Slovenská technická univerzita

Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Odhaľovanie emocionálneho stavu používateľa

Team EmLog

Dokumentácia k inžinierskemu dielu

Tímový projekt 2012/2013

Vedúca tímu:
Doc. Mgr. Daniela Chudá, PhD.

Členovia tímu:
Bc. Jozef Gajdoš
Bc. Martin Geier
Bc. Peter Greguš
Bc. Miroslav Hudák
Bc. Peter Sivák
Bc. Peter Šinský

Kontakt: team.10.tp@gmail.com

1 OBSAH

2 ÚVOD

2.1 ÚČEL DOKUMENTU

Tento dokument vznikol na predmete tímový projekt ako dokumentáciou k projektu pre zisťovanie emocionálneho stavu používateľa.

2.2 ŠTRUKTÚRA DOKUMENTU

Štruktúra dokumentu je prispôsobená vývoju metódou *Scrum.* V každej kapitole je zdokumentovaný jeden šprint. V jednotlivých podkapitolách sú opísané činnosti, ktoré sa v danom šprinte vypracovávali.

2.3 ZADANIE PROJEKTU

Každý používateľ počas práce s počítačom zažíva momenty, v ktorých sa cíti rôzne. Keď sa mu podarí dokončiť projekt, žiari šťastím, naopak, keď stratí neuloženú prácu v dôsledku chyby programu, počítač by najradšej rozbil. Emocionálny stav používateľa však môže poskytnúť spätnú väzbu aj k tomu, čím sa práve zaoberá, napríklad pri prezeraní časti zdrojového kódu softvérového projektu, ktorú vytvoril jeho kolega. Ak je zase používateľ unavený a frustrovaný pri vytváraní zdrojového kódu, je pravdepodobné, že tento kód bude obsahovať chyby. Vtedy mu môžeme odporučiť krátku prestávku.

Úlohou tímov bude vytvorenie modelu pre reprezentáciu emócií používateľa a následné zachytávanie emocionálneho stavu používateľa do tohto modelu. Pri tom sa jeden tím zameria na prácu používateľa s bežnými vstupnými zariadeniami, klávesnicou a myšou (ako rýchlo používateľ píše, ako často a akým spôsobom sa mýli, ...), druhý tím využije obraz používateľa získavaný webovou kamerou (výraz tváre, smer pohľadu, ...). Cieľom je vytvorenie aplikačného rozhrania umožňujúceho využitie rozpoznaného stavu používateľa v rôznych projektoch, napríklad vo vývojom prostredí v rámci projektu podpory vývoja softvéru v prostredí firmy, ako i overenie a porovnanie vytvorených riešení v rôznych situáciách: odporúčanie prestávok, dôvera v hodnotenie poskytnuté používateľom a pod.

3 ŠPRINT 0 – ANTÓNIA

3.1 ZAZNAMENÁVANIE AKTIVITY POUŽÍVATEĽA

Zodpovedná osoba: Martin Geier

Táto kapitola opisuje možné alternatívy k programu *PerConIK*. Hlavnými požiadavkami boli množstvo funkcií ktoré by sme mohli uplatniť pri analýze emocionálneho stavu používateľa a licencia umožňujúca úpravu zdrojových kódov s ich následnou distribúciou.

Auto clicker typer:

- Poskytuje funkcionalitu na zachytávanie obrazovky s možnosťou následného spustenia ako video,
- zaznamenáva:
  - klávesnicu,
  - myš,
  - obrazovku,
- licencia – uzavretý zdrojový kód.

KidLogger:

- skladá sa z aplikácie na lokálnej stanici a serverovej časti,
- lokálna aplikácia odosiela nazbierané údaje na server, na ktorom je možné prehliadať aktivity a štatistiky,
- zaznamenáva:
  - klávesnicu,
  - myš,
  - obrazovku,
  - nahrávanie hlasu,
  - komunikačné nástroje – *icq, skype,*
  - spustené aplikácie,
- licencia – zverejnený zdrojový kód klientskej aplikácie pre Windows na účely štúdia či program nerobí inú činnosť ako má deklarovanú,
- programovací jazyk – *C++*.

OsdHotkey:

- jednoduchá aplikácia slúžiaca ako *keylogger*,
- zaznamenáva:
  - klávesnicu,
  - myš,
- licencia – *open source*,

- programovací jazyk – *AutoHotkey language*.

PyKeylogger:

- aplikácia slúžiaca prevažne ako *keylogger*,
- zaznamenáva:
  - klávesnicu,
  - pozíciu myši pri kliknutí,
  - obrazovku.
- licencia – *open source*,
- programovací jazyk – *Python*.

3.2 MOŽNÉ SMERY ODPORÚČANÍ

Zodpovedná osoba: Peter Greguš

Smery odporúčaní pre používateľa môžeme rozdeliť do dvoch kategórií a to okamžité a odporúčanie na základe dlhodobého pozorovania.

Do okamžitých môžeme zaradiť odporúčanie prestávok, zmena farebnej schémy na počítači, zmeniť „zameranie“ (ísť na chvíľu navštíviť sociálnu sieť a pod.), zahrať si krátku hru a iné.

Pri odporúčaní na základe dlhodobého pozorovania sa najskôr zistí štatistika používateľa napríklad akú hudbu počúva pri danej emócií a následne sa vytvorí *playlist*. Potom sa budú odporúčať tie piesne, pri ktorých mal používateľ dobrú náladu. Ďalej je možné vytvárať štatistiku v kombinácií z množstvom vykonanej práce. Na základe výstupného grafu bolo vidno, kto koľko vypracoval pri akom emočnom stave. Z toho môže šéf odporúčať svojim podriadeným dovolenku, prípadne inú formu motivácie.

4 Šprint 1 – Frederika

4.1 Analýza programu *PerConIK*

Zodpovedná osoba: Miroslav Hudák

Cieľom aplikácie *PerConIK* je zaznamenanie aktivity používateľa za účelom analýzy správania sa. Výsledkom tejto analýzy je model používateľa, ktorý opisuje jeho charakteristické črty správania sa v čase.

Udalosť je činnosť používateľa na *GUI*, ktorá vyvolá nejaký proces. Delí sa na:

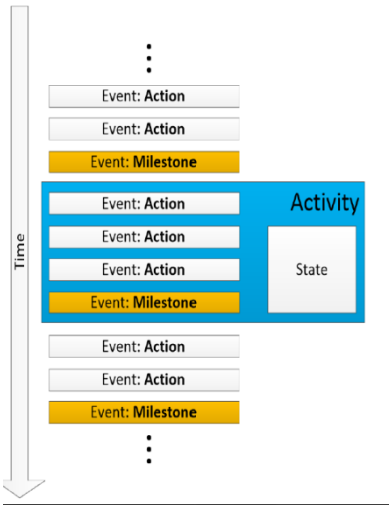
- *akciu* – predstavuje bežnú činnosť v pracovnej aktivite používateľa,
- *míľnik* – významná – spúšťa zber a odosielanie údajov o aktivite.

Udalosti vzťahujúce sa na činnosť používateľa sa zaznamenávajú podľa možností chronologicky tak, ako nastali. Udalosťou môže byť spustenie *buildu* v *IDE*, prepnutie stránky vo *OneNote*, prepnutie aktívnej aplikácie a iné.

Aktivita používateľa je postupnosť udalostí (akcií) zakončená významnou udalosťou (míľnikom). Každá aktivita obsahuje práve jeden míľnik.

Po ukončení aktivity používateľa sa vytvorí balík údajov, ktorý sa skladá z:

- akcií,
- míľnika,
- statických údajov – biometria, stav aplikácií, využitie *HW* prostriedkov.



Obrázok 1 - Aktivita obsahujúca zoznam akcií, míľnik a stav

Zdroje informácií:

- *MS Visual Studio 2010* – sledovanie kontextu programovania a zostavovania programov v *IDE* vo forme rozšírenia,
- *MS Office 2010 OneNote* – aktivity v kontexte otvorených poznámkových blokov cez *COM* rozhranie,
- biometria – biometrické údaje použitím klávesnice a myši,
- *Application Activity* – stav spustených aplikácií a okien,
- *HW* prostriedky – sledovanie procesora, využitie pamäte cez rozhranie *Win32 API*,
- *MS Outlook a Lync 2010* – vytváranie a čítanie elektronickej komunikácie vo forme rozšírenia,
- rozšírenie webového prehliadača – načítané webové zdroje.

Pre účely tohto projektu sú najpodstatnejšie biometrické údaje.

4.1.1 Biometria

Biometrické údaje sa získavajú sledovaním použitia zariadení ako je klávesnica a myš. Údaje o používaní týchto prostriedkov sa získavajú prostredníctvom *Win32 API*.

Nasledujúci obrázok zobrazuje sledované údaje a možnosti ich získania.

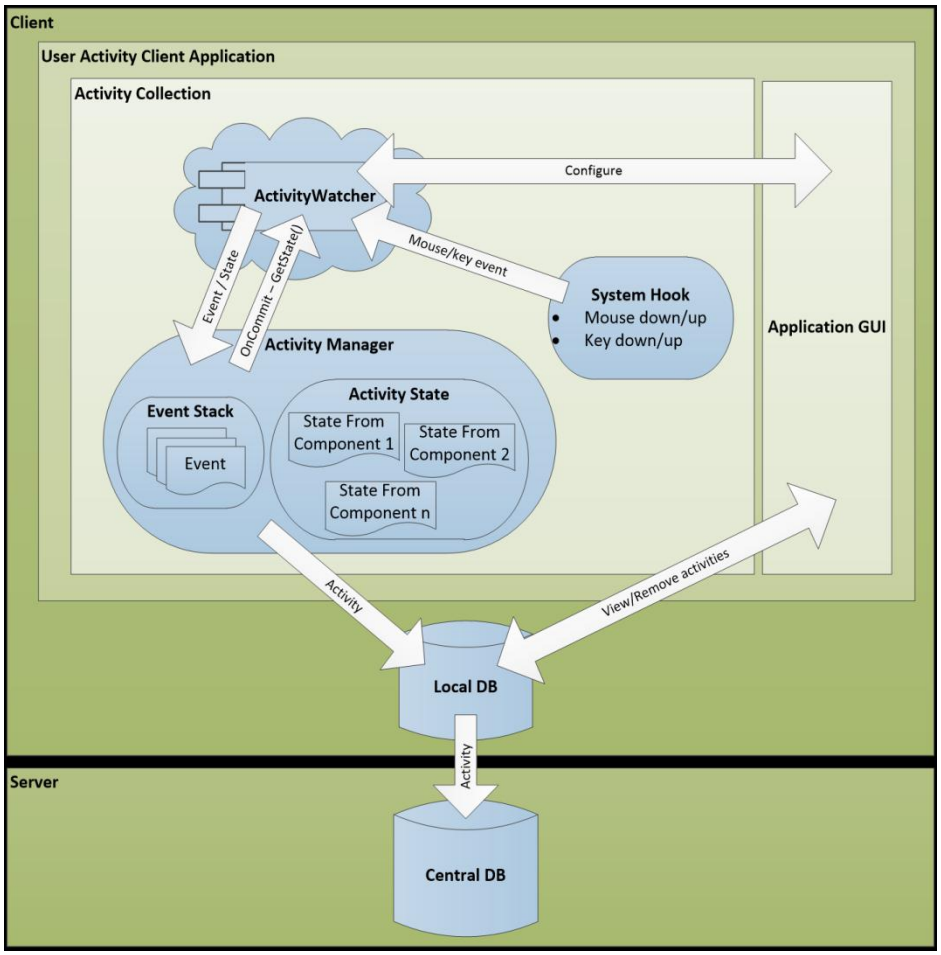
Údaj	Funkcia	Typ	Spôsob získania
Myš...			
Poloha -> intenzita pohybu	WM* mouse-events		event /suma za interval
Klik -> intenzita klikania	WM* mouse-events		event /suma za interval
Scroll -> intenzita pouzitia	WM* mouse-events		event /suma za interval
Klávesnica...			
Tlačidlo dole	WM,WK* keyb.events		event
Tlačidlo hore	WM,WK* keyb.events		event
• **Historia stlacenych klavesov,** • **Intenzita stlacania klaves** • **Intervaly medzi stlacenim vybranych klaves**			postupnost klavesov s casovou znackou, alebo len suma za interval

Obrázok 2 - Sledované Biometrické údaje

4.1.2 Architektúra

Zber aktivít jedného používateľa riadi na jeho klientskom počítači aplikácia *User Activity Client Application* (*UACA*, v jednoduchosti *logger*), ktorá beží v jeho *system tray*. Aktivity sa

ukladajú na strane klienta do lokálneho úložiska *LocalDB*, odkiaľ sú postupne prenášané do centrálneho úložiska na serveri *CentralDB* (pozri obrázok).



OBRÁZOK 3 - ARCHITEKTÚRA SYSTÉMU *PERCONIK*

Aplikácia *UACA* obsahuje komponenty (*ActivityWatcher*) zodpovedné za notifikáciu udalostí keď nastanú. *ActivityWatcher* notifikuje o udalosti (akcii alebo míľniku) *ActivityManager*. Ten vloží udalosť do zásobníka udalostí – *EventStack*. V zásobníku sú len udalosti pre aktuálnu aktivitu. Ak je notifikovaný míľnik, došlo k zakončeniu aktivity. V tom prípade sa od každého komponentu *ActivityWathcer* vyžiada informácia o stave, ktorých množina predstavuje celkový stav aktivity. Usporiadaný zoznam udalostí spolu s celkovým stavom aktivity sú poslané do lokálneho úložiska (*LocalDB*). Zásobník udalostí *EventStack* sa vyprázdni a začína konštrukcia novej aktivity.

4.1.3 ZBER BIOMETRICKÝCH ÚDAJOV

Zber údajov z klávesnice a myši má na starosti modul *BiometryWatcher*. Je rozdelený na dva komponenty:

- *KeyboardStateWatcher* – klávesnica,
- *MouseStateWatcher* – myš.

Modul sleduje generované zmeny a tie následne posúva na konkrétny komponent. Údaje sú poskytované vo forme udalostí:

- *MouseChanged* (*Point*, *Message*, *Delta*),
  - *Point* – pozícia myši (x, y),
  - *Message* – typ akcie (*Win32.MouseMessages*),
  - *Delta* – hodnota otočenia kolieska,
- *KeyboardChanged*(*Message*, *Key*),
  - *Message* – typ akcie (*Win32.KeyboardMessages*),
  - *Key* – hodnota stlačenej klávesy (*System.Windows.Input.Key*).

Modul zbiera údaje počas kompletnej aktivity používateľa. Odoslanie údajov nastane, keď je identifikovaný míľnik. Samotný modul negeneruje míľnik, preto sa spolieha na iné moduly. Zozbierané dáta sa posielajú do databázy vo formáte *BLOB*.

KeyboardStateWatcher

Tento komponent zachytáva údaje generované klávesnicou a ukladá ich do výstupnej štruktúry *KeyboardStateBlobDto*. Táto trieda reprezentuje zoznamy latencií (List<*KeyboardGraphDto*>) pre grafy (znaky), digrafy (dvojice znakov), trigrafy (trojice znakov) a klávesové skratky. Trieda *KeyboardGraphDto* obsahuje trojicu údajov:

- *Character* – hodnota stlačených znakov,
- *Latency* – latencia stlačenia,
- *FlightTime* – interval medzi stlačením rôznych kláves.

Pre každý stlačený znak na klávesnici sa vygenerujú dve udalosti, ktoré reprezentujú čas stlačenia a čas uvoľnenia klávesy. Pomocou týchto údajov sa vypočítajú latencie a čas letu.

Komponent sleduje len tie najfrekventovanejšie kombinácie kláves, identifikované na základe frekvenčnej analýzy anglického a slovenského jazyka, resp. najpoužívanejších klávesových skratiek.

MouseStateWatcher

Tento komponent zachytáva údaje generované myšou a ukladá ich do výstupnej štruktúry *MouseStateBlobDto*. Táto trieda reprezentuje zoznamy špecifických akcií myši:

- zoznam pohybov,
- zoznam akcií *Drag&Drop*,
- zoznam pohybov kolieska myši,
- zoznam všetkých typov stlačenia tlačidiel myši.

Medzi sledované údaje patria: začiatočná pozícia, koncová pozícia, čas pohybu, prejdená vzdialenosť, typ stlačeného tlačidla, hodnota otočenia kolieska.

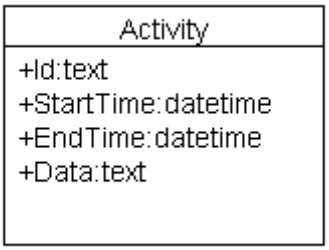
Aby sa eliminovali náhodné pohyby, odosielajú sa len pohyby, ktoré spĺňajú základné kritéria ako minimálna dĺžka pohybu a maximálne prerušenie pohybu.

4.2 DÁTOVÝ MODEL PROGRAMU *PERCONIK*

Zodpovedná osoba: Jozef Gajdoš

4.2.1 KLIENTSKÁ ČASŤ APLIKÁCIE

Klientská časť aplikácie používa *SQLite* databázu verzia 3, v ktorej má uchovanú iba jednu tabuľku s názvom *Activity*. Jej schéma je na nasledovnom obrázku. Entita *Activity* reprezentuje aktivitu používateľa ktorá bola sledovaná medzi dvoma míľnikmi.

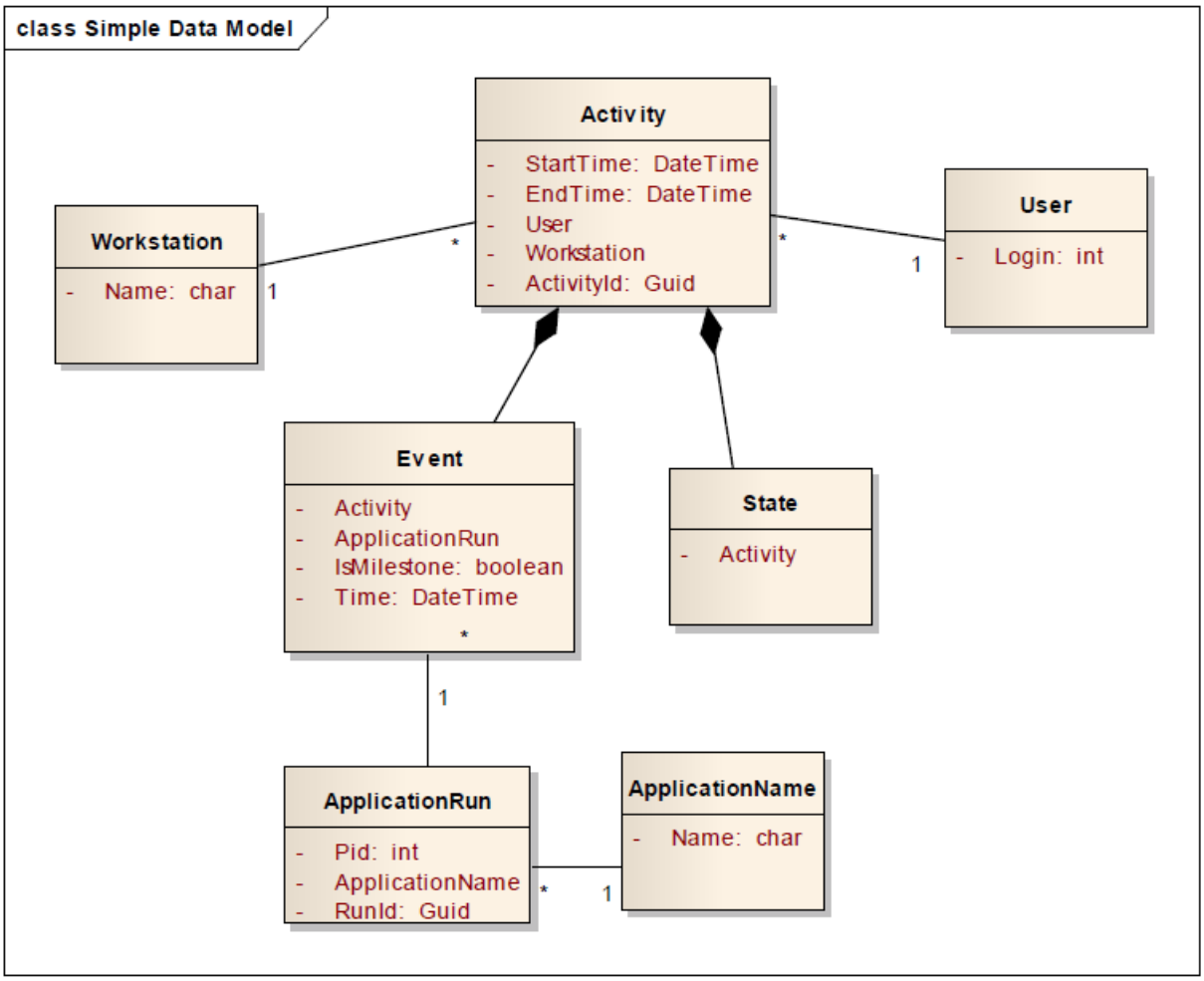


OBRÁZOK 4 - DÁTOVÝ MODEL LOKÁLNEJ DATABÁZY

Id aktivity je jedinečný text generovaný pre každú aktivitu zvlášť (*guid*). Atribúty *StartTime* a *EndTime* reprezentujú začiatok a koniec aktivity. Atribút *Data* obsahuje serializované objekty reprezentujúce aktivitu používateľa. Sú uložené vo forme *XML*.

4.2.2 SERVEROVÁ ČASŤ APLIKÁCIE

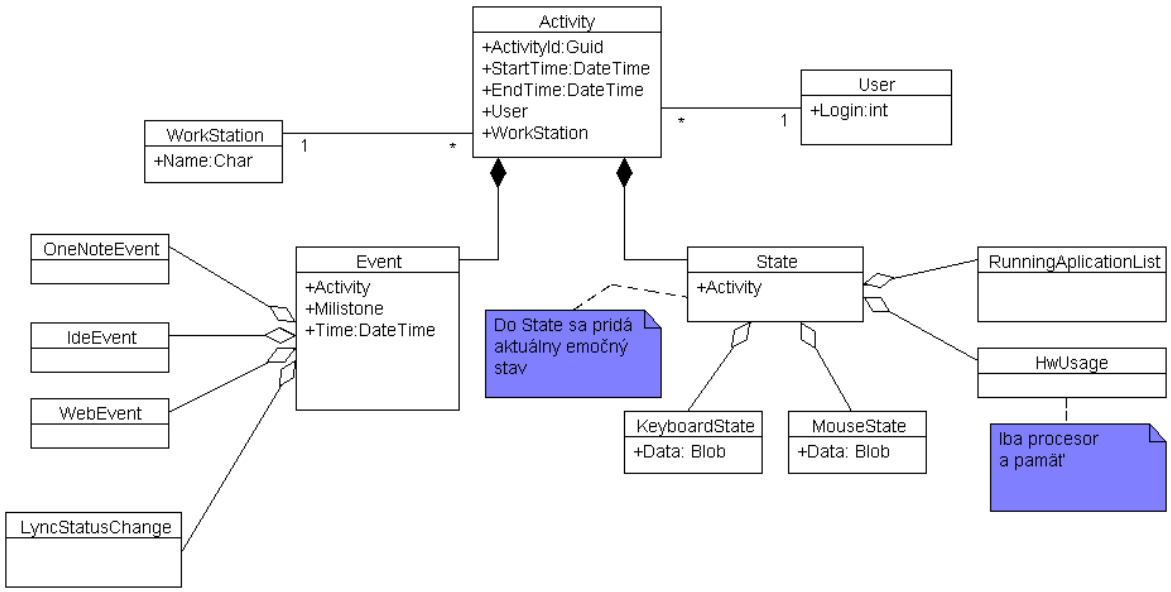
Obrázok 5 znázorňuje logický model serverovej časti aplikácie, ktorá používa *MS SQL* server.



OBRÁZOK 5 – LOGICKÝ MODEL DATABÁZY *PERKONIK*

Ústrednou entitou je Aktivita, ktorá uchováva udalosti medzi dvoma míľnikmi a stav sledovaného systému na konci aktivity. Každá udalosť prislúcha k behu konkrétnej aplikácie.

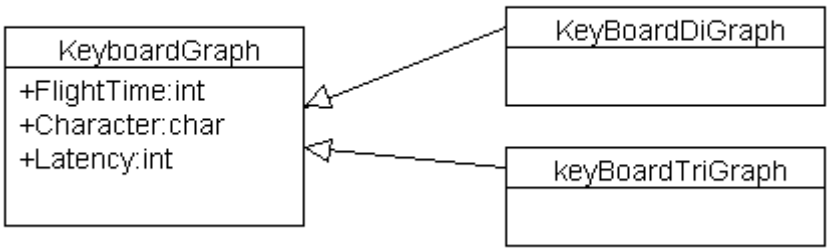
Na ďalšom obrázku je znázornený logický model serverovej časti aplikácie ktorý abstrahuje od detailov *Eventu* (stavu aplikácií) a zameriava sa na spôsob ukladania klávesnice a myši.



OBRÁZOK 6 – DETAILNÝ LOGICKY MODEL DATABÁZY

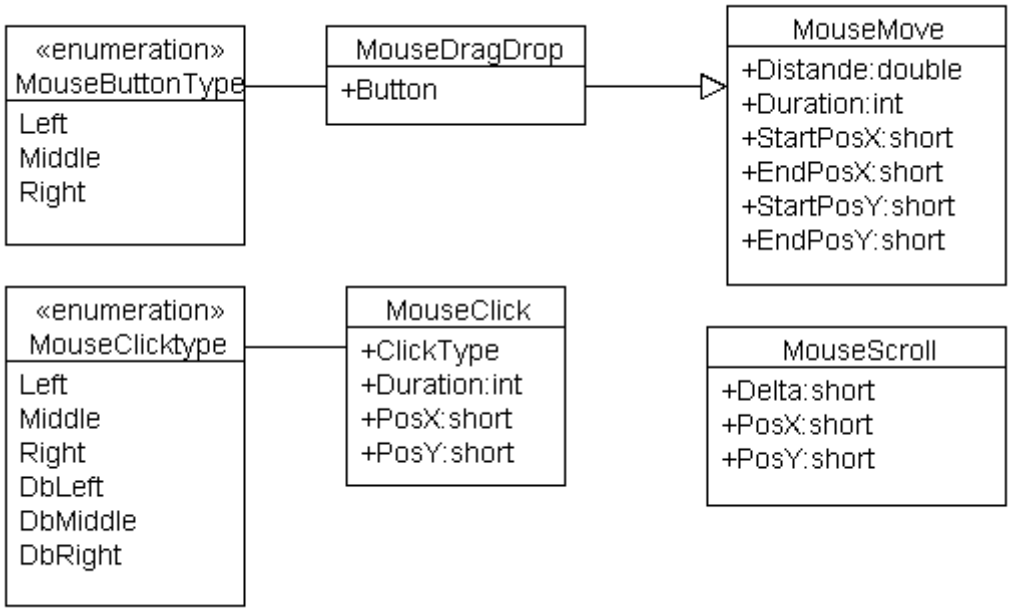
Entity uchovávajúce stav klávesnice a myši sú agregované v stave (entita *State*).

KeyboardState má atribút *Data* v ktorom sú uložená kolekcia serializovaných objektov vyjadrujúce stlčenia kláves, digrafy a trigrafy.



OBRÁZOK 7 –MODEL TRIED SÚVISIACI S KLÁVESNICOU

Entita *MouseState,* podobne ako *KeyboardState*, obsahuje kolekciu serializovaných objektov pre kliknutia, posun myši, dvojkliky atď. Ich diagram tried je na nasledovnom obrázku.



OBRÁZOK 8 – MODEL TRIED SÚVISIACICH Z MYŠOU

Dáta klávesnice a myši sa ukladajú v binárnom formáte kvôli predpokladanému veľkému objemu a kvôli rýchlosti načítania a ukladania záznamov.

4.3 PRIDANIE EMOCIONÁLNEHO STAVU POUŽÍVATEĽA DO LOGOV A DÁTOVÉHO MODELU

Zodpovedné osoby: Martin Geier, Peter Greguš

V tejto časti sme sa sústredili na analýzu zdrojového kódu pri vytváraní logov používateľových akcií, ich uložení na lokálnu databázu, následnom odosielaní na server a pridania emocionálneho stavu používateľa do týchto logov.

4.3.1 ANALÝZA

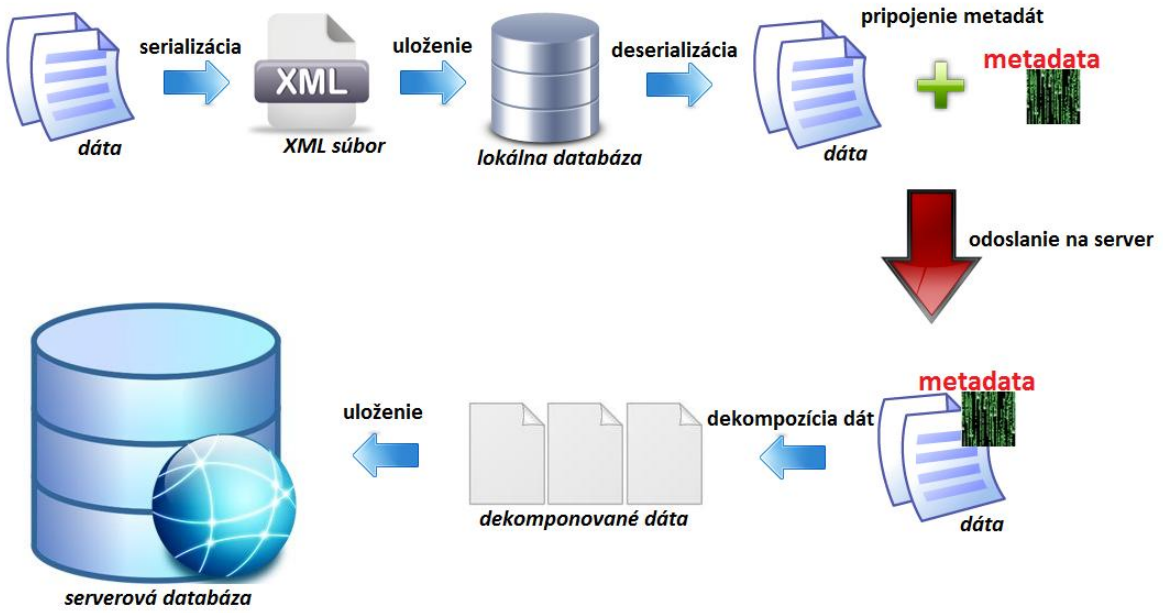
Z analýzy vyplýva, že každá sledovaná aktivita (stav klávesnice, myši) je zaznamenávaná triedou zdedenými od rozhrania *IActivityWatcher* (napr. *KeyboardStateWatcher*, *MouseStateWatcher*). Rozhranie *IActivityWatcher* predpisuje metódu *OnActivityFinishing* vracajúcu objekt typu *StateDto* nesúci konkrétny stav. Jednotlivé *Watcher-y* vytvárajú špecializované objekty zdedené od abstraktnej triedy *StateDto* (napr. *KeyboardStateDto*, *MouseStateDto*). Tieto jednotlivé objekty sú následne serializované do formátu *XML* uložené do lokálnej databázy. *Serializácia* vytvára *XML* na základe názvov atribútov dátových objektov <*Type*>*StateDto* a ich hodnôt.

Lokálna databáza pozostáva s jednej tabuľky, ktorej stĺpce sú nasledovné: *id (TEXT PRIMARY KEY)*, *StartTime (DataTim)*, *EndTime (DataTime)* a *Data (TEXT)*. Začiatočný

a koncový čas nesú informáciu o trvaní konkrétneho logu a v položke *Data* sa nachádza samotný log uložený vo formáte *XML*.

Po uplynutí časového intervalu sa dáta z lokálnej databázy odošlú na server. Časový interval je nastavený v súbore *app.config* s prednastavenou hodnotou 30 minút. V tomto súbore je uložená aj adresa servera, na ktorý sa dáta odosielajú. Po uplynutí stanoveného intervalu sa vyberú dáta z lokálnej databázy, deserializujú sa na objekty typu *ActivityDto*, ktorá obsahuje kolekciu objektov *<Type>StateDto* a pridá metadáta (začiatočný čas, koncový čas, názov pracovnej stanice, identifikátor používateľa). Odosielanie týchto dát je realizované prostredníctvom volania vzdialenej metódy *CommitActivity*, ktorej argumentom je objekt typu *ActivityDto*. V tejto metóde sa dáta skonvertujú na objekty, ktoré automaticky ukladajú hodnoty ich atribútov do databázy.

Autentifikácia používateľa je realizovaná pomocou atribútov triedy *ActivityDto* - názov pracovnej stanice a identifikátor používateľa.



OBRÁZOK 9 - TOK ZÍSKANÝCH DÁT O POUŽÍVATEĽOVI

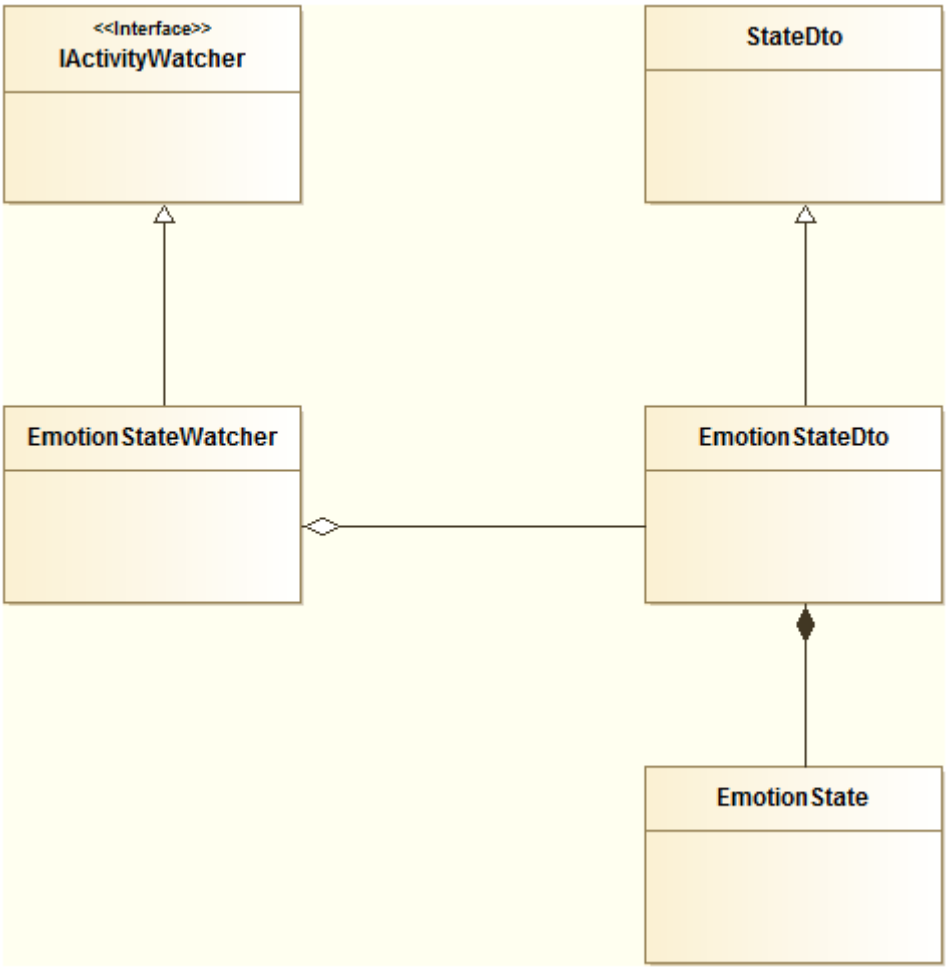
4.3.2 NÁVRH

Na základe analýzy navrhujeme vytvoriť triedu *EmotionStateWatcher* zdedenú od rozhrania IActivityWatcher, ktorá bude uchovávať aktuálny emocionálny stav používateľa. Ten to stav sa zmení po zadaní nového stavu používateľom v *GUI* časti. Stav uvedený v tejto triede je potrebné odovzdať predpísanou metódou *OnActivityFinishing* v objekte typu *StateDto*. Preto je potrebné vytvoriť triedu *EmotionStateDto* zdedenú od *StateDto*. Táto trieda bude obsahovať jeden argument typu enumeračný typ, ktorý môže nadobúdať hodnoty (Radosť, Normálny, Stres, Únava, Hnev). Informácia o emocionálnom stave používateľa sa automaticky pridá do *XML* súboru, uloží do lokálnej databázy a odošle na server. Tam bude

v nasledujúcom kroku potrebné pridať položku emocionálny stav do databázy a pridať konvertor k týmto dátam.

4.3.3 Implementácia

Návrh bol realizovaný implementáciou uvedených tried *EmotionStateWatcher*, *EmotionStateDto* a enumeračného typu *EmotionState*.



Obrázok 10 - Implementácia logovania emocionálneho stavu používateľa

4.4 Analýza emocionálneho stavu používateľa

Zodpovedné osoby: Martin Geier, Peter Greguš, Peter Šinský

Emócie používateľa sú kľúčovou oblasťou pre náš projekt. V tejto kapitole sa venujeme analýze štyroch emočných stavov, ktoré budeme rozpoznávať na základe používateľského vstupu. Analýza každej emócie prináša opis, príčiny a prejavy, a následne vhodné odporúčania, ktorých cieľom je zefektívniť prácu používateľa a dostať ho stavu negatívnej emócie do stavu pozitívnej.

4.4.1 HNEV

Hnev je negatívny emocionálny prejav, ktorý vyjadruje istú reakciu na prekážku, ktorá sa stavia do cesty pri dosahovaní nejakého cieľa. To znamená, že ak používateľovi niečo napríklad bráni v plnení aktuálnej úlohy, má tendenciu začať sa rozčuľovať a potom sa práca iba zhoršuje. Hnev ako emócia má štyri úrovne: rozčúlenie (najslabšia), hnev, zlosť a zúrivosť (najsilnejšia).

Príčiny hnevu

Existuje mnoho príčin, ktoré u ľudí môžu vyvolávať prejavy hnevu. Ak berieme do úvahy prácu pri počítači a na pracovisku, hnev môžu vyvolať nasledovné podnety:

- problémy s počítačom (pomalý internet, vyskakovanie okien, atď.),
- neschopnosť plniť úlohu (používateľ začína byť frustrovaný, mrzutý a nahnevaný keď nezvláda plniť zdaný problém atď.),
- preťaženie z práce,
- potýčky na pracovisku (hádky z vedením, nedostatočné ohodnotenie, neférové zaobchádzanie atď.),
- a mnohé iné ...

Prejavy hnevu

Ak sa používateľ nachádza v stave hnevu, jeho prejavy môžu byť rozličné. V závislosti od úrovne rozčúlenia, sa môžu prejaviť prejavy ako plač, krik, búchanie po stole atď. V kontexte práce pri počítači sa hnev prejavuje najmä na schopnosti plniť si úlohy. Ak používateľ pracuje pod vplyvom hnevu, je veľmi pravdepodobné, že:

- nevie sa sústrediť na prácu,
- pri písaní textu robí veľa chýb (napr. stláča viacero kláves naraz),
- pohyby myšou môžu byť ostrejšie, tak isto kliknutia na tlačidlo.

Odporúčania

Existuje mnoho odporúčaní a návodov, ktoré umožňujú zvládať hnev. Pri práci s počítačom je dôležité túto emóciu odbúrať a nasledovné príklady odporúčaní by mali pomôcť:

- prestávka od aktuálnej práce,
- dychové cvičenia (+ počítanie),
- vyhľadať zábavu (prečítať si vtip, pozrieť si vtipné video, zahrať si hru),
- dať si malú prechádzku (ísť sa vyvetrať, pozrieť sa na výhľad z kancelárie).

4.4.2 ÚNAVA

Prirodzeným dôsledkom dostatočne intenzívnej a dlhotrvajúcej činnosti je únava. Môže byť charakterizovaná ako vyčerpanie alebo strata energie a životného elánu. Toto môže byť dôsledkom jednak fyzického a v prípade kontextu našej práce najmä duševného vypätia.

Príčiny únavy

Únava môže byť spôsobená viacerými príčinami. Napríklad z fyziologického hľadiska, to môže byť nevhodná strava, nedostatok vitamínov alebo spánku. Tieto príčiny môžu byť ešte umocnené príčinami, ktoré vyplývajú práve z práce pri počítači. Keď používateľ pracuje priveľa, môže byť stresovaný a práve únava môže byť podvedomá obrana proti nemu. Ďalšími príčinami môže byť napríklad nedostatočne osvetlený priestor alebo zle vetraná kancelária. Únava je emocionálny stav, ktorý sa však môže prejaviť bez vysvetliteľných príčin.

Prejavy únavy

Únava je emocionálny stav, ktorý sa pri práce s počítačom môže prejaviť nasledovne:

- znížená produktivita,
- neschopnosť sústrediť sa na prácu,
- pomalé písanie na klávesnici,
- pomalé pohyby myšou a klikanie,
- robenie chýb.

Odporúčania

Únava môže byť sledovaná aj z dlhodobého hľadiska a na základe toho je možné používateľovi podať rôzne odporúčania. Pri dlhodobom prejavovaní je možné odporučiť:

- skvalitniť spánok (7 – 9 hodín),
- doplnenie vitamínov,
- výživové doplnky určené pre sústredenie.

Z krátkodobého hľadiska je možné únavu poraziť viacerými spôsobmi a to napríklad:

- prestávka od aktuálnej činnosti,
- káva alebo energetický nápoj,
- vyvetrať priestor,
- malú fyzickú aktivitu.

4.4.3 STRES

Podľa [Selye, 1976b, p. 64] stres je

> *"A state manifested by a specific syndrome which consists of all the nonspecifically induced changes within the biological system."*

Stres je psychický stav, kedy je človek priamo alebo nepriamo ohrozovaný alebo ohrozenie očakáva. Stres vyvoláva mobilizovanie všetkých síl človeka na prichádzajúcu udalosť. Stres rozdeľujeme na:

- *eustres* – pozitívny stres, dosahujeme vyšší výkon, napr. očakávanie príchodu milovanej osoby,
- *distres* – negatívny stres, poškodzuje psychické aj fyzické zdravie,
- *prestres* – zvyšuje odolnosť k stresu - napr. adrenalínové športy.

Príčiny stresu

Stres vyvoláva tzv. stresor. Môže to byť:

- životná udalosť – smrť niekoho blízkeho,
- vnútorné prežívanie – spomienky na nejakú životnú udalosť,
- vonkajšie prostredie – hluk, uzavretý priestor, teplota,
- vnútorné prostredie:
    - životný štýl – nedostatok spánku, nabitý program,
    - negatívny postoj – sebakritika, pesimistické myslenie,
    - psychický postoj – nerealistické očakávanie od seba, branie vecí osobne.

Prejavy stresu

Stres má vonkajšie a vnútorné prejavy, čo spôsobuje že na prvý pohľad sa môže javiť ako neškodný problém. Prejavy stresu môžeme rozdeliť na:

- fyzické symptómy – únava, tráviace problémy,
- duševné symptómy – strata koncentrácie, problémy pri rozhodovaní,
- symptómy v správaní – nepokoj alebo nervozita,
- citové symptómy – smútok, netrpezlivosť, podráždenosť.

Odporúčania

Na stres neexistujú krátkodobé metódy riešenia. Je potrebné ho riešiť dlhodobým prístupom. Medzi tieto patrí:

- budovanie medziľudských vzťahov,
- psychoterapeutický výcvik,
- venovanie sa záľubám,

- zlepšiť stravovanie - vylúčiť kofeín,
- praktizovanie relaxačných techník,
- v najťažších prípadoch medikamentná liečba.

4.4.4 RADOSŤ

Radosť je pozitívna emócia, ktorá je spojená so šťastím a dobrou náladou.

Príčiny radosti

Medzi hlavné príčiny radosti môžeme zaradiť:

- potešenie,
- dobré vzťahy,
- úspechy.

Potešenie zapríčiňuje strávenie človeku príjemnej aktivity (dobré jedlo, šport, hobby a pod.). Ďalším faktorom, ktorý významne vplýva na dobrú náladu človeka sú dobré vzťahy v jeho zázemí (s jeho rodinou, priateľmi, susedmi, kolegami a pod.). Nakoľko sú vzťahy väčšinou dlhodobé, majú aj väčší vplyv na samotnú emóciu radosti, ako potešenie z určitých aktivít, pretože tieto aktivity trvajú rádovo oveľa menej ako je trvanie vzťahov. Úspechy či už dosiahnuté v práci alebo osobnom živote majú taktiež dlhodobejší vplyv na náladu ľudí, keďže sa jedná o udalosť, ktorá ovplyvňuje ľudí na dlhší čas (povýšenie v práci, svadba, narodenie potomka a pod.). Emóciu radosti spôsobujú aj iné udalosti, avšak tri uvedené sa na nej podieľajú v najväčšej miere.

Prejavy radosti

Radosť sa pri počítači môže prejavovať nasledovne:

- zvýšená produktivita,
- väčšia miera kreativity,
- vyhýbanie sa chybám,
- rýchlejšia a cieľavedomejšia práca s myšou a klávesnicou.

Odporúčania

Ak sa človek dostane do stavu šťastia, je náročné v ňom zotrvať dlhodobo. Medzi dlhodobé „udržiavače“ človeka v dobrej nálade môžeme zaradiť

- ocenenia,
- rozmanitosť spúšťacích udalostí.

Pri oceneniach sa ľuďom vždy zvýši dobrá nálada nezávisle na počte ocenení. Pri zvyšných spúšťačoch šťastia je potrebné udržiavať širokú paletu týchto spúšťačov, aby sa predišlo

efektu rýchlej straty emócie radosti a dobrej nálady z dôvodu príliš veľkej frekvencie prežitia konkrétneho spúšťača.

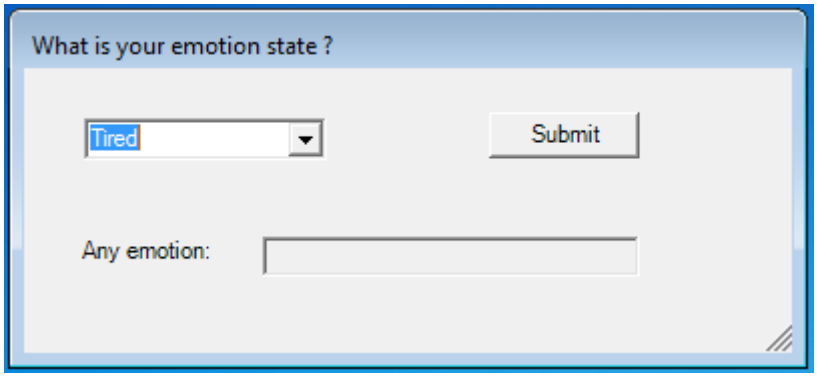
5 Šprint 2 - Henrieta

5.1 Pridanie funkcionality do programu *PerConik*

Zodpovedné osoby: Jozef Gajdoš, Martin Geier, Miroslav Hudák

5.1.1 Logovanie emócií

Na zadávanie emócií používateľom sa používa formulár, ktorý sa pýta používateľa na aktuálnu emóciu každých 30 minút.



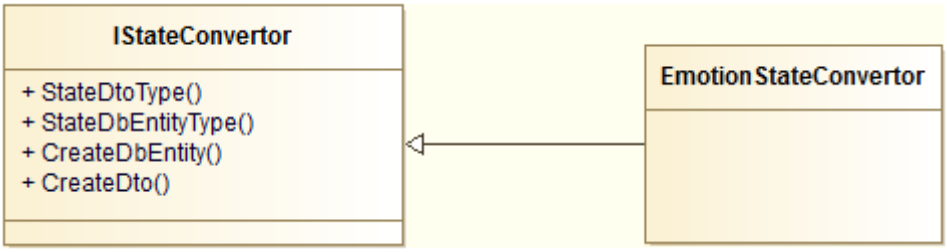
Obrázok 11 - Formulár pre zadávanie emocionálneho stavu

V ďalšej fáze bude pridané kontextové menu, aby sa emócia dala zadať kedykoľvek.

Zmena zadanej emócie vygeneruje udalosť v aplikácii, ktorá vytvorí *milestone*.

5.1.2 Pridanie konvertoru pre emócie

Po pridaní emócie do tabuľky stavy bolo potrebné vytvoriť konvertor z triedy *EmotionStateDto* do *Emotion*, čo je fyzická reprezentácia dát v databáze.

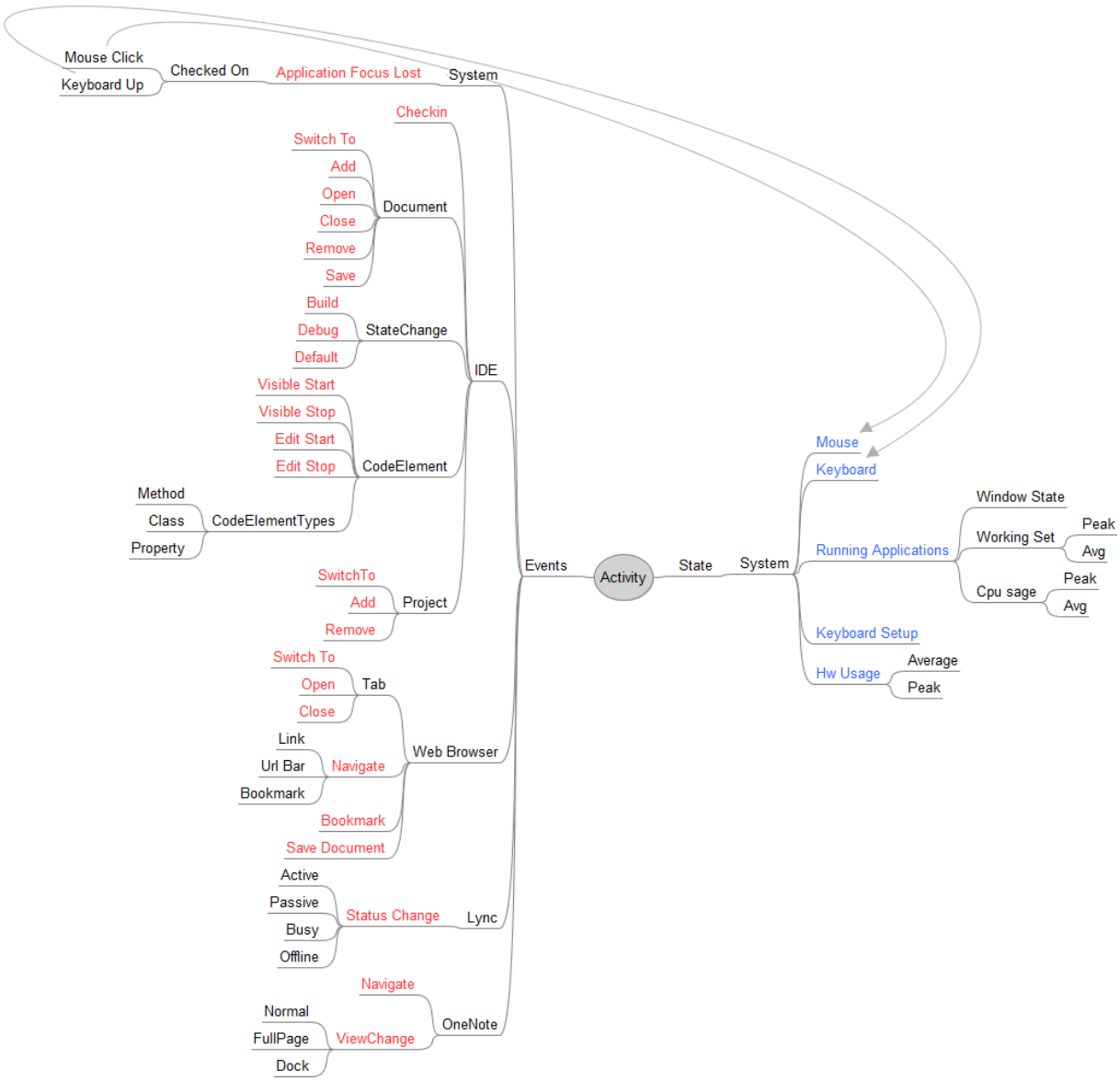


Obrázok 12 - Kovertor pre emócie

V triede *EmotionStateConvertor* sú implementované všetky metódy definované rozhraním. Objekt je automaticky volaný podľa typu ktorý vracia metódou *StateDtoType*. Prijatá emócia je prostredníctvom objektovo relačného mapovania uložená do tabuľky *Emotion*.

5.1.3 PRIDANIE GENEROVANIA MÍĽNIKA

V tejto kapitole je opísaná analýza, návrh a implementácia vlastnej významnej udalosti – míľnika. Míľnik je špeciálny typ udalosti, ktorý spúšťa zber a odosielanie údajov o aktivite. Každá aktivita je zakončená práve jedným míľnikom. *PerConIK* má implementovaných viacero udalostí, ktoré sú generované jednotlivými objektami triedy *ActivityWatcher*. Nie každá udalosť je zároveň aj míľnikom. Kompletný prehľad sledovaných udalostí je znázornený na nasledujúcom obrázku.



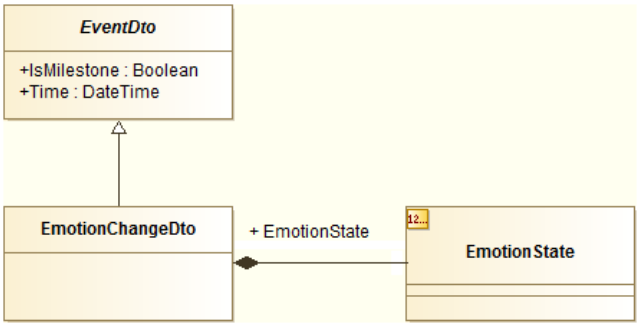
OBRÁZOK 13 - SLEDOVANÉ UDALOSTI A STAVY

Z množiny všetkých udalostí medzi míľniky patria:

- *ApplicationFocusLost*
- *LyncStatusChange*
- *IdeStateChange*
- *IdeProjectOperation*
- *IdeCheckin*

Ostatné udalosti nie sú nastavené ako míľniky, ale je možné, v prípade potreby, ich takto nakonfigurovať. Pre nás je dôležité zaznamenať udalosť vtedy, keď bola zmenená emócia. Po zmene emócie je potrebné vyvolať udalosť-míľnik, aby sa uložila predchádzajúca aktivita do databázy aj s príslušnou emóciou. Následne začne prebiehať nová aktivita s novou zmenenou emóciou.

Keď nastane zmena emócie, potrebujeme vyvolať metódu *BeginAddEvent* objektu triedy *ActivityComposer* a ako parameter zadať náš dátový objekt popisujúci emóciu. Navrhujeme vytvoriť triedu *EmotionChangeDto* zdedenú od abstraktnej triedy *EventDto*, ktorá bude uchovávať emóciu pre ukončenú aktivitu (*EmotionState*).



OBRÁZOK 14 - IMPLEMENTÁCIA VLASTNÉHO TYPU PRE UDALOSŤ

5.2 ANALÝZA DÁTOVÝCH ŠTRUKTÚR VSTUPNÝCH ZARIADENÍ

Zodpovedné osoby: Jozef Gajdoš, Martin Geier

V nasledujúcej časti sú opísané dátové triedy uchovávajúce informácie o vstupných zariadeniach, ktoré sú uložené v databáze a umožňujú analýzu práce s počítačom, ktorá by neskôr mohla viesť ku odhaľovaniu emocionálnych stavov používateľa.

5.2.1 ANALÝZA DÁTOVÝCH ŠTRUKTÚR MYŠI

MouseMoveDto:

- začiatočná pozícia,

- koncová pozícia,
- trvanie.

Prejdená vzdialenosť (súčasne je aj rýchlosťou $v = \frac{\partial d}{\partial t} \overset{t=1}{\longrightarrow} v_i = d_i$).

Zrýchlenie $a = \frac{\partial v}{\partial t} \overset{t=1}{\longrightarrow} a_i = v_i - v_{i-1} = d_i - d_{i-1}$.

MouseDragDropDto:

To isté ako *MouseMoveDto*, dedí od nej.

MouseScrollDto:

- pozícia,
- trvanie,
- otáčanie kolieskom.

MouseClickDto:

- pozícia,
- trvanie,
- typ tlačidla myši.

MousePosDto:

Vyjadruje pozíciu v daných štruktúrach, obsahuje súradnice x a y.

5.2.2 Analýza dátových štruktúr klávesnice

KeyboardGraphDto:

- znaky,
- latencia,
- doba letu.

Latencia je doba medzi stlačením a uvoľnením klávesy. Doba letu je doba medzi stlačením prvej a stlačením druhej klávesy.

Program *PerConIK* ukladá informácie o stlačených klávesoch, A – Z, 0 – 9 a špeciálnych znakoch. Ďalej kombinácie dvoch kláves, konkrétne: TH, IN, EH, RE, AN, ES, ER, ON, AT, TO, ST, NT, OR, TI, EN, EA, ND, LE, NG, ED, OF, CO, IS, AR, ET, PR, OV, PO, NA, NE, KO, VA, RO, RA, OU, OM, OS, NO, LI, LA, DO, VE, VO, HO, VI, LO, TE, SA a troch kláves: THE, ING, AND, ION, ENT, ATI, TIO, ESS, THA, FOR, ERE, TTH, EAR, SIN, STH, HAT, LEA, ILL, HER, ERS, COM, RES, INT, MEN, ETO, OVA, PRE, ALI, STA, TOR, STI, OVE, EHO, PRI, NOS, KTO, AKO, PRA, SPO, LOV, POD, OLO, CAS, KOU.

5.3 NÁVRH VÝPOČTU METRÍK PRE MYŠ

Zodpovedná osoba: Peter Šinský

Myš je vstupné počítačové zariadenie, ktoré sa používa na ovládanie polohy kurzora na obrazovke a vykonávanie operácií pomocou tlačidiel. Štandardne má myš tri tlačidlá vrátane rolovacieho kolieska. Pri práci s počítačom je myš neoddeliteľnou súčasťou a okrem klávesnice je vhodným nástrojom pre zachytávanie emocionálneho stavu používateľa.

V nasledujúcej časti sme analyzovali zachytávanie akcií vytváraných myšou v programe *PerConIK*. Na základe týchto vedomostí sme vymysleli metriky, podľa ktorých budeme vytvárať model používateľa.

5.3.1 SLEDOVANIE MYŠI POMOCOU PROGRAMU *PERCONIK*

Program *PerConIK* je schopný zachytávať tieto základné akcie:

- pohyb,
  - štartovná a koncová pozícia (X,Y),
  - trvanie pohybu,
  - vzdialenosť,
- „*drag and drop*“,
  - štartovná a koncová pozícia (X,Y),
  - trvanie pohybu,
  - vzdialenosť,
  - stlačené tlačidlo,
- rolovanie kolieska,
  - bod,
  - trvanie,
  - delta (otočenie kolieska)
- stlačenie tlačidla,
  - pozícia (X,Y),
  - trvanie,
  - stlačené tlačidlo.

5.3.2 METRIKY PRE SLEDOVANIE EMOČNÉHO STAVU POUŽÍVATEĽA

Pre vytvorenie modelu používateľa, na základe ktorého budeme schopný rozoznať používateľovo správanie, je potrebné vytvoriť metriky – sledovateľný údaj, ktorý budeme pozorovať pri práci používateľa s myšou. Tieto metriky sú vytvorené na základe vyššie spomenutých údajov, ktoré zachytáva program *PerConIK*.

Rýchlosť kurzora

Pohyb je najzákladnejšia akcia, ktorú je možné s myšou vykonávať. Pohybom myši nastavujeme kurzor na obrazovke na miesto, kde chceme vykonávať ďalšiu interakciu. Vzhľadom k tomu sa metrika javí ako vhodný kandidát pre rozoznávanie emócie. Rýchlosť pohybu kurzora ako metrika je vyjadrená nasledovne:

$$Rýchlosť\ kurzora = \frac{prejdená\ vzdialenosť}{Čas}$$

Rýchlosť je možné takto jednoducho sledovať. Predpokladáme, že pri zmene emočného stavu je sa bude meniť aj rýchlosť pohybu kurzora. Napr. pri rozčúlení môže používateľ reagovať agresívnejšie a rýchlosť bude vyššia, naopak pri únave môže byť rýchlosť výrazne pomalšia.

Rýchlosť rolovania kolieska

Pri prehliadaní internetu alebo rôznych dokumentov sa veľmi často používa na posúvanie koliesko. Týmto je možné posúvať prehliadaný obsah. Rovnako ako pri rýchlosti kurzora predpokladáme, že sa bude meniť vzhľadom na emočný stav. Rýchlosť rolovania kolieska je vyjadrená nasledovne:

$$Rýchlosť\ rolovania\ = \frac{Delta}{Čas}$$

Trvanie stlačenia tlačidiel

Pre vykonanie rôznych akcií je potrebné používať tlačidlá myši. Štandardne má myš tri tlačidlá. Trvanie stlačenia tlačidiel sa budeme sledovať pre všetky typy tlačidiel a ich stlačení, t.j. jednoduché stlačenie tlačidla, alebo dvojité.

5.4 NÁVRH VÝPOČTU METRÍK PRE KLÁVESNICU

Zodpovedná osoba: Peter Sivák

Klávesnica je vstupné počítačové zariadenie, ktoré sa používa prevažne na písanie textu, ale takisto slúži aj na vykonávanie operácií pomocou klávesových skratiek. Štandardne má klávesnica alfanumerickú, funkčnú a niekedy aj numerickú časť. Pri práci s počítačom je klávesnica neoddeliteľnou súčasťou a okrem myši je vhodným nástrojom pre zachytávanie emocionálneho stavu používateľa.

V nasledujúcej časti sme analyzovali zachytávanie akcií vytváraných klávesnicou v programe *PerConIK*. Na základe týchto vedomostí sme vymysleli metriky, podľa ktorých budeme vytvárať model používateľa.

5.4.1 SLEDOVANIE KLÁVESNICE POMOCOU PROGRAMU *PERCONIK*

Program *PerConIK* je schopný vytvárať nasledovné zoznamy:

- „*Graphs*“ – zoznam stlačených znakov

- „*DiGraphs*“ – zoznam stlačených dvojíc znakov
- „*TriGraphs*“ – zoznam stlačených trojíc znakov
- „*Shortcuts*“ – zoznam klávesových skratiek

Každý z vymenovaných zoznamov si v sebe uchováva nasledovné trojice údajov:

- „*Character*“ – hodnota stlačených znakov
- „*Latency*“ – latencia stlačenia
- „*FlightTime*“ – čas letu

Pre každý stlačený znak (s) na klávesnici sa vygenerujú dve udalosti, pričom prvá predstavuje čas stlačenia (T_p) a druhá, čas uvoľnenia klávesy (T_u). Pomocou týchto údajov sa pre znak vypočítajú 3 typy latencií:

- Doba stlačenia klávesy – „*Graph*“ latencia (L_t)
- „*DiGraph*“ latencia (D_t) – dvojica kláves
- „*TriGraph*“ latencia (T_t) – trojica kláves

Pre každý typ latencií sa súčasne vyráta aj čas letu stlačenia (F_t), ktorý predstavuje interval medzi stlačením rôznych kláves, takže zároveň určujú, či došlo k ich prekrývaniu.

Prostredníctvom nasledovných vzorcov budú vyjadrené časové hodnoty pre každý znak, takže celkový počet vyrátaných latencií.

$$L_t(s) = T_u(s) - T_p(s)$$

$$F_t(s) = T_p(s+1) - T_p(s)$$

$$D_t(s) = T_u(s) - T_p(s-1) = F_t(s-1) + L_t(s)$$

$$T_t(s) = T_u(s) - T_p(s-2) = F_t(s-2) + D_t(s)$$

Je nám známe, ktoré latencie boli vytvorené na základe odchytených udalostí z jedného procesu. Na základe procesov vieme potom povedať, o akú aplikáciu išlo, čo nám poskytuje možnosť odlišovať latencie (štýl písania používateľa) v rôznych aplikáciách.

5.4.2 METRIKY PRE SLEDOVANIE EMOČNÉHO STAVU POUŽÍVATEĽA

Pre vytvorenie modelu používateľa, na základe ktorého budeme schopný rozoznať používateľovo správanie, je potrebné vytvoriť metriky – sledovateľný údaj, ktorý budeme pozorovať pri práci používateľa s klávesnicou. Tieto metriky sú vytvorené na základe vyššie spomenutých údajov, ktoré zachytáva program *PerConIK*.

Frekvencia stlačených kláves

Stláčanie kláves je hlavná akcia, ktorú je možné s klávesnicou vykonávať. Stláčaním kláves píšeme text do aplikácií, alebo vykonávame rôzne operácie použitím klávesových skratiek.

Vzhľadom k tomu sa metrika javí ako vhodný kandidát pre rozpoznávanie emócie. Frekvencia stlačených kláves ako metrika je vyjadrená nasledovne:

$$Frekvencia\ stlačených\ kláves = \frac{počet\ stlačených\ kláves}{Čas}$$

Frekvenciu je možné takto jednoducho sledovať. Predpokladáme, že pri zmene emočného stavu sa bude meniť aj frekvencia stlačených kláves. Napr. pri rozčúlení môže používateľ reagovať agresívnejšie a frekvencia bude vyššia, naopak pri únave môže byť frekvencia výrazne menšia.

Doba letu

Ďalšou metrikou je doba letu. Reprezentuje trvanie medzi stlačením kláves. Hodnoty získame priamo zo získaných dát logovača.

Latencia

Ďalšou metrikou je latencia. Reprezentuje dobu stlačenia klávesy. Hodnoty získame priamo zo získaných dát logovača.

Vyrovnanosť pri písaní

Ďalšou zaujímavou metrikou je vyrovnanosť pri písaní textu. Keď používateľ píše na klávesnici nevyrovnane, môžeme predpokladať, že je nervóznejší ako obvykle. Naopak keď je jeho štýl písania vyrovnanejší, t.j. čas letu medzi jednotlivými klávesmi je približne rovnaký, používateľ sa javí byť pokojnejší a sústredenejší. Vyrovnanosť pri písaní je vyjadrená nasledovne:

$$Vyrovnanosť\ pri\ písaní\ = \sum_{i=1}^{n} (FlightTime_i - PriemernýFlightTime)^2$$

Výsledok 0 vyjadruje maximálnu vyrovnanosť pri písaní a čím je daný výsledok vyšší, tým je používateľ menej vyrovnaný.

Frekvencia výskytu chýb

Pri častom písaní textu na klávesnici sa používateľ dopúšťa chýb. Niekedy urobí viac chýb pri písaní a niekedy menej. Túto skutočnosť taktiež považujeme za dôležitú metriku, pretože čím viac chýb používateľ robí, tým je vzhľadom na emočný stav nervóznejší, ale takisto môže u neho prevládať hnev alebo aj únava, kedy už nie je plne koncentrovaný. Naopak, keď sa používateľ dopustí menšieho počtu chýb, môžeme predpokladať, že je pokojnejší a sústredenejší. Frekvencia výskytu chýb je vyjadrená nasledovne:

$$Frekvencia\ výskytu\ chýb\ = \frac{Počet\ stlačení\ backspace\ klávesy}{Čas}$$

5.5 NÁVRH MODELU POUŽÍVATEĽA

Zodpovedná osoba: Peter Greguš

Model používateľa z predchádzajúcej časti môžeme reprezentovať vektorom hodnôt metrík pre myš a klávesnicu. Môžeme si zhrnúť uvedené metriky.

Metriky pre myš:

- rýchlosť kurzora,
- rýchlosť rolovania kolieska,
- trvanie stlačenia tlačidiel.

Metriky pre klávesnicu:

- latencia *graph-ov, digraph-ov, trigraph-ov,*
- doba letu *digraph-ov, trigraph-ov,*
- frekvencia stlačených kláves,
- vyrovnanosť pri písaní,
- frekvencia výskytu chýb.

Tieto metriky budú reprezentované ako grafy početností jednotlivých hodnôt. Pomocou nich dokážeme vidieť správanie používateľa v jednotlivých emocionálnych rozpoloženiach. Po ich normalizácií ku množstvu získaných dát pomocou metódy logovania môžeme porovnávať konkrétne emócie, čo nám môže pomôcť pri rozpoznávaní modelu používateľa.

5.6 IMPLEMENTÁCIA MODELU POUŽÍVATEĽA

Zodpovedné osoby: Jozef Gajdoš, Martin Geier, Peter Greguš

5.6.1 PRIDANÉ ENTITY DO DÁTOVÉHO MODELU

Do modelu databázy boli pridané tri nové tabuľky:

- *UserModel,*
- *EmotionVector,*
- *Emotion.*

Teraz si ich podobnejšie popíšeme:

UserModel

Entita predstavuje model používateľa, ktorý v sebe agreguje modeli jednotlivých emócií. Tie sú uložené ako binárne dáta do typu *BLOB*. Modeli jednotlivých emócii sa ukladajú pomocou serializácie. Ďalej obsahuje svoje *Id* a je asociovaný ku konkrétnemu používateľovi.

EmotionVector

Entita *EmotionVector* slúži na dočasné ukladanie modelu emócie. Priamo v sebe má uvedenú emóciu a čas uloženia. Takisto sú jednotlivé záznamy asociované ku konkrétnym používateľom.

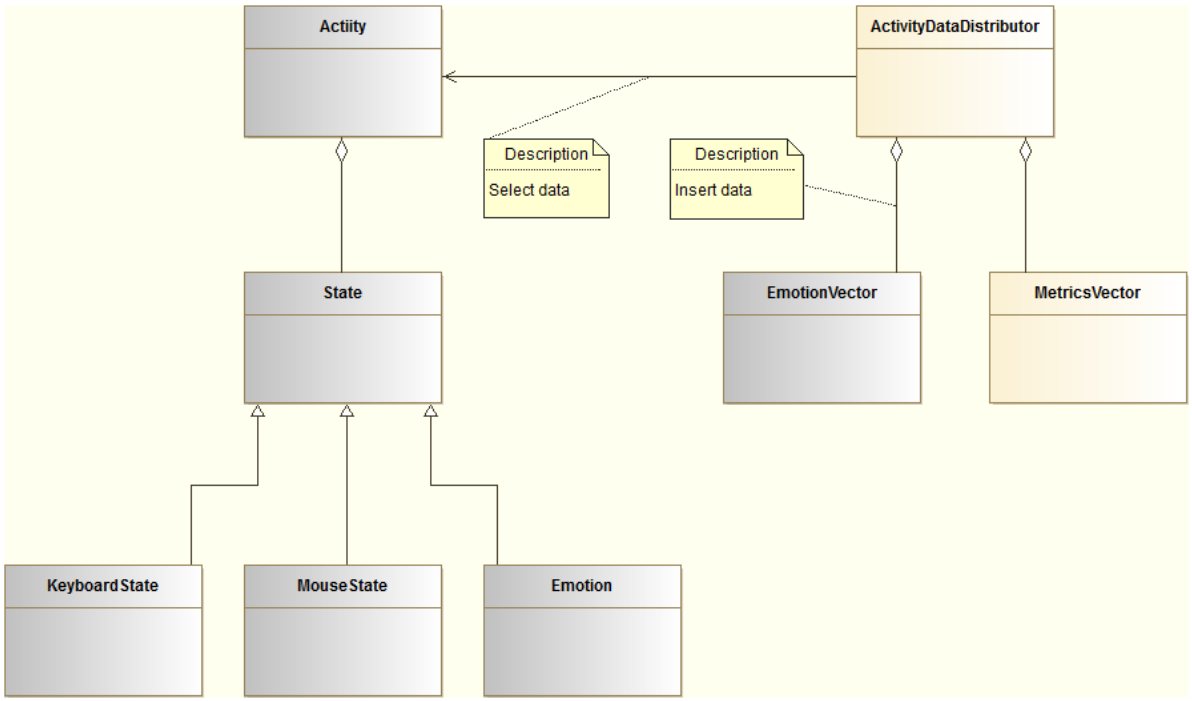
Emotion

Entita v sebe ukladá emóciu zadanú používateľom a aj vypočítanú hodnotu, obe hodnoty v čase vzniku nemusia byť uvedené.

Entita *Emotion* dedí od entity *State*.

5.6.2 Vytváranie vektoru metrík ako model používateľa

Vektor emócie vytvoríme naplnením vektora metrík pre jednotlivé emócie. Z databázy vyberieme dáta získané od používateľa, ktoré následne spracujeme a pomocou vzorcov uvedených v časti návrh výpočtu metrík a uložíme do príslušného vektora. Po príchode ďalších dát sa bude tento vektor prepočítavať a meniť podľa správania používateľa.



Obrázok 15 - Vytvorenie modelu používateľa

Na predošlom obrázku je šedou farbou znázornená databázová časť a hnedou časť implementačná.

Implementované metriky:

- rýchlosť kurzora,
- akcelerácia kurzora,
- rýchlosť rolovania kolieska,

- trvanie stlačenia tlačidiel.
- latencia *graph-ov*, *digraph-ov*, *trigraph-ov*,
- doba letu *digraph-ov*, *trigraph-ov*.

Trieda *MetricsVector* reprezentuje model používateľa v ktorom sú údaje zapísané nasledovnou formou:

```
/*KEYBOARD*/
public Dictionary<string,int[]> graphLatency = new Dictionary<string, int[]>();
public Dictionary<string,int[]> diGraphLatency = new Dictionary<string, int[]>();
public Dictionary<string,int[]> triGraphLatency = new Dictionary<string, int[]>();
public Dictionary<string,int[]> diGraphFlightTime = new Dictionary<string, int[]>();
public Dictionary<string,int[]> triGraphFlightTime = new Dictionary<string, int[]>();
/*MOUSE*/
public int[] mouseSpeed = new int[vectorSize];
public int[] mouseAcceleration = new int[vectorSize];
public int[] scrollSpeed = new int[vectorSize];
public Dictionary<MouseClickTypeEnum,int[]> clickDuration = new
Dictionary<MouseClickTypeEnum, int[]>();
```

Tieto dáta sú následne uložené do databázy ako serializované dáta v *BLOB* forme.

6 ŠPRINT 3 – IZABELA

6.1 DOPLNENIE MODELU POUŽÍVATEĽA

Zodpovedné osoby: Martin Geier, Peter Greguš

K modelu používateľa boli doplnené nasledovné metriky:

- *grapfFlifhtTime,*
- *shortcutLatency,*
- *shortcutFlightTime.*

Tieto metriky sú reprezentované obdobne ako už uvedené a implementované metriky, teda:

```
private Dictionary<string, int[]> graphFlightTime = new Dictionary<string, int[]>();
private Dictionary<string, int[]> shortcutLatency = new Dictionary<string, int[]>();
private Dictionary<string, int[]> shortcutFlightTime = new Dictionary<string, int[]>();
```

Ďalej boli implementované nasledovné metriky z časti 5.4.2, ktoré sa vypočítajú z predošle uvedených metrík:

- vyrovnanosť písania (*writing divergence*),
- frekvencia stláčania kláves (*key push frequency*),
- frekvencia výskytu chýb (*error frequency*).

6.2 METÓDY ROZPOZNÁVANIA MODELU

Zodpovedné osoby: Jozef Gajdoš, Martin Geier, Peter Sivák, Peter Šinský

6.2.1 NAIVNÁ METÓDA

Metóda porovnáva dva vektory, avšak tie je potrebné najprv vytvoriť, pretože sa v pamäti neuchovávajú. Jeden vektor je vytvorený z aktuálne získaných dát a druhý vektor z dát, ktoré sú uchovávané v databáze a reprezentujú model používateľa.

Vektor sa naplní tak, že pre každú hodnotu jednotlivých metrík (*graph, bigraph, trigraph,* atď.), ktoré sú uchovávané v poliach, sa vypočíta priemer nasledovne:

$$Priemer = \frac{\sum_{i=0}^{n} pole[i] * i}{\sum_{i=0}^{n} pole[i]}$$

Takto získané priemery sa postupne vkladajú do vektora.

Po vytvorení vektorov sa môže pristúpiť ku porovnaniu pomocou kosínusovej podobnosti. Tá sa vypočíta nasledovne:

$$Podobnosť = \frac{\sum_{i=0}^{n} A_i * B_i}{\sqrt{\sum_{i=0}^{n}(A_i)^2} * \sqrt{\sum_{i=0}^{n}(B_i)^2}}$$

Hodnoty A a B predstavujú porovnávané vektory. Kosínusová podobnosť môže mať na výstupe číslo od -1 až 1. Ak je na výstupe 1, oba vektory sú zhodné. V ostatných prípadoch zhodné nie sú.

6.2.2 METÓDA ZALOŽENÁ NA MANHATTANOVSKEJ VZDIALENOSTI VEKTOROV

Porovnávané vektory, sa rozdelia na 4 polia nasledovne:

1. latencia stlačenia kláves,
2. *flight time* stlačenia kláves,
3. dĺžky stlačení tlačidiel myši,
4. rýchlosť a zrýchlenie myši.

Pre každé pole sa vypočíta *Manhattanovskú* vzdialenosť nasledovne:

$$d = \sum_{i=0}^{n-1} |p_i - q_i|$$

Kde, d je *Manhattanovská* vzdialenosť, n je počet prvkov v poli, p_i sú prvky modelového vektora a q_i sú prvky porovnávaného vektora.

Pre získanie podobnosti vektorov v rozmedzí 0 až 1, je potrebné vzdialenosť onormovať. Preto sa vypočíta najhorší možný scenár pre 2 vektory a to nasledovne:

$$wc = \sum_{i=0}^{n-1} (p_i + q_i)$$

Po vypočítaní najhoršieho prípadu sa výsledná hodnota onormuje nasledovne:

$$sim_j = \frac{wc - d}{wc}$$

Takto sa získajú 4 hodnoty podobnosti polí. Pre výslednú podobnosť sa vypočíta vážený priemer podobností, ako:

$$sim = \frac{\sum_{i=0}^{4} sim_i w_i}{\sum_{i=0}^{4} w_i}$$

Kde, w_i je váha pre podobnosť sim_i. Váhy je možné dynamicky v programe meniť, prednastavené sú však na hodnotu 1.

6.2.3 METÓDA VÁŽENEJ KOSÍNUSOVEJ PODOBNOSTI VEKTOROV

Na porovnávanie podobnosti vektorov sa štandardne používajú vzdialenostné vektorové funkcie, medzi ktorú patrí kosínusová miera podobnosti.

V našom prípade je ale problém, že vektory emócií obsahujú veľké množstvo nulových hodnôt na rovnakých miestach a vyskytujú sa v nich aj podobné hodnoty, preto vypočítané podobnosti rôznych vektorov sa od seba iba málo líšia.

Váženie vychádza z myšlienky, prisúdiť vyššie váhy prvkom vektorov, ktoré sa nachádzajú na miestach v ktorých sa vektory emócií modelu používateľa vzájomne líšia viac. Tomuto zodpovedá smerodajná odchýlka.

E je množina vektorov emócii v modeli používateľa, $e_{v,i}$ je hodnota vektora emócie q na i-tom mieste. Potom hodnotu váhy w_i vypočítame ako:

$$\bar{e}_i = \frac{1}{|E|} \sum_{v \,\epsilon\, E} e_{v,i}$$

$$w_i = \sqrt{\frac{1}{|E|} \sum_{v \,\epsilon\, E} \left(e_{v,i} - \bar{e}_i\right)^2}$$

Vektor váh je charakteristický pre konkrétneho používateľa a jeho veľkosť môže poskytnúť dodatočnú informáciu o vzájomnej rozlíšiteľnosti jednotlivých emócií.

Podobnosť dvoch emočných vektorov $sim(p, q, w)$, kde p je vstupný vektor, $q \,\epsilon\, E$ je vektor emócie z modelu používateľa a w je vektor váh vypočítaný z modelu používateľa, vypočítame:

$$sim(p, q, w) = \frac{p \cdot q \cdot w}{|p|.|q|.|w|} = \frac{\sum_{i=0}^{len(w)} p_i.q_i.w_i}{\sqrt{\sum_{i=0}^{len(p)} p_i^2} \cdot \sqrt{\sum_{i=0}^{len(q)} q_i^2} \cdot \sum_{i=0}^{len(w)} w_i}$$

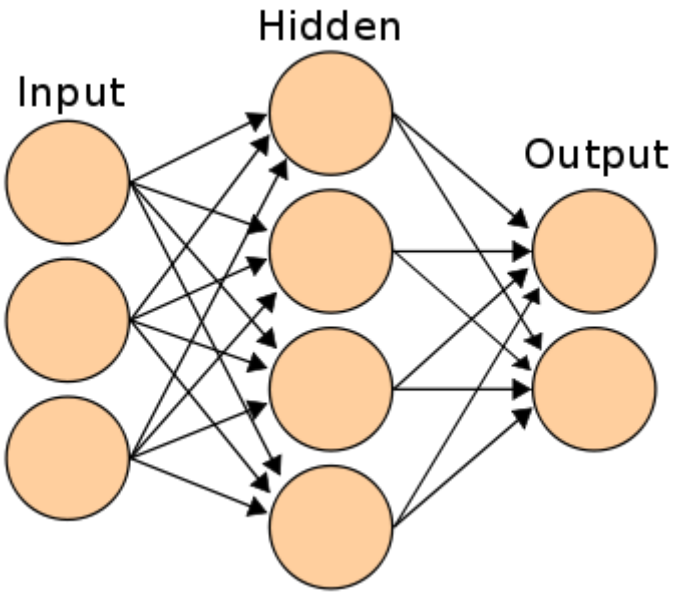
Vypočítaná podobnosť vektorov je v rozsahu <*-1,1*> preto ju je ešte potrebné normovať do rozsahu 0 až 1.

Vážená kosínusová podobnosť vektorov je implementovaná v triede *WeightingCosinuseComparator*.

6.2.4 METÓDA NEURÓNOVEJ SIETE

Neurónová sieť je všeobecný klasifikátor, ktorý sa dá využiť na mnohé rozpoznávacie problémy. Práve v našom systéme sme ju použili ako jeden z klasifikátorov emocionálnych stavov používateľa.

Základná štruktúra neurónovej siete je znázornená na nasledujúcom obrázku:



OBRÁZOK 16 - ZÁKLADNÁ SCHÉMA NEURÓNOVEJ SIETE

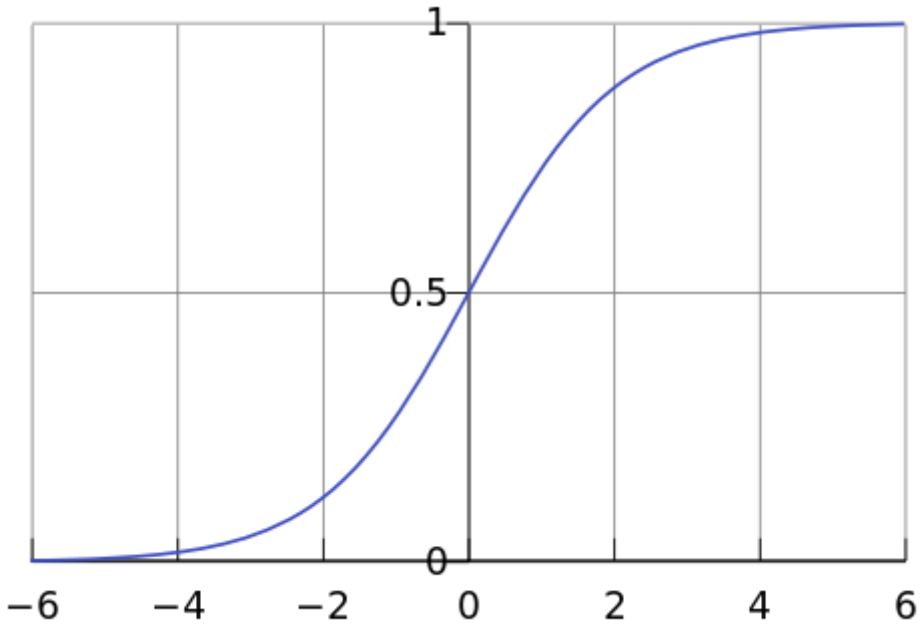
Všeobecne sa neurónová sieť skladá z troch typov vrstiev: vstupná vrstva, skrytá vrstva a výstupná vrstva. Skrytá vrstva sa ešte môže ďalej skladať z viacerých podvrstiev, čo je ale zvyčajne viac kontraproduktívne ako prínosné, preto sme v našej implementácii použili len jednu skrytú vrstvu.

Vstupná vrstva má toľko neurónov, koľko má vstupov a podobne výstupná vrstva obsahuje práve toľko neurónov, koľko má výstupov. Počet neurónov v skrytej vrstve sa nedá na začiatku optimálne určiť a preto je určený experimentálne.

Samotný neurón sa dá predstaviť ako funkcia, ktorá prijíma jeden alebo viac vstupov a na základe nich vráti jeden výstup:

$$y = f(\sum_{i=1}^{n} w_i x_i)$$

Z predchádzajúcej rovnice je vidieť, že výstup sa vypočíta ako suma váhovaných vstupov, ktorá sa ďalej použije ako vstup do aktivačnej funkcie, ktorá môže byť tiež rôzneho typu. Najčastejšie sa používa *sigmoidálna* aktivačná funkcia, ktorá nadobúda hodnoty od 0 po 1 a je zobrazená na nasledovnom obrázku:



OBRÁZOK 17 – *SIGMOIDÁLNA* AKTIVAČNÁ FUNKCIA

Vstupom do našej neurónovej siete sú dve skupiny reálnych čísel – údaje z modelu používateľa a hodnoty jeho aktuálnej emócie. Rozpoznávanie emocionálneho stavu pomocou neurónovej siete sa skladá z dvoch fáz – fázy trénovania a testovania. Pri trénovaní sú ako vstupy poskytnuté postupne všetky doposiaľ získané údaje od používateľa z databázy a ak sa model používateľa a jeho aktuálna emócia rovná, nastavíme ako očakávaný výstup neurónovej siete hodnotu 1. V opačnom prípade nastavíme očakávanú výstupnú hodnotu na 0. Postupne v niekoľkých iteráciách nastavujeme jednotlivé váhy neurónovej siete, aby sa reálny výstup blížil k očakávanému výstupu. Ak chyba pri trénovaní klesne pod určitú nami zvolenú hranicu, trénovanie končí a sieť je následne uložená do súboru.

Namiesto štandardného trénovacieho algoritmu známeho ako *backpropagation* algoritmus sme použili techniku *resilient propagation*, ktorá dokáže oveľa rýchlejšie konvergovať k požadovanému výsledku ako spomenutá štandardná metóda. Naviac je schopná plne využiť naraz viac jadier procesora, takže aj v tomto ohľade sa dá jej priebeh ešte zrýchliť. Na samotné trénovanie neurónovej siete sme využili voľne dostupný rámec *encog*, ktorý spomedzi všetkých voľne dostupných rámcov pre neurónovú sieť a strojové učenie dosahuje najlepšie výsledky.

Čo sa týka implementácie, rozdelili sme kód ohľadne trénovania neurónovej siete na tri hlavné triedy

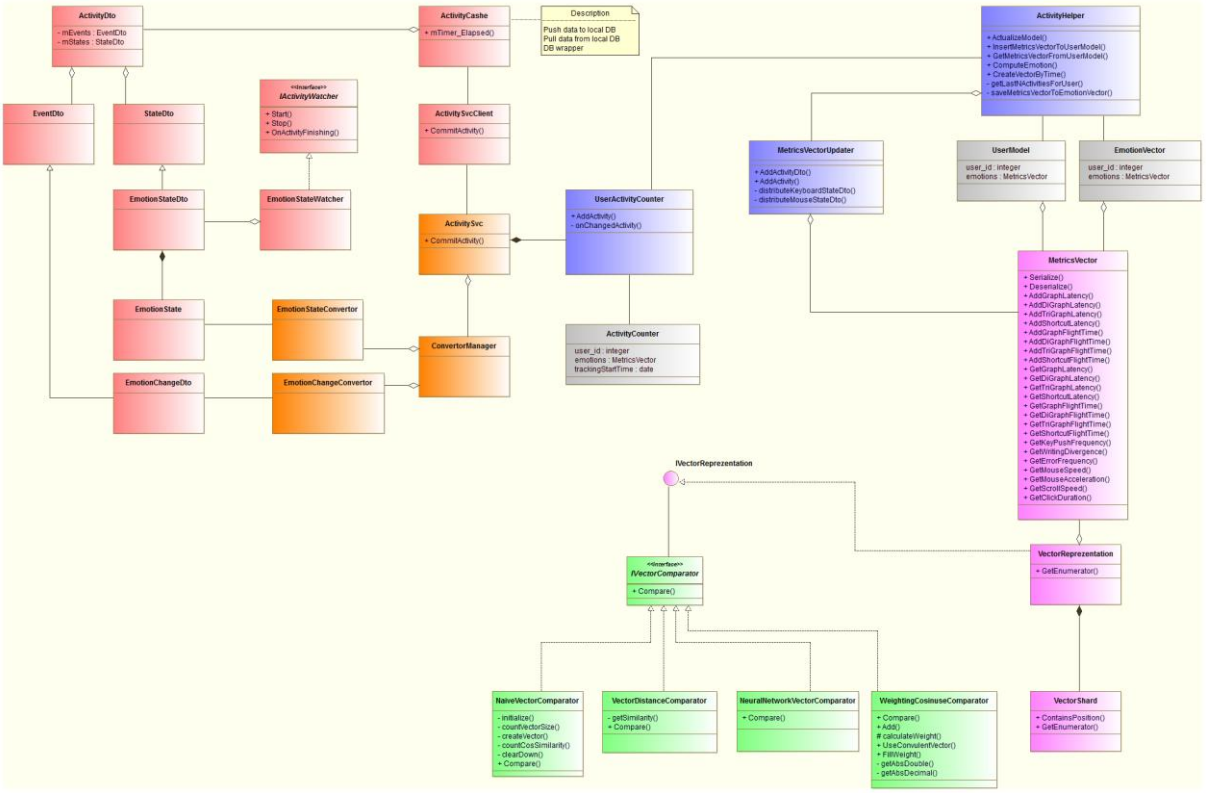
- *Trainer,*
- *DataSetBuilder,*
- *Serializer.*

Trieda *Trainer* slúži na samotné trénovanie neurónovej siete. Trieda *DataSetBuilder* slúži na transformáciu dát z databázy do tvaru, ktorý sa dá použiť ako vstup do neurónovej siete. Trieda *Serializer* slúži na uloženie samotnej neurónovej siete do súboru a neskôr na jej načítanie zo súboru.

Aktuálna verzia trénovania neurónovej siete ešte nie je úplne použiteľná, pretože pri veľkom množstve načítaných trénovacích dát sa pamäť preplní a nie je možné spracovať všetky dáta potrebné na postačujúce natrénovanie neurónovej siete. Preto bude treba v budúcnosti zmeniť systém načítavania údajov, aby sa tento problém odstránil. Takisto sa momentálne neurónová sieť ukladá do textového súboru, ktorý je zbytočné veľký. Bude treba zmeniť jeho ukladanie do binárneho súboru, čo rapídne zmenší veľkosť súboru s natrénovanou sieťou.

6.3 UML DIAGRAM DOPOSIAĽ VYTVORENEJ APLIKÁCIE

Zodpovedná osoba: Peter Greguš



OBRÁZOK 18 - UML DIAGRAM TRIED AKTUÁLNEHO STAVU APLIKÁCIE

Popis farebného usporiadania diagramu:

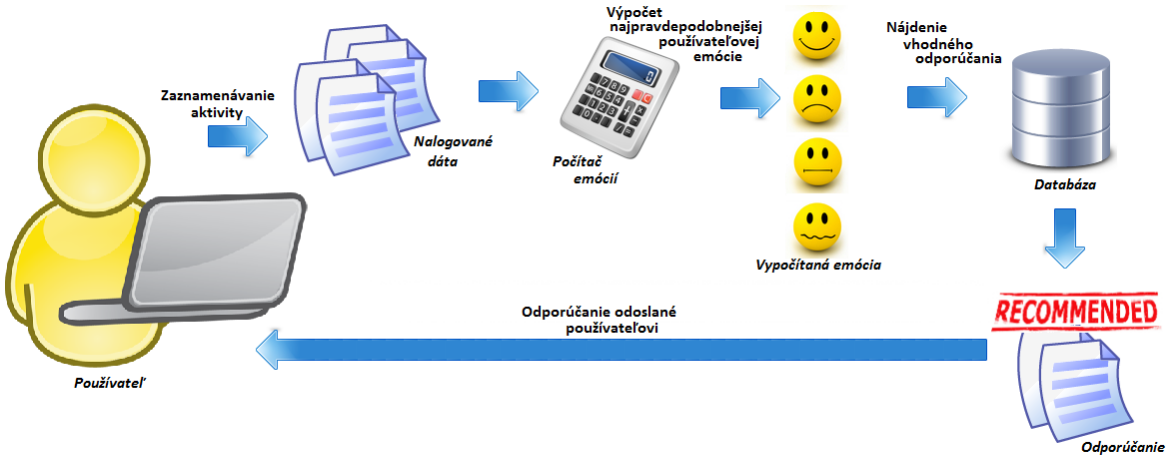
- červená – triedy programu *PerConIK* (klientska časť),
- oranžová – triedy programu *PerConIK* (serverová časť),
- sivá – databázová časť,
- modrá – výpočet metrík modelu,
- fialová – model,
- zelená – rozpoznávanie modelu.

7 Šprint 4 - Karolína

7.1 Odporúčanie pre používateľa

Zodpovedné osoby: Peter Greguš, Miroslav Hudák

Odporúčanie pre používateľa dokážeme vybrať na základe nalogovaných dát. Po určitom čase sa nazbiera dostatok dát, ktoré porovnáme s modelom konkrétneho používateľa a zistíme jeho najpravdepodobnejší emocionálny stav. Na základe toho vyberieme odporúčanie ktoré mu ponúkneme pre zlepšenie jeho stavu z negatívnej emócie na emóciu pozitívnu.



Obrázok 19 - Proces odporúčania

7.1.1 Vybranie odporúčania

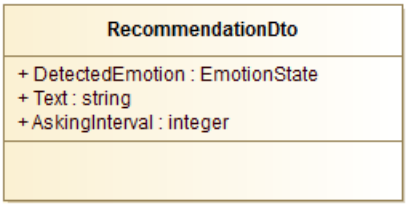
Po príchode dopytu od klientskej časti programu, ktorá si pýta odporúčanie sa požiadavka spracuje triedou *Recommender*. Táto trieda pozrie či sa v databáze nachádza vypočítaná hodnota emócie používateľa v nedávnom časovom období napr. pol hodiny. Ak áno, tak vyberie posledný takýto záznam a zistí, aká je najpravdepodobnejšia vypočítaná emócia a na základe nej vyberie z databázy odporúčanie pre túto konkrétnu emóciu. Tú potom vráti klientovi v podobe triedy *RecommendationDto*, ktorá obsahuje samotné odporúčanie.

7.1.2 Služba vracajúca odporúčanie

Do programu PerConIK bola doimplementovaná metóda vracajúca odporúčanie pre používateľa ako rozšírenie existujúcej služby definovanej rozhraním *IActivitySvc*. Bola doplnená metóda *GetRecommendation*, ktorá berie ako parameter ID používateľa (ako reťazec) a vracia odporúčanie vo forme objektu typu *RecommendationDto*.

Trieda *RecommendationDto* obsahuje:

- *DetectedEmotion* : *EmotionState* – vypočítaná emócia odporúčačom
- *Text* : *string* – textový opis odporúčania
- *AskingInterval* : *integer* – nový interval opýtania sa emocionálneho stavu používateľa



OBRÁZOK 20 - TRIEDA RECOMMENDATIONDTO

Služba odporúčania sa vykonáva v pravidelných intervaloch. Interval je možné si nastaviť v časti *Settings UACA* aplikácie. Odporúčanie sa zobrazí používateľovi iba v prípade úspešnej detekcie emocionálneho stavu a zostavenia primeraného odporúčania. Odporúčanie sa v tejto fáze projektu zobrazuje ako bublinové okno nad *tray* ikonkou.

7.2 IMPLEMENTÁCIA SERVEROVEJ ČASTI

Zodpovedná osoba: Martin Geier

7.2.1 SPUSTENIE SLUŽBY *PERCONIK* NA SERVERI

Postup nasadenia služby na server:

1. Vytvorenie *ApplicationPool* pre službu *PerConIK* s nastaveným *NetFramework-om* 4.0 a *PipeLine* typu *integrated*.
2. Vytvorenie novej web stránky s menom *PerConIK* spustenej na porte 3389.
3. Vytvorenie aplikácie vo webovej stránke *PerConIK* s virtuálnym priečinkom *PerConIK* a mapovaním na fyzický disk do priečinku wwwroot/svc.
4. Nastavenie väzby webovej stránky na team10-12.ucebne.fiit.stuba.sk.
5. Nakopírovanie služby do priečinku wwwroot/svc.
6. Webová služba je dostupná na adrese http://team10-12.ucebne.fiit.stuba.sk:3389/PerConIK/.

7.2.2 KOMPOZÍCIA SERVEROVÝCH KOMPONENTOV

Serverová časť aplikácie *PerConIK* bola rozšírená o automatické vytváranie a ukladanie *EmotionVector-ov*, ktoré sú generované z 10 minútových intervalov. Po implementovaní tejto časti bola aplikácia nasadená na server. Na server boli postupne odoslané aktivity členov tímu. Nad týmito dátami bola následne robená analýza.

7.3 VYHODNOTENIE VÝSLEDKOV

Zodpovedné osoby: Jozef Gajdoš, Martin Geier, Peter Greguš

7.3.1 PREDSPRACOVANIE DÁT

Program Fill Temporary Calculated Emotions

Program slúži na vypočítanie výsledkov porovnávania emočných vektorov s modelom používateľa a tým nám umožňuje robiť vzájomné porovnanie porovnávacích algoritmov. Program si najskôr z databázy vytiahne zoznam používateľov a pre každého si v tabuľke *EmotionVector* vyberie emočné vektory, ktoré postupne porovná s každým modelom emócie v modeli používateľa. Na porovnávanie sa používajú všetky aktuálne implementované porovnávacie algoritmy (Naivná metóda, Porovnávanie na základe manhattanovskej vzdialenosti, Váhovaná kosínusová podobnosť). Výsledkom porovnávania je vždy emócia ktorej model sa najviac podobá na emočný vektor a podobnosť. Tieto údaje pre každý emočný vektor sa ukladajú do tabuľky *TemporaryCalculadedEmotions*.

Program Import Export Model

Program slúži na importovanie modelu používateľa zo súboru alebo na export modelu používateľa z databázy do súboru. Je implementovaný ako jednoduchá GUI aplikácia.

7.3.2 ANALÝZA VÝSLEDKOV

Analýza prebiehala na jednom členovi tímu. Člen postupne odoslal všetky svoje nazbierané dáta na server, z ktorých boli automaticky generované testovacie vektory. Testovacie vektory majú jednoznačne určenú emóciu podľa toho, akú emóciu mal používateľ nastavenú v čase keď sa zbierali dáta na testovací vektor.

Používateľ odoslal na server približne 12 000 aktivít, z nich bolo vygenerovaných cca 900 testovacích vektorov. Testovacie vektory boli spracované troma metódami na porovnávanie vektorov s modelom. Po spracovaní testovacieho vektora boli uložené najpravdepodobnejšie emócie aj s vypočítanou podobnosťou. Podobnosť bola využitá na odstránenie tých vypočítaných emócií ktoré majú menšiu podobnosť najpravdepodobnejšej emócie akú sme zadali.

Nasledujúce tabuľka zobrazuje výsledky pre jednotlivé metódy .V ľavom stĺpci sú metódou vypočítané emócie, stĺpec *True* zobrazuje podiel úspešných a všetkých vypočítaných emócií. *Count of Activity* zobrazuje počet aktivít z ktorých bol vytváraný model a posledný stĺpec zobrazuje stĺpec *True* ováhovaný stĺpcom *Count*. *Accuracy* zobrazuje celkovú úspešnosť metódy.

Naivná metóda, orezanie na 80% podobnosť.

Emotion	True	False	Count	Count of Activity	Weight
Normal				8400	0

Happiness	0,129032	0,870968	31	589	4
Surprise	0,0625	0,9375	64	13	4
Anger	0,033333	0,966667	90	120	3
Disgust	0,714286	0,285714	7	526	5
Fear	0,014493	0,985507	138	6	2
Sadness	0,023256	0,976744	43	59	1
Tired	0,5	0,5	2	2441	1
Stressed	0,027523	0,972477	109	181	3
Nervous	0,063158	0,936842	95	119	6
Any	0,018519	0,981481	54	21	1
Accuracy					4,739336493

Metóda nedáva postačujúce výsledky, preto s touto metódou nie sú robené ďalšie pozorovania.

Metóda používajúca *VectorDistanceComparator*, orezanie na 7% podobnosti.

Emotion	True	False	Count	Count of Activity	Weight
Normal	1	0	30	8400	30
Happiness	0,933333	0,066667	15	589	14
Surprise	0,16	0,84	25	13	4
Anger	0,076923	0,923077	39	120	3
Disgust	0,666667	0,333333	24	526	16
Fear	0,018692	0,981308	107	6	2
Sadness	0,071429	0,928571	14	59	1
Tired	0,892857	0,107143	28	2441	25
Stressed	0,076923	0,923077	52	181	4
Nervous	0,175	0,825	40	119	7
Any	0,071429	0,928571	14	21	1
Accuracy					27,57732

Vypočítaná pravdepodobnosť presného určenia reálnej emócie je relatívne malá, no vo veľkej miere je ovplyvnená emóciami, ktorých model vznikol z mála aktivít. Pri odstránení emócií, ktorých model vznikol z menej ako 100 aktivít dostaneme nasledovnú tabuľku.

Emotion	True	False	Count	Count of Activity	Weight
Normal	1	0	30	8400	30
Happiness	0,933333	0,066667	15	589	14
Anger	0,076923	0,923077	39	120	3
Disgust	0,666667	0,333333	24	526	16
Tired	0,892857	0,107143	28	2441	25
Stressed	0,076923	0,923077	52	181	4
Nervous	0,175	0,825	40	119	7
Accuracy					43,42105

Z ktorej je vidieť, že pravdepodobnosť presne vypočítanej emócie sa takmer zdvojnásobená. Ak odstránime aj emócie ktorých podiel je pri vytváraní modelu stále relatívne malý, dostaneme nasledujúce údaje.

Emotion	True	False	Count	Count of Activity	Weight
Normal	1	0	30	8400	30
Happiness	0,933333	0,066667	15	589	14
Disgust	0,666667	0,333333	24	526	16
Tired	0,892857	0,107143	28	2441	25
Accuracy					87,62887

Tabuľka zobrazuje emócie, ktoré používateľ prežíva najčastejšie. Ich odhad podľa tejto metódy je takmer v 9 z 10 prípadov správny.

Metóda používajúca *WeightingCosinusComparator*, orezanie na 0,00080 podobnosti.

Emotion	True	False	Count	Count of Activity	Weight
Normal	0,768559	0,231441	229	8400	176
Happiness	1	0	5	589	5
Surprise	0,363636	0,636364	11	13	4
Anger	0,428571	0,571429	7	120	3
Disgust	0,666667	0,333333	9	526	6
Fear	0,008065	0,991935	248	6	2
Sadness	0,25	0,75	4	59	1
Tired	0,888889	0,111111	9	2441	8
Stressed	0,666667	0,333333	6	181	4
Nervous	0,666667	0,333333	9	119	6
Any	0,166667	0,833333	6	21	1
Accuracy					39,77901

Podobnou úvahou ako pri predchádzajúcich tabuľkách, môžeme aj tu odstrániť emócie s malým počtom aktivít vytvárajúcich model pre emóciu.

Emotion	True	False	Count	Count of Activity	Weight
Normal	0,768559	0,231441	229	8400	176
Happiness	1	0	5	589	5
Anger	0,428571	0,571429	7	120	3
Disgust	0,666667	0,333333	9	526	6
Tired	0,888889	0,111111	9	2441	8
Stressed	0,666667	0,333333	6	181	4
Nervous	0,666667	0,333333	9	119	6
Accuracy					75,91241

Pravdepodobnosť správne vypočítanej emócie sa už pri týchto nastaveniach dostala cez správnosť v troch prípadoch zo štyroch. Hlavným určujúcim faktorom je tu emócia Normal, ktorá má vysokú váhu z dôvodu jej častého výskytu.

7.4 PRIPOJENIE SA NA SERVER CEZ *REMOTE DESKTOP*

Zodpovedná osoba: Peter Sivák

Doposiaľ sme na pripojenie sa k serveru používali program typu *VNC*, konkrétne *UltraVNC*. Jeho problémom bolo ale, že kurzor myši na lokálnom počítači nebol synchronizovaný s kurzorom myši na serveri, čo spôsobovalo značné problémy pri práci so serverom. Problém sme odstránili začatím používania programu *Remote Desktop*, ktorý je bežne dostupný v operačnom systéme *Windows*.

Samotné pripojenie sa na server cez program *Remote Desktop* je uskutočnené prostredníctvom jedného portu, preto musel byť aspoň jeden port voľný. Doposiaľ sme mali zo štyroch dostupných portov dva vyhradené, takže na túto činnosť sme si vyhradili tretí port, konkrétne port s číslom *443*. Takisto bolo treba povoliť prístup pre jednotlivých členov tímu, takže sa vytvorilo samostatné konto pre každého člena tímu, cez ktoré sa na server prihlasuje.

Následne bol spísaný jednoduchý postup na pripojenie sa na server a odoslaný všetkým členom tímu. Pripojenie sa na server je jednoduché. Klikne sa na tlačidlo štart, vyhľadá sa program *Remote Desktop Connection* a spustí sa. Po spustení programu sa ako názov počítača zadá presná adresa servera a port, cez ktorý sa chceme na server pripojiť, konkrétne sa zadá nasledovný údaj: „*147.175.159.147:443*“. Následne sa už iba zadá meno a heslo používateľa a klikne sa na pripojenie sa na server.

Následne na serveri, ak je aplikácia maximalizovaná, v hornej časti obrazovky je zobrazená lišta, na ktorej sa nachádzajú štandardné tlačidlá pre minimalizovanie, maximalizovanie a zavretie programu. Čo sa týka klávesových skratiek, tak kombinácia „*Ctrl + Alt + Del*“ vykoná príslušnú akciu na lokálnom počítači používateľa, ale kombinácia „*Ctrl + Shift + Esc*“ spustí *task manager* na priamo na serveri.