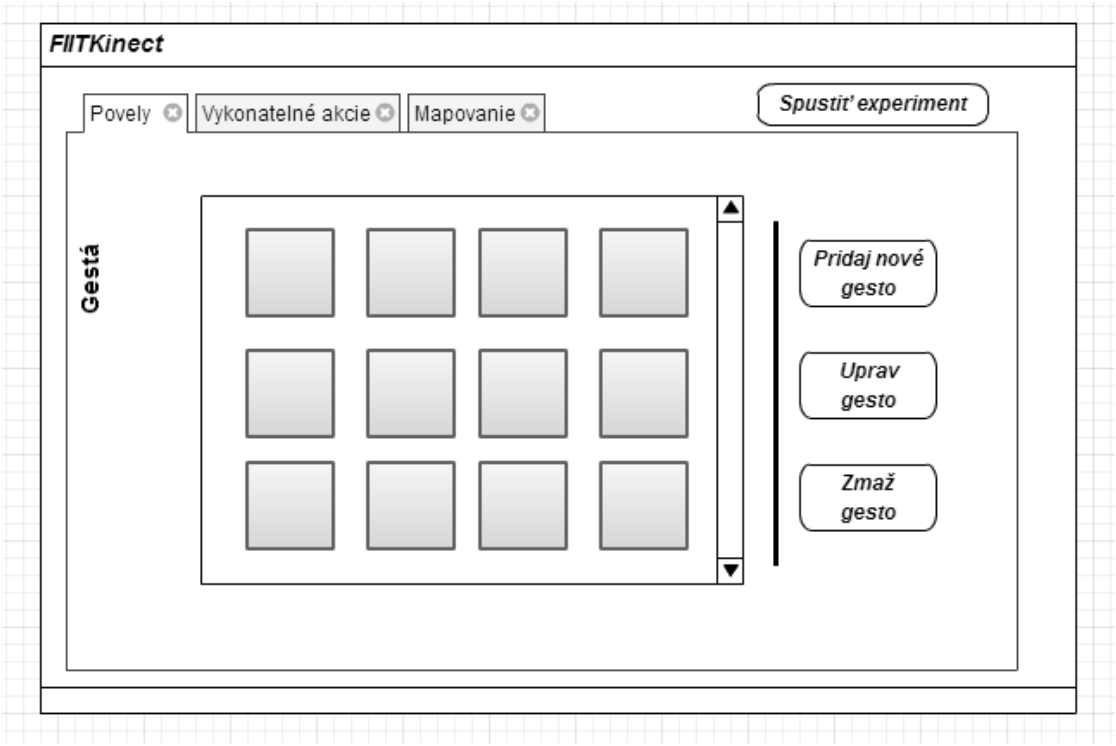
5.1.1.1.3 Práca s povelmi

Záložka „Povely“ obsahuje na ľavej strane ďalšie pracovné záložky na prácu s gestami, nastavovaním zvukových povelov, dotykových gest prostredníctvom mobilných zariadení.

Pri úvodnom spustení prázdneho experimentu je používateľovi ponúknutá východisková množina gest. Gesto, ktoré nechcem v experimente uchovať možno označiť a z experiment vyňať.

V pravej časti obrazovky sa nachádzajú tlačidlá na pridanie nového gesta – stlačenie vyvolá zobrazenie obrazovky umožňujúcej natrénovanie gesta a pridanie do množiny gest, upravenie existujúceho gesta - ktoré rovnako zobrazí obrazovku znovu natrénovanie existujúceho gesta, a tlačidlo na zmazanie gesta z experimentu.

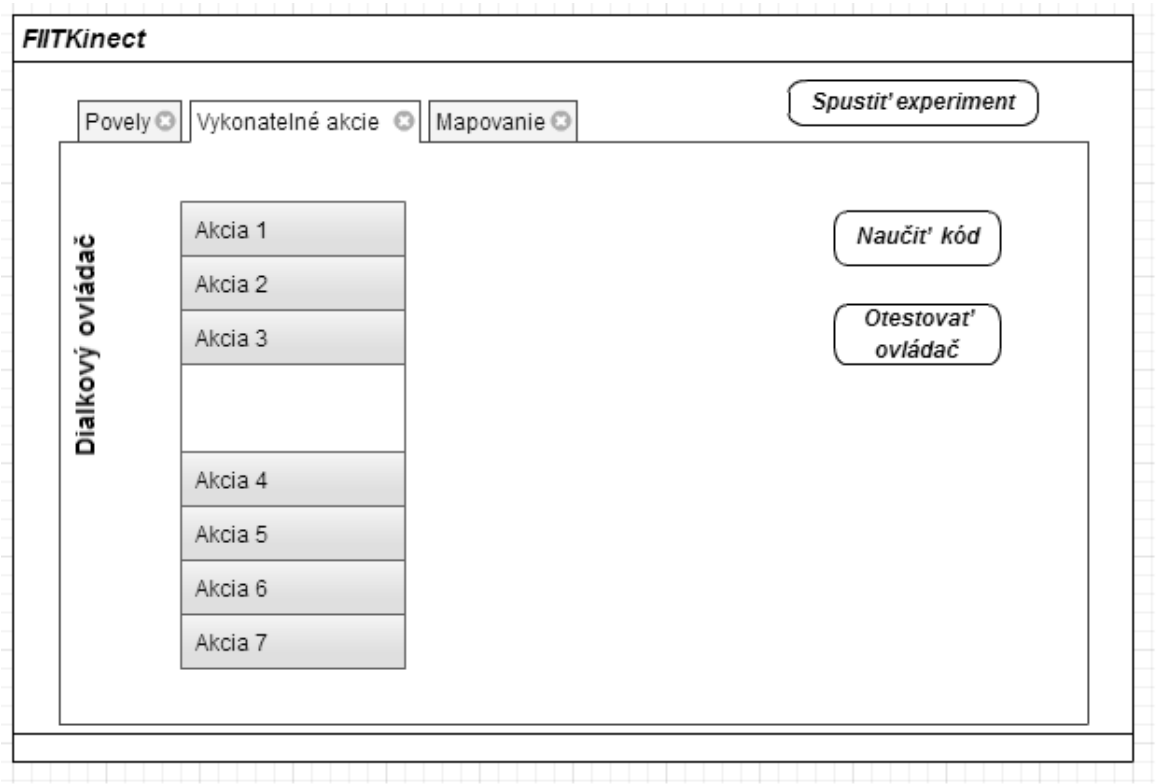
Prevažnú časť obrazovky tvorí mozaika obrazových informácií o jestvujúcich gestách v podobe obrázku a krátkej spustiteľnej sekvencie videa, ktoré zaznamenáva postupné vykonanie gesta.



Obr. č. 10: Obrazovka na prácu s povelmi.

5.1.1.1.4 Práca s akciami

Obrazovka ponúka možnosť práce s akciami, ktoré je možné vykonávať na výstupných zariadeniach. Na ľavej strane záložky *Vykonateľné akcie* sa nachádza zoznam existujúcich akcií v systéme. Na pravej strane sa po stlačení tlačidla *Naučiť kód* zobrazí obrazovka umožňujúca natrénovanie diaľkového ovládača k výstupným zariadeniam. Pod týmto tlačidlom sa nachádza tlačidlo *Otestovať ovládač,* ktoré po stlačení zobrazí možnosť reálneho otestovania si funkčnosti nastavenia jednotlivých akcii ovládača.

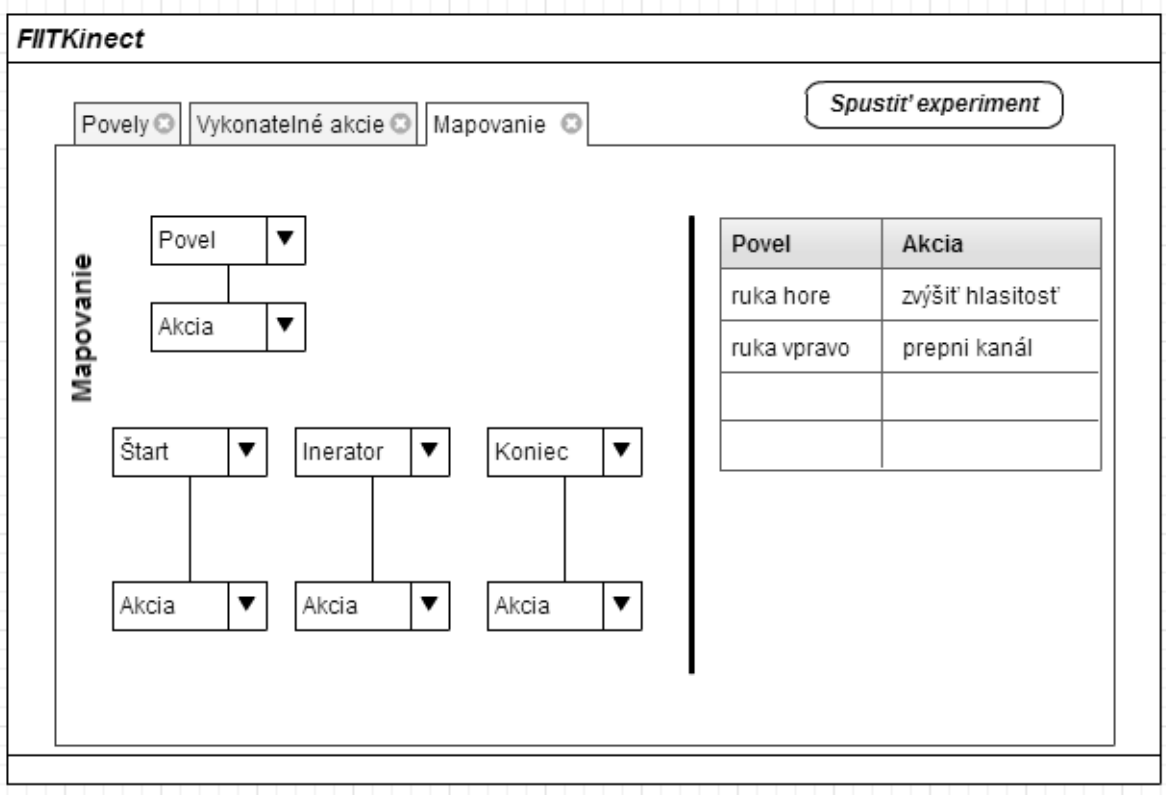


Obr. č. 11: Obrazovka s akciami.

5.1.1.1.5 Mapovanie

Náčrt obrazovky zachytáva rozmiestnenie prvkov pri mapovaní povelov a akcii. Obrazovka je oddeľovačom rozdelená na 2 časti a možnosti práce s mapovaním sú realizované na záložke – tabe mapovanie. V ľavej časti sa nachádzajú možnosti na priradenie povelov k akciám viacerými spôsobmi v závislosti na definovaní podmienok. Ak sa jedná o jednoznačnú dvojicu „povel-akcia“ v hornej časti si experimentátor kliknutím na možnosť výberu zvolí povel a následne akciu, ktorú k nemu požaduje priradiť. Po kliknutí oboch sa priradená dvojica bez potreby manuálneho ukladania automaticky uloží pre ďalšie použitie.

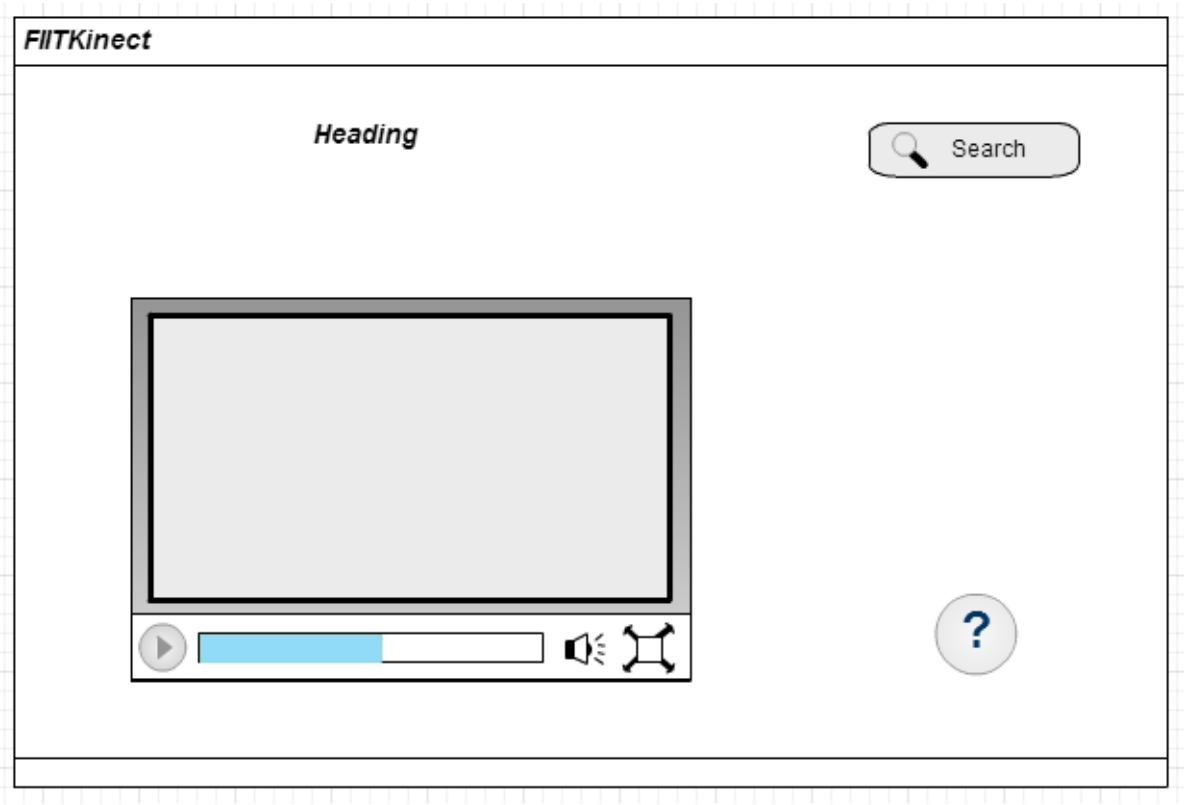
V dolnej ľavej časti sa nachádza 3 tlačidile na výber povelu – po kliknutí na štart sa nastaví spúšťacia akcia, po kliknutí a stredné tlačidlo iteračný povel a po kliknutí na koniec sa nastaví ukončovací povel. Pod týmto radom tlačidiel sa nachádzajú tlačidlá ktoré zodpovedajú povelom z pohľadu akcii a umožňujú aj podľa vizuálneho prepojenia namapovať začiatočnú, iteračnú a koncovú akciu. Priradenie povel - akcia je po zvolení automaticky uložená o čom je používateľ informovaný vizuálne.



Obr. č. 12: Obrazovka mapovania.

5.1.1.2 Aplikácia bežného používateľa

Rozhranie sa vyznačuje jednoduchosťou formy a minimalistickosťou. Používateľovi poskytuje informáciu o stave rozpoznávanie senzoru Kinect. Rovnako umožňuje spustenie krátkeho návodu – tutoriálu pre prípad potreby objasnenia spôsobu používanie aplikácie.



Obr. č. 13: Východzia obrazovka pre bežného užívateľa.

5.1.2 Database structure

5.1.2.1 Databases

5.1.2.1.1 MAIN_DB - select 0

```
experiments, online devices, expID(auto increment)
```

5.1.2.1.2 EVENT_DB - select 1

```
events, devices lists, active events lists for experiments
```

5.1.2.1.3 ACTION_DB - select 2

```
actions, devices lists, active actions lists for experiments
```

5.1.2.1.4 MAPPING_DB - select 3

```
mappings, mappings lists for experiments, mappingID(auto increment)
```

5.1.2.2 Main_DB

5.1.2.2.1 keys

```
expID - INCR
```

5.1.2.2.2 sets

```
experiments
["exp1", "exp2" ...]


onlineDevices_in
["dev1", "dev2" ...]


onlineDevices_out
["dev1", "dev2" ...]
```

5.1.2.2.3 hashes

```
{
   "expID" :
       {
           "create_date"          : "date"
           "last_modified_date"    : "date"
           "mapping_list_name"    : "exp1_mapping",
           "events_list_name"    : "exp1_events",
           "actions_list_name"    : "exp1_actions",
       }
}
```

5.1.2.3 Event_DB

5.1.2.3.1 hashes

```
{
     "eventID" :
       {
           "event"          : "eventID"
           "deviceID"         : "devID"
           "deviceType"     : "in/out"
       }
}
```

5.1.2.3.2 sets

```
devicesList
["dev1", "dev2" ...]

dev1_list
["eventID1", "eventID2" ...]
expID_list
["eventID1", "eventID2" ...]
```

5.1.2.4 Action_DB

5.1.2.4.1 hashes

```
{
     "actionID" :
       {
           "action"         : "actionID"
           "deviceID"        : "devID"
           "deviceType"    : "in/out"
           "IP"             : "127.0.0.1"
           "PORT"            : "456856"
           "val"            : "true"

       }
}
```

5.1.2.4.2 sets

```
devicesList
["dev1", "dev2" ...]


dev1_list
["actionID1", "actionID1" ...]


expID_list (set)
["actionID1", "actionID1" ...]
```

5.1.2.5 Mapping_DB

5.1.2.5.1 keys

```
mappingID - INCR
```

5.1.2.5.2 hashes

```
{
       "mappingID" :
           {
               "event"     : "event_name"
               "deviceIn"    : "devID"
               "action"    : "action_name"
               "deviceOut"    : "devID"
               "val"        : "15"
               "exp"        : "expID"
           }
}
```

5.1.2.5.3 sets

```
expID_mapping
["mappingID1", "mappingID2" ...]
```

5.1.3 Attributes

All attributes are required.

5.1.4 Examples

5.1.4.1 Event

```
{
    "ruka_hore" :
        {
            "event"          : "ruka_hore",
            "deviceID"       : "Kinect_cierny",
            "deviceType"     : "in",
            "link"           : "/multimedia/Kinect_videa/video05.mov"
        }
}
```

5.1.4.2 Action

```
{
    "volume-down" :
        {
            "action"         : "volume-down",
            "deviceID"       : "mac controller",
            "deviceType"     : "out",
            "val"            : "10",
            "IP"             : "127.0.0.1",
            "PORT"           : "4528",
            "val"            : "true"
        }
}
```

5.1.4.3 Mapping

```
{
    "4566" :
        {
            "event"          : "ruka-hore",
            "deviceIn"       : "Kinect_cierny",
            "action"         : "volume-down",
            "deviceOut"      : "mac controller",
            "val"            : "15",
            "exp"            : "28"
        }
}
```

6 Opis implementácie

Softvérový produkt je rozdelený na viacero častí.

6.1 Server – použité technológie

6.1.1 Node.js

- je platforma postavená na JavaScript runtime Chrome pre jednoduchú tvorbu rýchlych, škálovateľný sieťových aplikácií
- používa udalosťou riadený, neblokujúcej I/O model, ktorý ho robí ľahkým a výkonným
- ideálny pre dátovo náročné aplikácie využívajúce dáta v reálnom čase, ktoré sú spustené v distribuovaných zariadeniach
- poskytuje veľké množstvo modulov pre zjednodušenie práce

6.1.2 express.js

- pre tvorbu kódu je použitý framework express.js
- v hlavnom spúšťanom súbore *app.js* sú definované cesty k modulom (funkčne súvisiace celky), ktoré sú uložené v adresári *routes*
- definuje RESTful API funkcie poskytované serverom

6.1.3 redis

- pre pripojenie k databázovému serveru je využitý modul *redis*
- je to intuitívny ovládač na databázu poskytujúci všetky funkcie.

6.1.4 http

- pre sieťovú komunikáciu je využitý modul *http* s intuitívnymi funkciami

6.1.5 nodeunit, nodeunit-httpclient

- pre tvorbu unit testov je použitá knižnica *nodeunit-httpclient* pracujúca nad základným testovacím modulom *nodeunit*
- testy k jednotlivým modulom sú v adresári *test*

6.2 Inštalácia servera

6.2.1 Download link pre konkrétny OS

http://nodejs.org/download/

- postupovať ako pri bežnej inštalácii programov

6.2.2 Inštalácia modulov

- v príkazovom riadku/termináli nastaviť do projektového adresára
- inštalácia modulu lokálne pre projekt

  $ npm install modul_name

- inštalácia modulu lokálne pre projekt

  $ npm install –g modul_name

6.2.3 Spustenie servra

- v projektovom adresári spustiť
- *$ node názov_spúšťajúceho_súboru.js ($ node app.js)*

6.2.4 Spustenie unit testovania

- v projektovom adresári spustiť
- *$ nodeunit test*

6.3 Registrácia výstupného zariadenia na server

```
{
"deviceID": 'mac controller',
"deviceType": 'out',            // pre vstupne zariadenie je to 'in'
"IP": '127.0.0.1',
"PORT": '59596',
"actions":
    [
        {
            "volume-down": {"val": true}
        },
        {
            "vlc-stop": {"val": false}
        },
        {
            "vlc-next": {"val": false}
        },
        {
            "volume-up": {"val": true}
        },
        {
            "vlc-fullscreen": {"val": false}
        },
```

```
        {
            "system-right": {"val": false}
        },
        {
            "vlc-mute": {"val": false}
        },
        {
            "vlc-open-file": {"val": true}
        },
        {
            "vlc-previous": {"val": false}
        },
        {
            "system-left": {"val": false}
        },
        {
            "vlc-play-pause": {"val": false}
        }
    ]
}
```

7 Inštalácia potrebných nástrojov

Táto kapitola bola prebratá z analýzy architektúry Kinect tímu Art Quintet (č. 11) z akademického roku 2011/2012.

Sumarizuje na jedno miesto kroky, ktoré treba vykonať aby sa mohlo pokračovať vo vývoji s minimom počiatočných problémov. Používané vývojové prostredie bolo MS Visual Studio 2010 Ultimate. Niektoré kroky v návode môžu byť závislé na tejto skutočnosti. Nastavenia projektu VS 2010 vyžadujú existenciu premenných prostredia spomenutých v tomto návode.

7.1 Kompilácia QT

Aby sme mohli vyvíjať QT aplikáciu v prostredí Visual Studia treba postupovať podľa tohto návodu. Medzičasom možno existujú knižnice aj pre VS 2010, my sme iba pre 2008 narazili, preto bolo treba ručne skompilovať QT. Postupujeme podľa návodu na http://www.holoborodko.com/pavel/2011/02/01/how-to-compile-qt-4-7-with-visual-studio-2010/ s výnimkou/ modifikáciou týchto krokov:

- Krok 3: stiahneme verziu zdrojových kódov 4.7.4 (http://get.qt.nokia.com/qt/source/qt-everywhere-opensource-src-4.7.4.zip).
- Kroky 7 a 8 neaplikujeme (jom netreba).
- Namiesto kroku 10 vykonáme nasledovnú postupnosť príkazov:
    - cd c:\Qt\4.7.4
    - configure -debug-and-release -opensource -mp -platform win32-msvc2010
    - nmake

Dbajte aby ste mali nastavené rovnaké premenné prostredia ako spomínala web stránka, všetko to je nutné. Poznámka: toto nechajte buildovať na noc. Ten *nmake* tak dlho trvá.

7.2 OpenCV 2.3.0

Sprievodca kompiláciou OpenCV (v. 2.3.0) s použitím Threading Building Blocks v prostredí Visual Studio 2010.

1. Na stránke http://sourceforge.net/projects/opencvlibrary/files/opencv-win/ vyberieme požadovanú verziu OpenCV (v tomto návode inštalujeme verziu 2.3.0; inštalácia rôznych verzií sa môže mierne líšiť).
2. Stiahneme súbor so zdrojovými kódmi (OpenCV-2.3.0-win-src.zip) a rozbalíme ho do C:\OpenCV-2.3.0\.
3. Zo stránky http://threadingbuildingblocks.org stiahneme najnovšiu skompilovanú verziu Threading Building Blocks pre systém Windows.

4. Stiahnutý súbor (v tomto návode tbb40_233oss_win.zip) rozbalíme do adresára C:\OpenCV-2.3.0\build\common\tbb40_233oss.
5. Zo stránky http://www.cmake.org/ stiahneme najnovší inštalačný súbor CMake pre systém Windows.
6. Stiahnutý súbor (v tomto návode cmake-2.8.6-win32-x86.exe) nainštalujeme a otvoríme CMake GUI.
    a. Where is the source code -> Browse Soure: C:/OpenCV-2.3.0.
    b. Where to build the binaries -> Browse Build: C:/OpenCV-2.3.0/build/x86/vc10.
    c. Configure -> Visual Studio 10 -> Use default native compilers -> Finish.
    d. Zaškrtneme BUILD_EXAMPLES a WITH_TBB (prípadne iné) -> Configure.
    e. TBB_INCULDE_DIR nastavíme na C:/OpenCV-2.3.0/build/common/tbb40_233oss/include.
    f. Configure -> Configure -> Generate.
7. Otvoríme C:\OpenCV-2.3.0\build\x86\vc10\OpenCV.sln.
    a. Počkáme kým Visual Studio 2010 dokončí parsovanie (v stavovom riadku sa zobrazí Ready).
    b. Pravý klik na projekt -> Properties (ak nebude detegovaný nejaký hlavný projekt, Build All alebo niečo také, tak postupovať krokom c)
        i. C/C++ -> General -> Additional include directories: C:\OpenCV-2.3.0\build\common\tbb40_233oss\include
        ii. Linker-> General -> Additional library directories: C:\OpenCV-2.3.0\build\common\tbb40_233oss\lib\ia32\vc10
    c. (alternatíva za b.) Property Manager -> ALL_BUILD -> Debug Win32 -> Microsoft.Cpp.Win32.user -> VC++ Directories:
        i. Include directories: C:\OpenCV-2.3.0\build\common\tbb40_233oss \include
        ii. Library directories: C:\OpenCV-2.3.0\build\common\tbb40_233oss \lib\ia32\vc10
    d. Nastavíme rovnaké Additional include directories a Additional library directories aj pre release verziu.
8. Build -> Build Solution -> Coffee break...
9. Zmeníme typ buildu z Debug na Release.
10. Build -> Build Solution -> Coffee break...
11. Zavrieť VS, nastavenie premenných prostredia:

OPENCV (cesta k OpenCV)
príklad: C:\OpenCV-2.3.0

OPENCV_LIB (cesta ku knižniciam OpenCV)
príklad: C:\OpenCV-2.3.0\build\x86\vc10\lib*)

Pridať do **PATH** (cesta k adresárom bin\Debug a bin\Release OpenCV)
C:\OpenCV-2.3.0\build\x86\vc10\bin\Debug;C:\OpenCV-2.3.0\build\x86\vc10\bin\Release

TBB

príklad: C:\OpenCV-2.3.0\build\common\tbb40_233oss

Pridať do **PATH**
C:\OpenCV-2.3.0\build\common\tbb40_233oss\bin\ia32\vc10

12. Nakoniec do adresára C:\OpenCV-2.3.0\include\opencv2 nakopírujte obsah opencv2.zip (príloha- niekde v repozitári sa bude nachádzať). Dbajte aby ste toto spravili nakoniec, nie pred kompiláciou OpenCV.

7.3 Kinect SDK v1.0

Windows 7. Pre Kinect sme použili SDK od Microsoftu (v. 1.0), to vyžaduje operačný systém Windows 7 a vyššie. Inštalácia Kinect SDK v1.0:

Download z http://www.microsoft.com/en-us/Kinectforwindows/develop/overview.aspx
Postup: http://www.microsoft.com/en-us/Kinectforwindows/develop/release-notes.aspx
Vytvorenie premennej prostredia:

KINECTSDK10_DIR

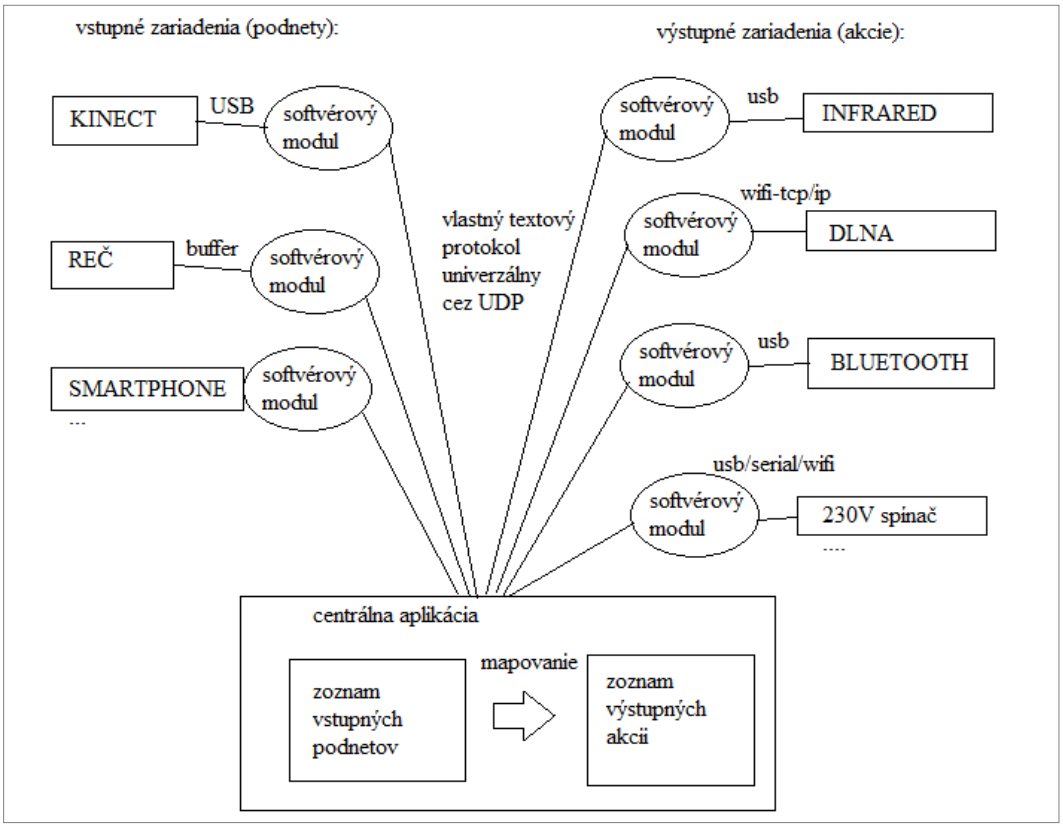
príklad : C:\Program Files\Microsoft SDKs\Kinect\v1.0\

Nastavenia projektu sú nastavené tak, aby fungovali u každého (nezávisle od umiestnenia potrebných knižníc a hlavičkových súborov). Preto si pri problémoch s kompiláciou (nenájdenie hlavičkových súborov, problémy s linkovaním) veľmi dôkladne skontrolujte nastavenie premenných prostredia (PATH a premenne pre knižnice Qt, Kinect SDK, OpenCV).

8 Celkový pohľad

Tento softvérový produkt bude spĺňať viaceré požiadavky na kvalitný softvér.

Návrh architektúry je možné vidieť na obrázku Obr. č. 14.



Obr. č. 14: Architektúra systému

Každé zariadenie bude mať implementované základné bezpečnostné prvky. Samozrejmosťou je logovanie činnosti systému. Autentifikácia a autorizácia užívateľov nebude na žiadosť klienta riešená. Aplikácia nebude obsahovať žiadne citlivé údaje, nie je preto nutná žiadna forma šifrovania.

Jedným z hlavných cieľov je rozšíriteľnosť produktu. Dodržiavaním stanovených konvencií je možné produkt nasadiť na neobmedzený počet počítačov za predpokladu dodržania minimálnych požiadaviek na hardvér.

Táto aplikácia je implementovaná vo viacerých programovacích jazykoch s využitím štandardizovaného protokolu na výmenu správ. Očakáva sa, že bude multiplatformová, primárnou platformou je aj naďalej Windows.

Pri tvorbe aplikácie sú používané metodiky písania a komentovania zdrojového kódu. Zdrojový kód je tiež pravidelne refaktorovaný tak, aby sa zvýšila jeho čitateľnosť a tiež znovupoužiteľnosť.

Architektúra bola navrhnutá tak, aby bola čo možno modulárna. Preto sa musí každý implementovaný modul riadiť metodikami uvedenými v dokumentácii riadenia tohto projektu. V jednotlivých moduloch sú použité modifikované návrhové vzory. Boli tam zavedené za účelom zvýšenia konzistencie. Názvoslovie premenných definuje metodika písania zdrojového kódu v relevantných jazykoch.

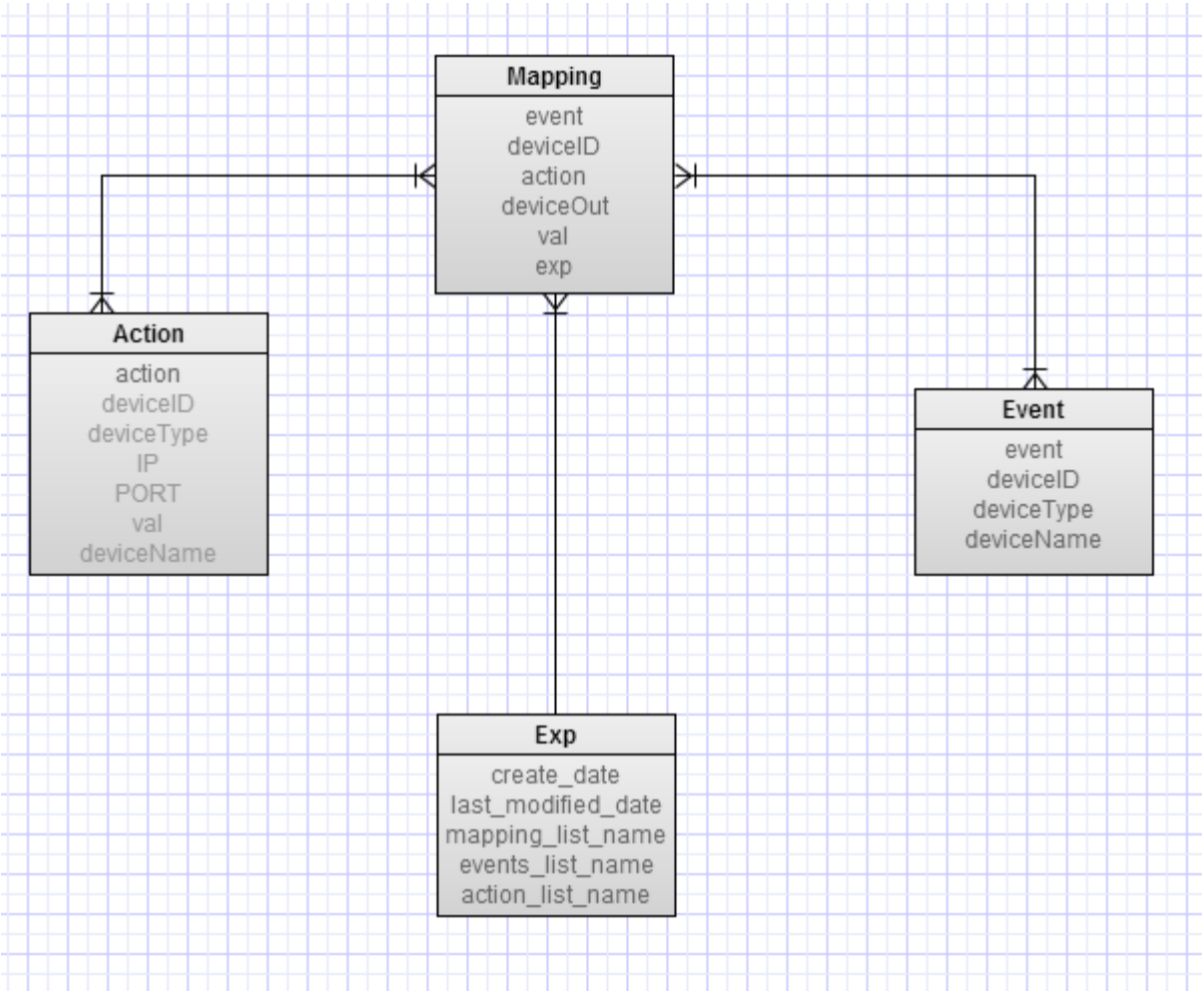
Softvérový projekt je stále v rannom štádiu vývoja. Nie je preto možné určiť, nakoľko bude finálny produkt spĺňať špecifikáciu. V tomto štádiu sa však dá povedať, že softvér implementuje funkcionalitu zhodnú s vytvorenou špecifikáciou.

9 Po druhom šprinte

Po uplynutí druhého šprintu, kde bol prvý kontrolný bod odovzdávania dokumentácie sme na základe ďalšieho vývoja produktu dopracovali a prepracovali produkt. V tejto časti dokumentácie sa nachádzajú materiály dopĺňajúce nami vyvíjané inžinierske dielo.

9.1 Model dát

Pri návrhu systému bola po analýza vybraná ako vhodná databáza Redis, ktorá predstavuje NO SQL prístup, v schéme na obr. 15 je pohľad na model dát, z ktorými je v systéme na úrovni databázy pracované. Vytvorený model poskytuje zjednodušení pohľad na databázový model systému.



Obr. č. 155: Dátový model

9.2 Android aplikácia

Súčasťou riešenie systému je aj samostatná aplikácia vyvinutá pre mobilné zariadenia používajúce platformu Android. Špecifikácia, aké funkcie má aplikácia poskytovať vychádzajú z celkových požiadaviek na aplikáciu a sú doplnené o špeciálne funkcionálne požiadavky.

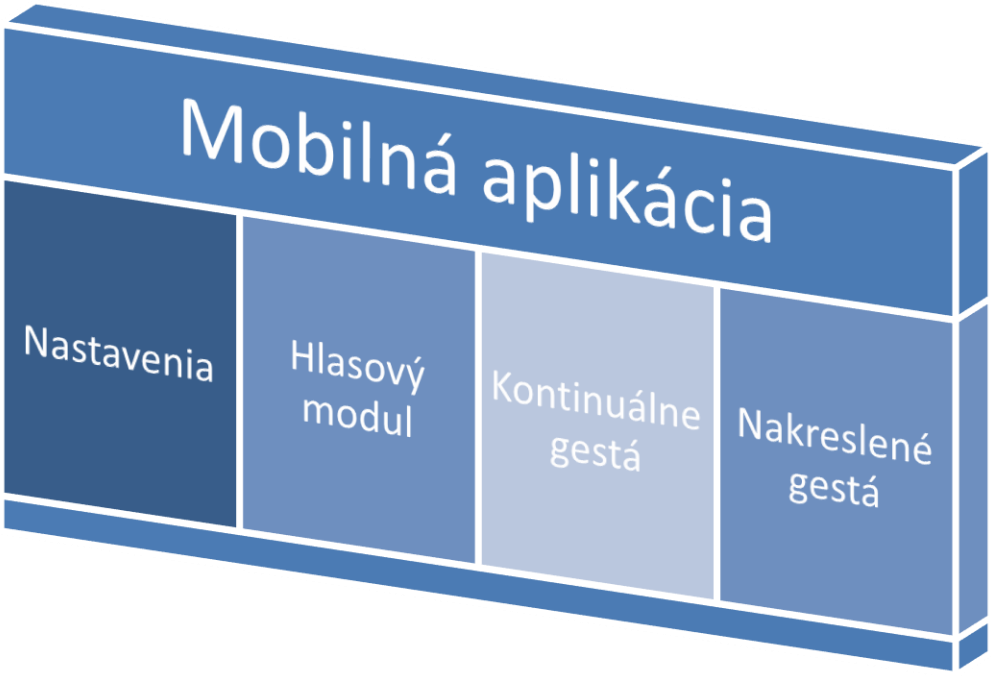
Požiadavky na aplikáciu:

- Možnosť definovania vlastných, používateľských gest
  - kontinuálne gestá (ovládanie)
  - jednoduché (lineárne) gestá
  - zložité, používateľom vytvárané (kreslené) gestá

- Možnosť zadávanie hlasových povelov
  - príjem hlasových povelov
  - rozpoznanie hlasových povelov
  - vyhodnotenie povelu
  - odoslanie eventu, ak nastal jeho spúšťač

- Poskytnutie možnosti na ovládanie aplikácie a spravovanie mapovania
  - zobrazenie definovaných gest
  - zobrazenie mapovania
  - zobrazenie dostupných akcií systému

9.2.1 Architektúra aplikácie



9.2.1.1 Popis jednotlivých modulov aplikácie

Nastavenia

Modul nastavenie poskytuje pre používateľa jednoduché rozhranie, kde je možné vykonávať mapovanie vstupných udalostí na výstupné akcie, ktoré sa majú vykonať. V tomto prostredí – záložke sú zobrazované aj informácie o dostupných vstupných a výstupných zariadeniach v systéme, ako ja všetky gestá, ktoré sú v zariadení definované. Táto časť je realizovaná ako webview integrované do prostredia natívnej Android aplikácie.

Hlasový modul

Hlasový modul umožňuje používateľovi v závislosti od dostupnosti a pripojiteľnosti do internetu rozoznávanie vysloveného slova v angličtine (offline režim), resp. v slovenčine (online režim). Rozpoznávanie slovenčiny nie je možné použiť v offline z dôvodu nedostupnosť binárnych dát potrebných na rozoznávanie slova, anglické výrazy sú priamo súčasťou aplikácie ako binárny súbor v definovanej štruktúre.

Kontinuálne gestá

Aplikácia rozpoznáva viacero typov používateľom realizovaných gest. Základným typom sú jednosuché gestá (napr. dotyk na obrazovke, presun – „slide“ po obrazovke), nadstavbou jednoduchých gest sú kontinuálne gestá, ktoré počas doby gest zaznamenávajú dotyk používateľa na obrazovke a následne komunikujú so serverom na zabezpečenie kontinuálneho ovládania.

Nakreslené gestá

Tento modul tvorí najzaujímavejšiu časť aplikácie. Používateľovi poskytuje možnosť rozpoznávať nim nakreslené gestá, ktoré sa môžu skladať z viacerých pohybov po obrazovke. Takýmto spôsobom je možné nechať používateľa experimentovať s ovládateľnosťou pomocou grafických obrazcov.

9.2.1.2 Ukážka kódu pre prácu s gestom

```
@Override

public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_DOWN){
            start = System.currentTimeMillis();
            x1 = event.getX();y1 = event.getY();
        }else if(event.getAction() == MotionEvent.ACTION_UP){
            if(((System.currentTimeMillis() - start) < clickTime) && (this.listener != null)) {
                listener.onTap();
            }else detectGesture(this.x1, this.y1, event.getX(), event.getY());
        }else if(event.getAction() == MotionEvent.ACTION_MOVE){
            if(detectGesture(this.x1, this.y1, event.getX(), event.getY())){
                x1 = event.getX();y1 = event.getY();
            }
        }
        return true;
    }

    private boolean detectGesture(float x1,float y1,float x2,float y2){
        if(Math.abs(x1 - x2) > distance){
            if((x1 < x2) && (this.listener != null)){
                this.listener.onRightSwipe();
                return true;
            }
            else if((x1 > x2) && (this.listener != null)){
                this.listener.onLeftSwipe();
                return true;
            }
        }
        if(Math.abs(y1 - y2) > distance){
            if((y1 < y2) && (this.listener != null)){
                listener.onBottomSwipe();
                return true;
            }
            else if((y1 > y2) && (this.listener != null)){
                listener.onTopSwipe();
                return true;
            }
        }
        return false;
    }
```

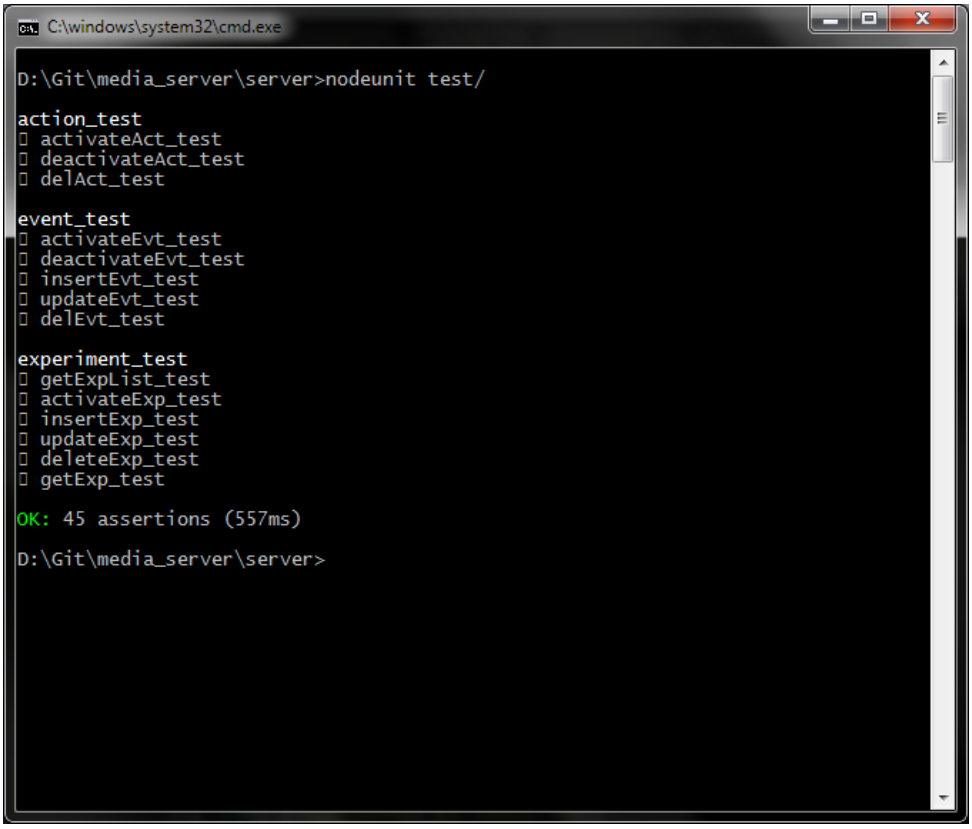
9.2.1.3 Ukážka kódu rozoznanie gest

```
@Override
    public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
        ArrayList<Prediction> predictions = FileHelper.getGestureLibrary().recognize(gesture);
        Collections.sort(predictions, COMPARATOR);
        Log.d(TAG, "count of predictions: "+ predictions.size());
        if(predictions.size() > 0){
            Prediction p = predictions.get(0);
            if(p.score > GESTURE_TRESHOLD){
                Log.d(TAG, "prediction score: " + p.score);
                Toast.makeText(this, p.name, Toast.LENGTH_SHORT).show();
                communicator.sendEvent(p.name);
            }
        }
    }
```

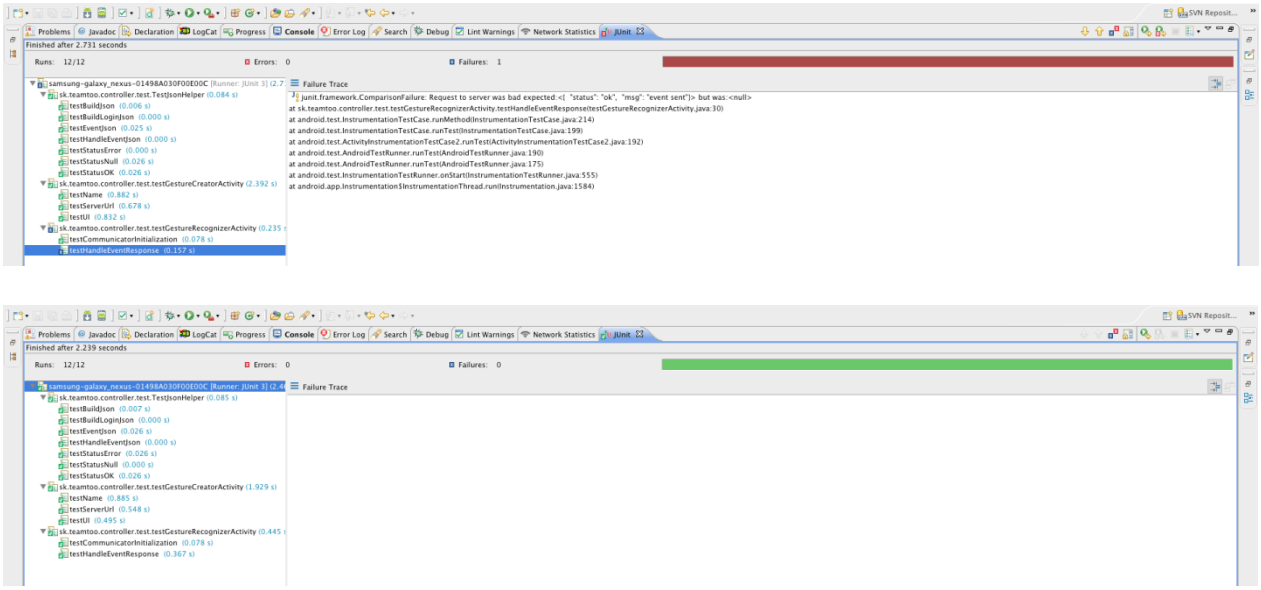
9.3 Testovanie aplikácie

9.3.1 Unit testy

Pri vývoji prebiehalo pri jednotlivých častiach testovanie kvality kódu. Pri testovaní sa využívali Unit testy, ktoré boli písané priamo s kódom, na ich základe bola overovaná funkčnosť daných programovaných častí.



Obr. č. 166: Unit test v Node.js –server



Obr. č. 177: Unit test – Android aplikácia

9.3.2 Akceptačné testy

Počas jednotlivých šprintov, kde sa vytvárala funkčná hodnotu produktu boli identifikované základné požiadavky na funkčnosť, na ktoré boli následne aplikované akceptačné testy pre danú funkcionalitu. Pri návrhu akceptačných testov sa bral ohľad predovšetkým na funkčnú stránku, pre akceptovanie daného výstupu boli overované v opodstatnených prípadoch aj ďalšie nefunkcionálne požiadavky na produktu – rýchlosť behu, používateľská prívetivosť, zdokumentovanosť danej odovzdávanej časti produktu.

9.3.2.1 Akceptačné testy po 1. odovzdaní (2. šprinte)

ID	01	**Názov**	Vytvorenie experimentu			
Prípad použitia		User story - Výskumník	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník / Experiment				
Účel		Overenie správne vytvorenie experimentu				
Vstupné podmienky		Existencia aspoň jedného eventu a aspoň jednej action				
Výstupné podmienky		Uložený nový experiment				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Vytvor experiment		Zobrazený formulár pre vytvorenie experimentu		Zobrazí sa formulár pre vytvorenie experimentu	
2	Zadanie názvu a stlačenie tlačidla „ok“		Experiment vytvorený		Experiment je viditeľný v paneli experimentov	

ID	02	**Názov**	Pridanie akcií do experimentu			
Prípad použitia		User story - Výskumník	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník / Action				
Účel		Overenie správneho pridávania action do experimentu				
Vstupné podmienky		Existencia aspoň jednej action				
Výstupné podmienky		Úspešné vloženie action do experimentu				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Pridaj akciu		Zobrazenie formuláru na pridávanie akcií		Akcia je viditeľná v hlavnom paneli	
2	Výber akcií		Označenie zvolených akcií		Akcia možno zvoliť v selecte akcii	
3	Zvolenie možnosti „ok“		Uloženie akcií		Táto vlastnosť nebola impelmentovaná	

ID	03	**Názov**	Pridanie eventov do experimentu			
Prípad použitia		User story - Výskumník	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník / Event				
Účel		Overenie správneho pridania event do experimentu				
Vstupné podmienky		Existencia aspoň jedného event				
Výstupné podmienky		Uloženie event do experimentu				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Pridaj eventu		Zobrazenie formuláru na pridávanie eventov		Event je viditeľný v hlavnom paneli	
2	Výber eventov		Označenie zvolených eventov		Event možno vybrať v select boxe eventov	
3	Zvolenie možnosti „ok“		Uloženie akcií		Nebolo implementované	

ID	04	**Názov**	Vytvorenie nového mapovania			
Prípad použitia		User story - Výskumník	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník / Mapovanie				
Účel		Overenie vytvárania nového mapovania				
Vstupné podmienky		Existencia aspoň jedného experimentu				
Výstupné podmienky		Uloženie nového mapovania				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Pridaj mapovanie		Zobrazí formulár na vytvorenie nového mapovania		Zobrazí sa okno z možnosťou mapovania	
2	Zvoľ event		Systém overí duplicitu a označí		Event je viditeľne označený	
3	Zvoľ akciu		Systém označí		Akcia je viditeľne označená	
4	Zvoľ „ok“		Systém uloží mapovanie		Zmena na stlačenie „mapuj“	

ID	05	**Názov**	Odobranie existujúceho mapovania			
Prípad použitia		User story - Výskumník	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník / Mapovanie				
Účel		Overenie odoberania existujúceho mapovania z experimentu				
Vstupné podmienky		Existencia aspoň jedného mapovania v rámci experimentu				
Výstupné podmienky		Odobranie existujúceho mapovania z mapovania				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Zvoľ mapovanie		Označí mapovanie		Označené mapovanie je vizuálne odlíšené od ostatných dostupných mapovaní	
2	Zvoľ odstrániť mapovanie		Odstráni mapovanie		Po stlačení „Odstráň“ nie je napovanie viditeľné v hlavnom paneli, ani databáze	

ID	06	**Názov**	Prepojenosť gesta (event) s akciou			
Prípad použitia		User story - Výskumník	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník / Gesto - Action				
Účel		Overenie prepojenia event – action v rámci experimentu				
Vstupné podmienky		Vytvorené aspoň jedno mapovanie				
Výstupné podmienky		Vykonanie action po zachytení event				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Prijatý event		Vykonaná akcia, ktorá bola na daný event namapovaná		Vykoná sa akcia, ktorá bola mapovaná na vstupný podnet - event	

ID	07	**Názov**	Zobrazenie pripojeného kinectu			
Prípad použitia		User story – Výskumník – Užívateľ	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník - Užívateľ / Kinect				
Účel		Overenie zobrazenia ikony pre aktívny Kinect				
Vstupné podmienky		-				
Výstupné podmienky		Zobrazená ikona				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Pripojený Kinect		Zobrazenie príslušnej ikony		Kinect je viditeľný ako aktívne zariadenie v hlavnom paneli	

ID	08	**Názov**	Zobrazenie gesta (event)			
Prípad použitia		User story – Výskumník – Užívateľ	**Úroveň splnenia testu**		**Autor**	Michal Igaz
Rozhranie		IS / Výskumník - Užívateľ/ Event				
Účel		Overenie zobrazovania event				
Vstupné podmienky		Existencia aspoň jedného eventu				
Výstupné podmienky		Event zobrazený/prehraný				
Krok	**Akcia**		**Očakávaná reakcia**		**Skutočná reakcia**	
1	Označ event		Systém označí event		Po označení je graficky rozlíšený event v ovládacom rozhraní	
2	Zobraz zvolený event		Systém zobrazí zvolený event		Vizuálne je zobrazené gesto patriace k eventu	

9.3.2.2 Akceptačné testy pred koncom 4. Šprintu

Po druhom šprinte sa zmenila forma akceptačných testov, okrem overovania priamej funkcionality sa kládol väčší dôraz aj na vizuálne prevedenie častí, ktoré zobrazujú používateľovi informácie ako aj na prenos informácií medzi zariadeniami a serverom. Časť toho testovania bola obsiahnuté v čiastkových *unit testoch* jednotlivých komunikujúcich modulov.
V tejto fáze bol najviac kladený dôraz na Android aplikáciu.

Úloha	**Predpokladané správanie**	**Skutočnosť**	**Spôsob overenia**
Zobrazenie GUI	Vytvorí sa grafické rozhranie	Po aktivovaní aplikácie sa zobrazilo jej grafické rozhranie	Vizuálne, unit testom – inicializácia GUI elementu
Načítanie informácií do webview	Zobrazí sa záložka z parametrami nastavovania	Po kliknutí na záložku bolo zobrazené rozhranie pre ovládanie mapovania, boli zobrazené eventy, akcie	Vizuálne
Overenie komunikácie so serverom - pripojenosť	Aplikácia komunikuje so serverom prostredníctvom siete	Aplikácia komunikuje so serverom, je možné nastavovať parametre, je viditeľná očakávaná reakcia, po vykonaní eventu sa vykoná namapovaná akcia	Vizuálne – načítavanie informácií zo servera Wireshark – zobrazenie sieťovej komunikácie
Overenie rozpoznania gesta	Po vykonaní nastaveného gesta sa rozpozná jeho názov	Gesto je rozpoznané, o čom svedčí výpis priamo v aplikácií, pri vizuálne podobných gestách je zaznamenané v niektorých prípadoch nesprávne rozoznanie	Viacnásobné nakreslenie gest v rôznych obmenách
Overenie hlasového povelu	Po aktivovaní rozoznávania povelu a rozoznaní nadefinovaného gesta sa zobrazí v textovej informácií rozpoznané slovo	Hlasový povel je rozpoznaní, v móde „slovenčina“ pri slovách z interpunkciou je zaznamenané rozdielne vyhodnocovanie v závislosti od slova	Vizuálne, akusticky

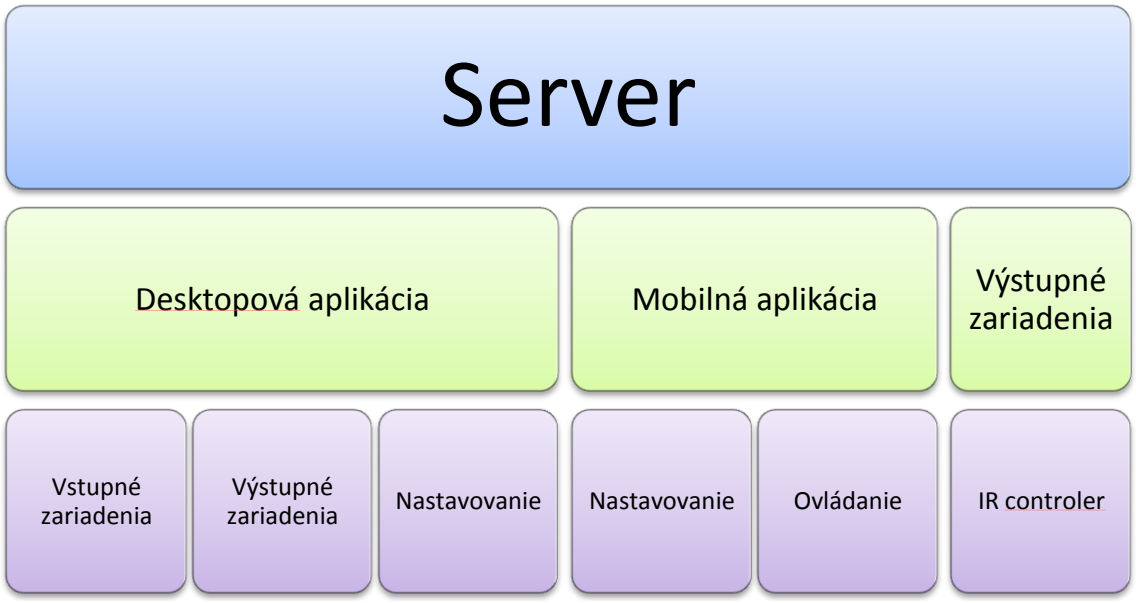
9.4 Architektúra systému

Pri pohľad na systém sú rozlíšiteľné jednotlivé jeho časti. Nosnou časťou je serverová aplikácia poskytujúce výstupným a výstupným zariadeniam v systéme potrebnú vzájomnú konektivitu.
Zariadenie sa pripájajú na server štandardizovaným TCP/IP protokolom, v riešení boli oproti pôvodnému pridané *Server sent events*.
Vzhľadom na požiadavku modulárnosti systému je vytvorené RESTful API poskytujúce možnosti rozšírenia pre akékoľvek zariadení schopné pracovať s danými technológiami. Systém poskytuje možnosť práce prostredníctvom desktopovej aplikácie, mobilnej aplikácie, výstupných zariadení.

9.4.1.1 Server sent events

Server sent events predstavujú štandard pre komunikáciu, pri ktorej server inicializuje prvý prenos dát smerom ku klientovi čím vytvorí spojenie s ním. Typicky sa tento štandard využíva pri kontinuálnom dátovom streamovaní ku klientovi.

9.4.2 Pohľad na systém ako celok



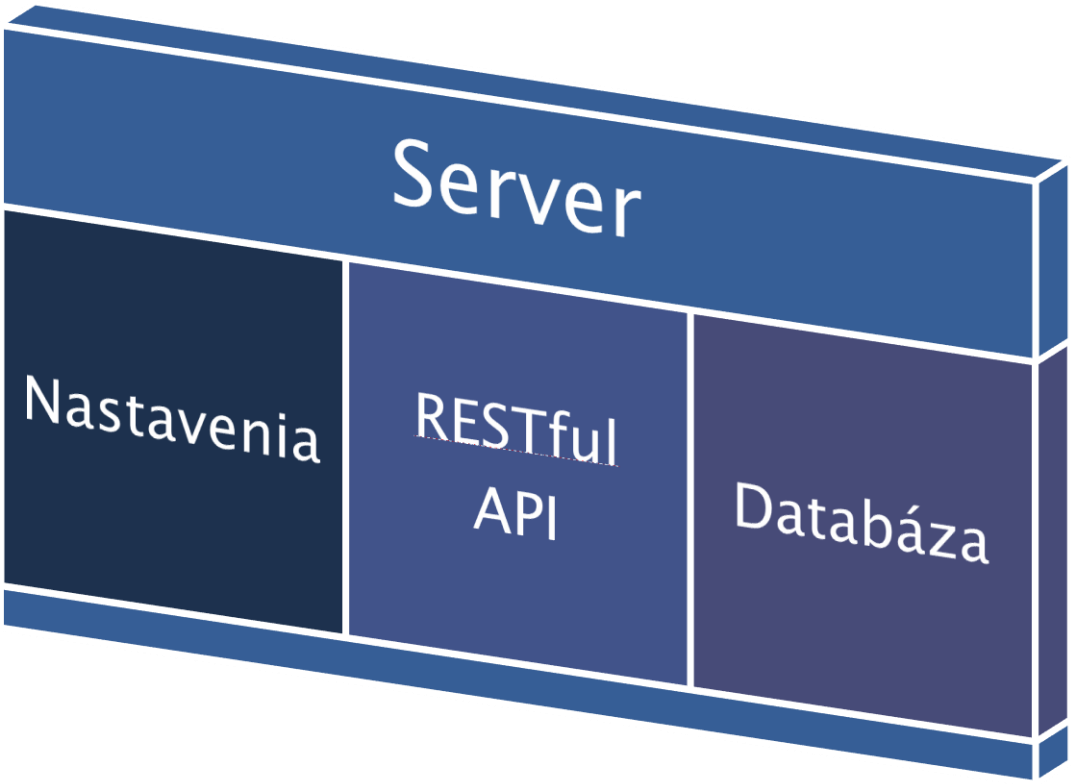
Obr. č. 188: Architektúra systému

9.4.3 Pohľad na desktopovú aplikáciu

Obr. č. 199: Desktopová aplikácia - architektúra

Desktopová aplikácia je realizovaná v jazyku C++. Používa časť z minuloročného projektu [1.6], ktorá zabezpečuje prácu so senzorom Kinect. Vzhľadom na prídavné moduly v systéme je do hlavnej aplikácie zahnutý aj modul na ovládanie IR zariadenia, prácu s rozpoznávaním hlasu. Nastavovanie celého systému v podobe mapovaní vstupných a výstupných udalostí sa realizuje v nastavovacom rozhraní, ktoré je vložené ako webview v desktopovej aplikácií.

9.4.4 Pohľad na server



Obr. č. 20: Desktopová aplikácia - architektúra

Serverová aplikácia je postavená nad jazykom Node.js, ktorý umožňuje konektivitu pre mnohonásobne viac pripojených zariadení, ako iný typ servera. Server poskytuje RESTful API pre pripojenie ďalších modulov pomocou POST, GET, čím poskytuje štandardizovaný prístup pre komunikáciu.

10 Pridanie funkcionality multipodnetového ovládania

10.1 Uloženie mapovania

Mapovanie sa ukladá ako array eventov a jedna výstupna akcia. Eventy v novom mapovaní nemôžu byť podpostupnosti s iným mapovanim.

napr.

```
['ruka hore']

['ruka hore', 'ruka dole']
```

V takomto prípade sa druhé mapovanie nikdy nevykoná. Algoritmus spracovania eventu:

1. Prijme sa event 'ruka hore'
2. Zruší sa aktívny časovač (ak je nejaký nastavený)
3. Skontroluje sa príznak na prefixové mapovanie
4. Ak nie je:
5. Prijatý event sa vloží do arrayu čakajúcich eventov
6. Array čakajúcich eventov sa porovnáva s mapovaniami
7. Ak sa arraye rovnajú, vykoná sa akcia a array čakajúcich eventov sa vymaže (naspat na 1)
8. Ak sa nerovnajú, spustí sa časovač na príjem ďalšieho eventu
9. Ak uplynul čas v časovači, array čakajúcich eventov sa vymaže (krok 1)

10.2 Prefixové mapovanie

Prefixové mapovanie je tvorené trojicov eventov: spušťač, vykonávač a ukončovač + je pridaný príznak pri vytvoren nového eventu

napr.

```
['ruka hore', 'ruka dole', 'swipe left']
```

Uprava algoritmu

1. Ak si príjme event 'ruka hore' a pri porovnaní arrayov sa zistí, že týmto eventom začína prefixové mapovanie, tak sa žiadny časovač nespustí
2. Spustí sa globálny príznak v express.js že sa používa prefixové mapovanie (prípadne sa odloží cela trojica mapovania aby sme nemuseli znovu hľadať)
3. Pri prijati dalsieho eventu sa skontroluje mapovamie:
4. Ak je event vykonavač, tak sa vykoná
5. Ak je event ukončovač, tak sa vymaže globálny príznak prefixového mapovania a vyprázdni array čakajúcich eventov (naspat na 1)

11 Zoznam obrázkov