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 4
2.1 Účel dokumentu 4
2.2 Štruktúra dokumentu 4
2.3 Zadanie projektu 4
2.4 Ciele projektu 4
3 Šprint 0 – Antónia 6
3.1 Zaznamenávanie aktivity používateľa 6
3.2 Možné smery odporúčaní 7
4 Šprint 1 – Frederika 8
4.1 Analýza programu *PerConIK* 8
4.1.1 Biometria 9
4.1.2 Architektúra 9
4.1.3 Zber biometrických údajov 10
4.2 Dátový model programu *PerConIK* 12
4.2.1 Klientská časť aplikácie 12
4.2.2 Serverová časť aplikácie 12
4.3 Pridanie emocionálneho stavu používateľa do logov a dátového modelu 15
4.3.1 Analýza 15
4.3.2 Návrh 16
4.3.3 Implementácia 17
4.4 Analýza emocionálneho stavu používateľa 17
4.4.1 Hnev 17
4.4.2 Únava 18
4.4.3 Stres 19
4.4.4 Radosť 21
5 Šprint 2 - Henrieta 22
5.1 Pridanie funkcionality do programu *PerConIK* 22
5.1.1 Logovanie emócií 22
5.1.2 Pridanie konvertoru pre emócie 22

5.1.3 Pridanie generovania míľnika 22
5.2 Analýza dátových štruktúr vstupných zariadení 24
5.2.1 Analýza dátových štruktúr myši 24
5.2.2 Analýza dátových štruktúr klávesnice 25
5.3 Návrh výpočtu metrík pre myš 25
5.3.1 Sledovanie myši pomocou programu *PerConIK* 26
5.3.2 Metriky pre sledovanie emočného stavu používateľa 26
5.4 Návrh výpočtu metrík pre klávesnicu 27
5.4.1 Sledovanie klávesnice pomocou programu *PerConIK* 27
5.4.2 Metriky pre sledovanie emočného stavu používateľa 28
5.5 Návrh modelu používateľa 29
5.6 Implementácia modelu používateľa 30
5.6.1 Pridané entity do dátového modelu 30
5.6.2 Vytváranie vektoru metrík ako model používateľa 30
6 Šprint 3 – Izabela 33
6.1 Doplnenie modelu používateľa 33
6.2 Metódy rozpoznávania modelu 33
6.2.1 Naivná metóda 33
6.2.2 Metóda založená na Manhattanovskej vzdialenosti vektorov 34
6.2.3 Metóda váženej kosínusovej podobnosti vektorov 34
6.2.4 Metóda neurónovej siete 35
6.3 UML diagram doposiaľ vytvorenej aplikácie 39
7 Šprint 4 - Karolína 40
7.1 Odporúčanie pre používateľa 40
7.1.1 Vybranie odporúčania 40
7.1.2 Služba vracajúca odporúčanie 40
7.2 Implementácia serverovej časti 41
7.2.1 Spustenie služby *PerConIK* na serveri 41
7.2.2 Kompozícia serverových komponentov 41
7.3 Vyhodnotenie výsledkov 41
7.3.1 Predspracovanie dát 41

7.3.2 Analýza výsledkov 42
7.4 Pripojenie sa na server cez *remote desktop* 44
8 Sumarizácia po zimnom semestri 46
8.1 Sumár vykonanej práce 46
8.1.1 Analýza 46
8.1.2 Návrh 46
8.1.3 Implementácia 47
8.1.4 Testovanie 48
8.2 Celkový obraz projektu po zimnom semestri 48
8.3 Backlog – čo bolo v zimnom semestri urobené 49
9 Šprint 5 – Lýdia 51
9.1 Zlepšenie priestorovej efektivity ukladania dát do databázy 51
9.2 Implementácia reprezentácie v pamäti počítača 52
9.2.1 Rozhranie IVectorRepresentation 52
9.2.2 ReadVectorRepresentation 52
9.2.3 WriteVectorRepresentation 53
9.3 Zapuzdrenie prístupu k modelu používateľa 53
9.4 Rozšírenie tabuľky UserModel 53
9.5 Dynamické pýtanie sa používateľa na emocionálny stav 53
9.5.1 Trieda WorkingWatcher 53
10 Šprint 6 – Marína 54
10.1 Implementácia DailyUserModelManager 54
10.2 Redukcia počtu emócií 54
10.3 Využitie Naive Bayes klasifikátora pri klasifikácií emócií 54
10.4 Analýza dát pre použitie Naive Bayes klasifikátora 56
10.5 Metóda rozpoznávania využívajúca SVM 58
10.6 Nové grafické používateľské rozhranie pýtača 58
10.7 Zmenšenie počtu sledovaných emócií 59
10.8 Rozšírenie dynamického pýtania sa používateľa na emocionálny stav 59
10.8.1 Trieda Asker 59
10.8.2 Trieda StaticAsker 60

10.8.3 Trieda DynamicAsker 60
11 Šprint 7 – Nikita 61
11.1 Odporúčania 61
11.1.1 Kategórie odporúčaní 61
11.1.2 Výber kategórie 61
11.2 Implementácia Ukončenia pýtania 62
11.3 Implementácia n-minútového sledovania používateľa 62
11.4 Aplikácia Fiit.Team10.CalculateResults 62
11.5 Aplikácia Fiit.Team10.ResultPrezenter 63
11.6 Prehrávač hudby 63
12 Šprint 8 – Olívia 63
12.1 Návrat dát pre odporúčanie klientovi 63
12.2 Pridanie stratégie vyhodnotenia emócie 64
12.3 Akceptačné testy 64
12.3.1 Logovanie 64
12.3.2 Modelovanie 66
12.3.3 Rozpoznávanie 68
12.3.4 Odporúčanie 69

Poďakovanie

Týmto by sme chceli vyjadriť vďaku Doc. Ing. Daniele Chudej, PhD. za jej cenné rady pri vypracovávaní tímového projektu.

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.

2.4 CIELE PROJEKTU

Emocionálny stav a celková psychická nálada ovplyvňuje našu výkonnosť a aj faktory, ktoré je možné merať pomocou bežne dostupných prostriedkov počítača ako klávesnica, myš a kamera. Človek, ktorý je v dobrej nálade má inú frekvenciu písania ako keď ju má zlú a je apatický. Únava a psychická vyčerpanosť ovplyvňuje aj žmurkanie, ale aj napríklad to, či oči sledujú pozíciu kurzora. Takéto meranie emocionálneho stavu nijako neobťažuje používateľa na rozdiel od klasických biometrických údajov ako tepová frekvencia či EEG.

Sledovanie, vyhodnocovanie a zaznamenávanie citov nám môže pomôcť pri mnohých veciach. Od výskumu použiteľnosti aplikácie, kde môžeme sledovať, aký má daná aplikácia

vplyv na používateľa. Vytvorenie spätnej väzby pre aplikácie a vývojové prostredia. Inou možnosťou je reportovanie spätnej väzby na prácu iných ľudí, napríklad frustráciu pri čítaní cudzích zdrojových kódov, prideľovanie krátkodobých úloh v práci na základe aktuálneho emocionálneho stavu.

Využitie sledovania emócií vidíme aj vo vytvorení spätnej väzby používateľovi, napríklad zobrazovať upozornenia, že je čas spraviť si prestávku, zmeniť typ práce a podobne. Takisto môže byť zaujímavé sledovať vplyv iných faktorov na emócie človeka, napríklad počúvanie hudby počas práce.

Cieľom našej práce je rozšíriť projekt *PerCoIK*, tak aby umožňoval zaznamenávanie aktivity používateľa na jeho počítači či notebooku. Následne sa používateľove aktivity odošlú na server, kde sa spracujú. Toto spracovanie predstavuje uloženie aktivít a následné vypočítanie aktuálneho emocionálneho stavu používateľa. Následne systém odporučí používateľovi najvhodnejšiu činnosť podľa vypočítaného emocionálneho stavu.

Našimi hlavnými cieľmi je to, aby systém mal čo najvyššiu úspešnosť detekcie používateľovej emócie a to, aby ponúknuté odporúčania pozitívne ovplyvnili stav používateľa a postupom času nestrácali svoj účinok.

3 ŠPRINT 0 – ANTÓNIA

3.1 ZAZNAMENÁVANIE AKTIVITY POUŽÍVATEĽA

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

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*

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:

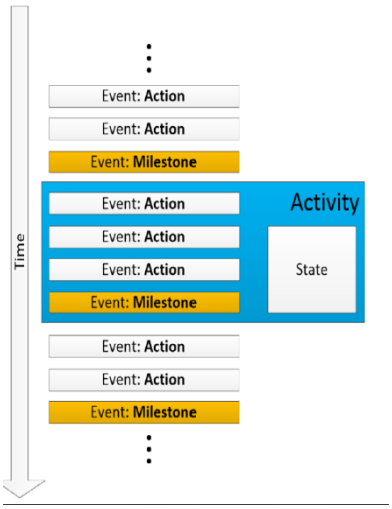
- *akciu* – predstavuje bežnú činnosť v pracovnej aktivite používateľa,
- *míľnik* – významná – spúšťa zber a odosielanie údajov o aktivite.

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*.

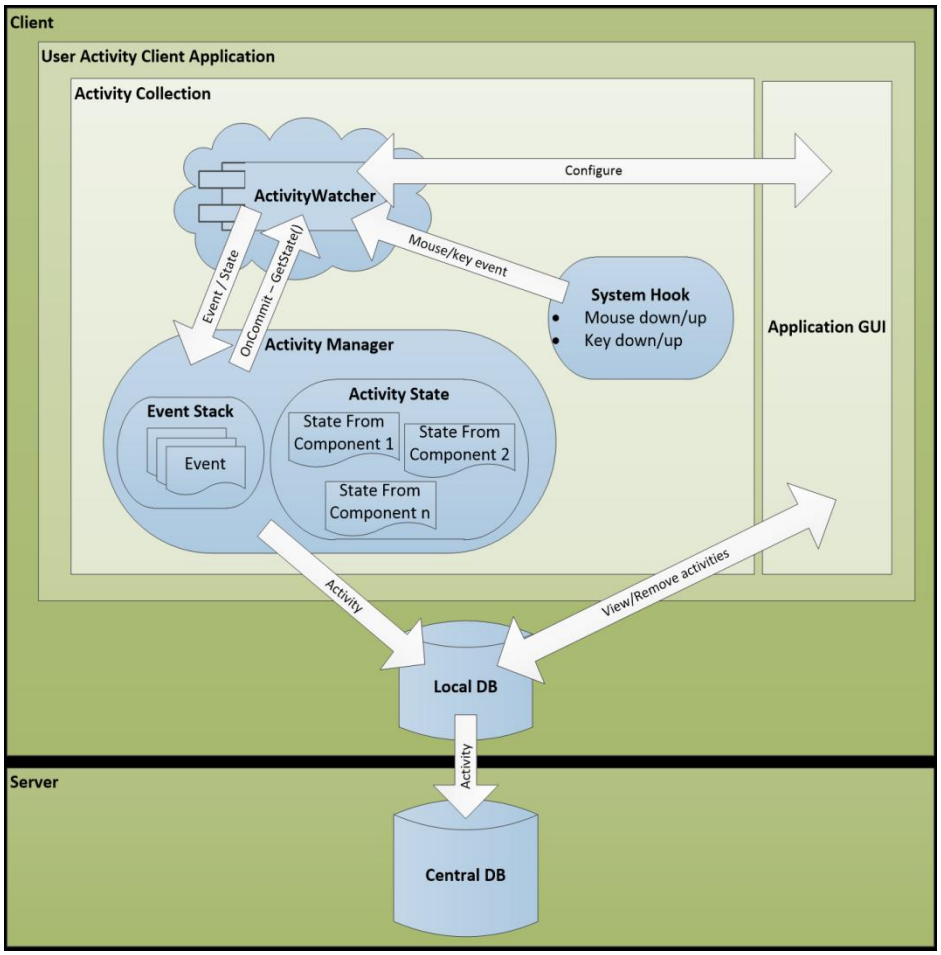
Nasledujúci obrázok zobrazuje sledované údaje a možnosti ich získania.

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.

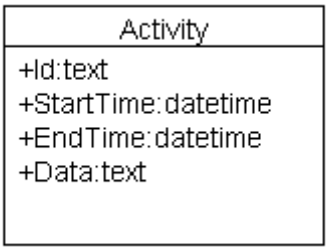
Medzi sledované údaje patria: začiatočná pozícia, koncová pozícia, čas pohybu, prejdená vzdialenosť, typ stlačeného tlačidla, hodnota otočenia kolieska.

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*

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.

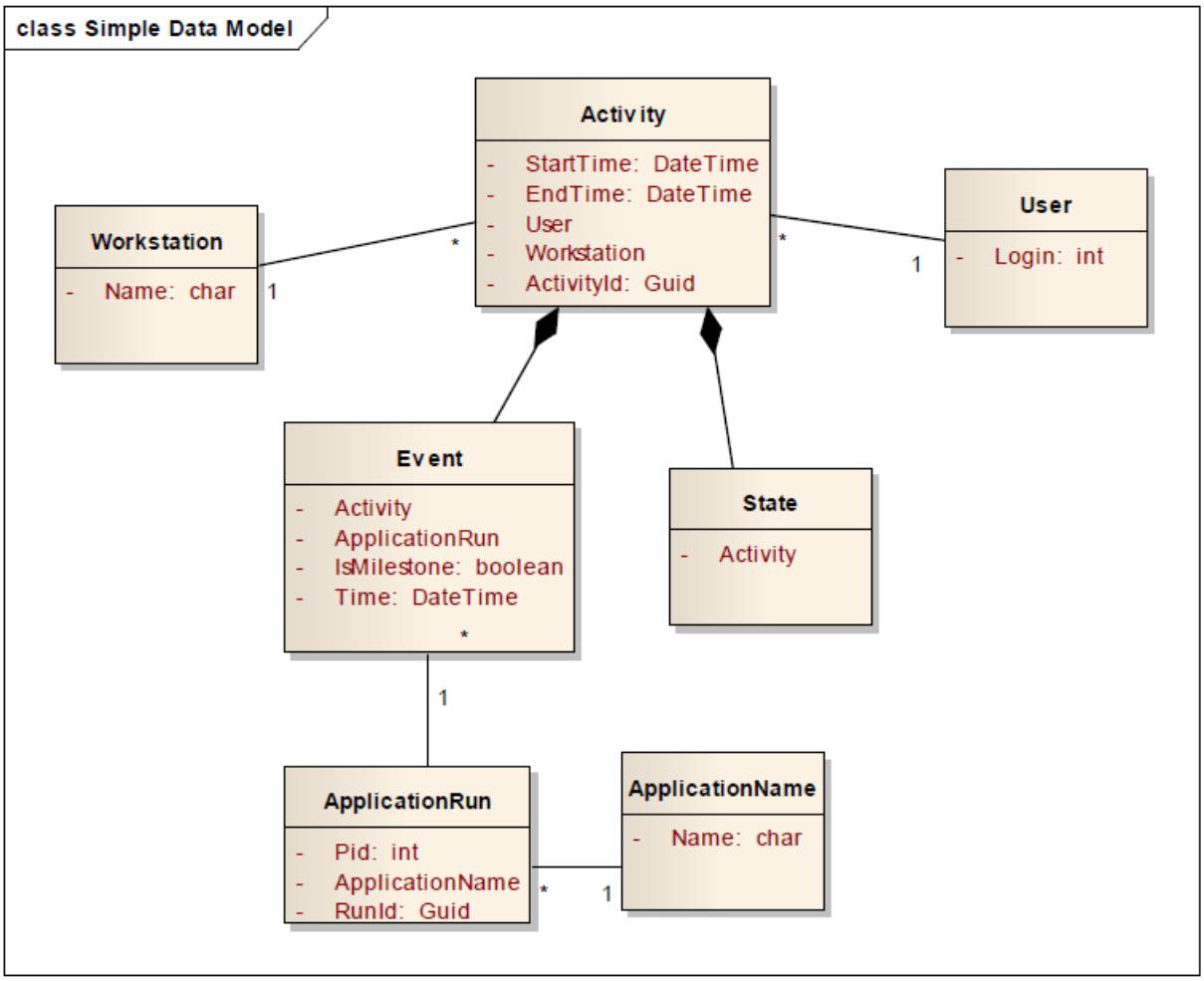


OBRÁZOK 4 - DÁTOVÝ MODEL LOKÁLNEJ DATABÁZY

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

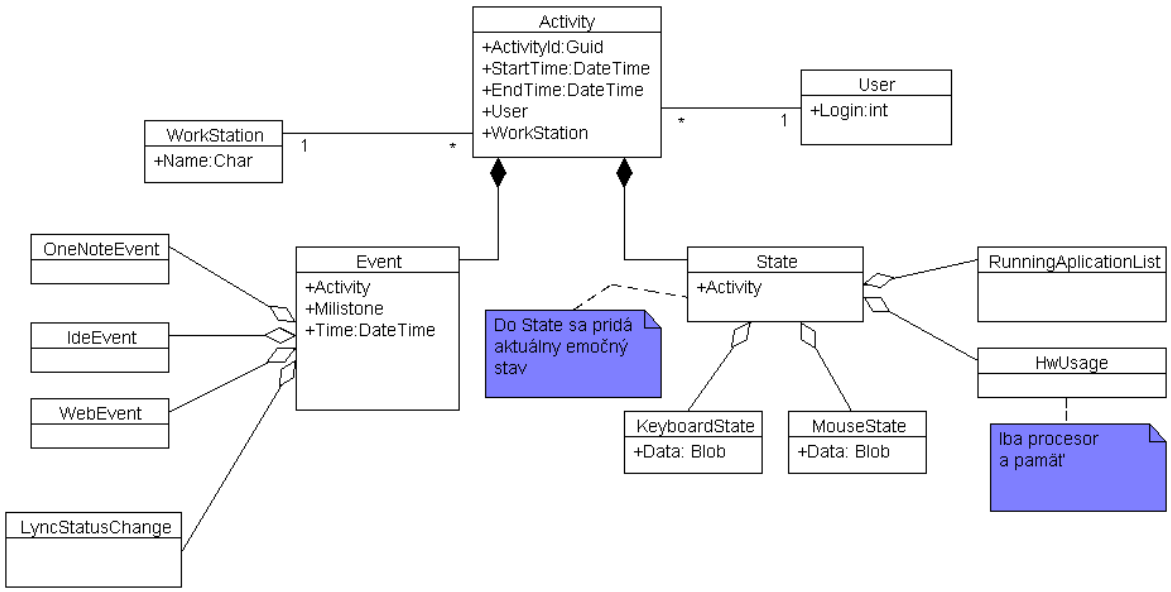
Obrázok 5 znázorňuje logický model serverovej časti aplikácie, ktorá používa *MS SQL* server.



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.

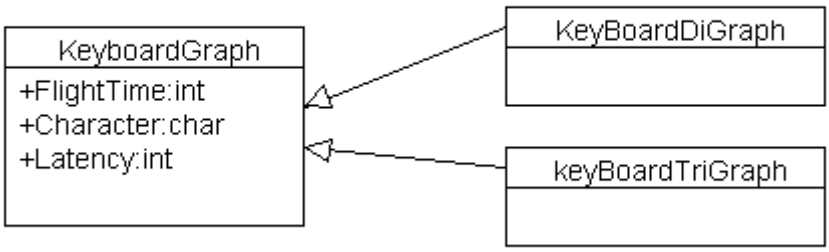
Na ďalšom obrázku je znázornený logický model serverovej časti aplikácie ktorý abstrahuje od detailov *Eventu* (stavu aplikácií) a zameriava sa na spôsob ukladania klávesnice a myši.



OBRÁZOK 6 – DETAILNÝ LOGICKY MODEL DATABÁZY

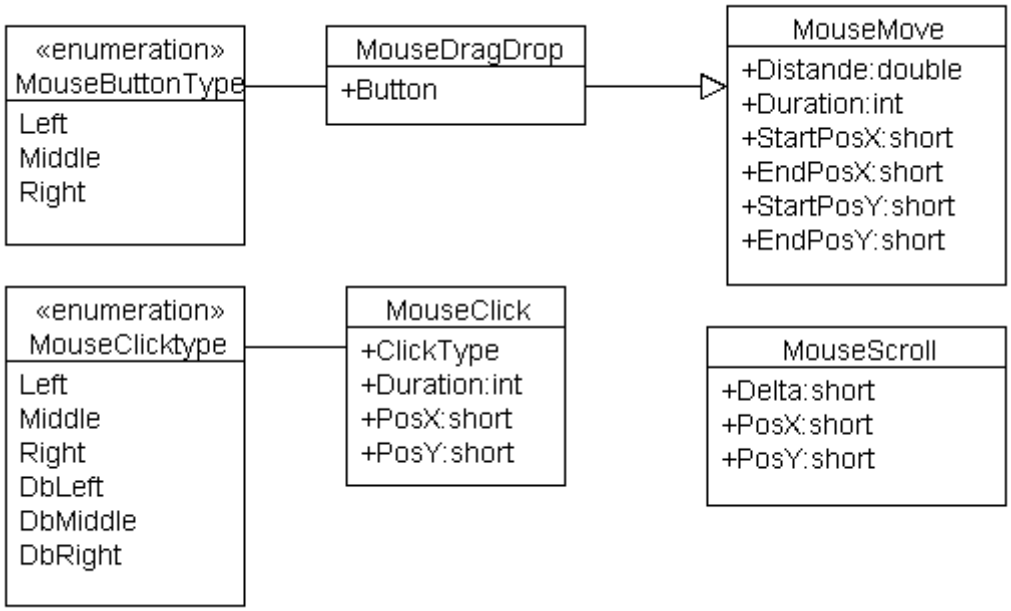
Entity uchovávajúce stav klávesnice a myši sú agregované v stave (entita *State*).

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

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.

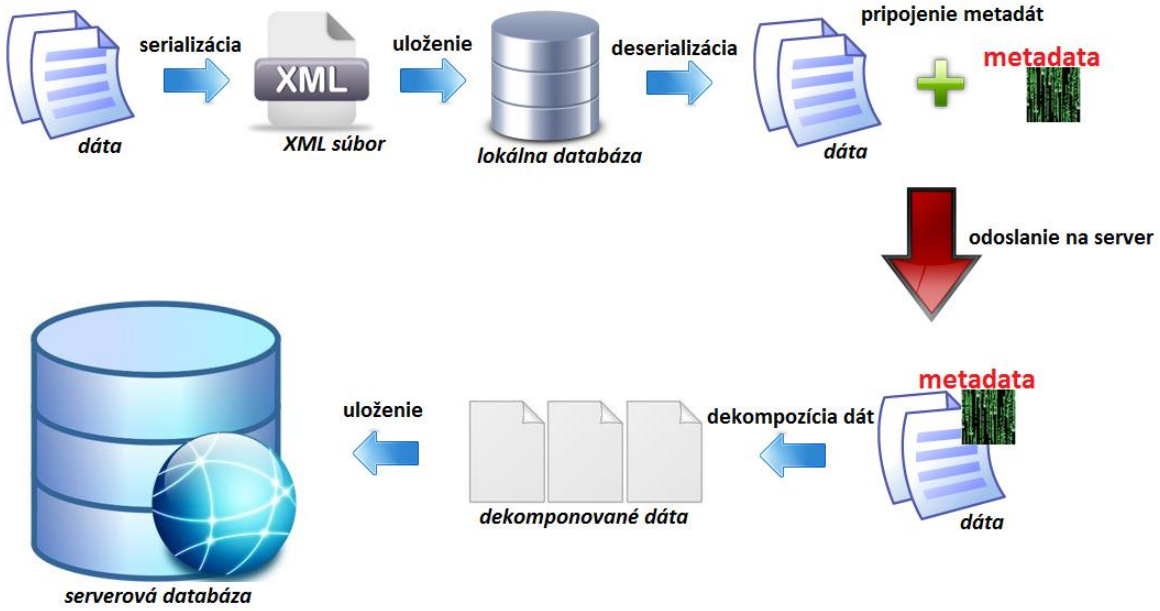
4.3.1 ANALÝZA

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.



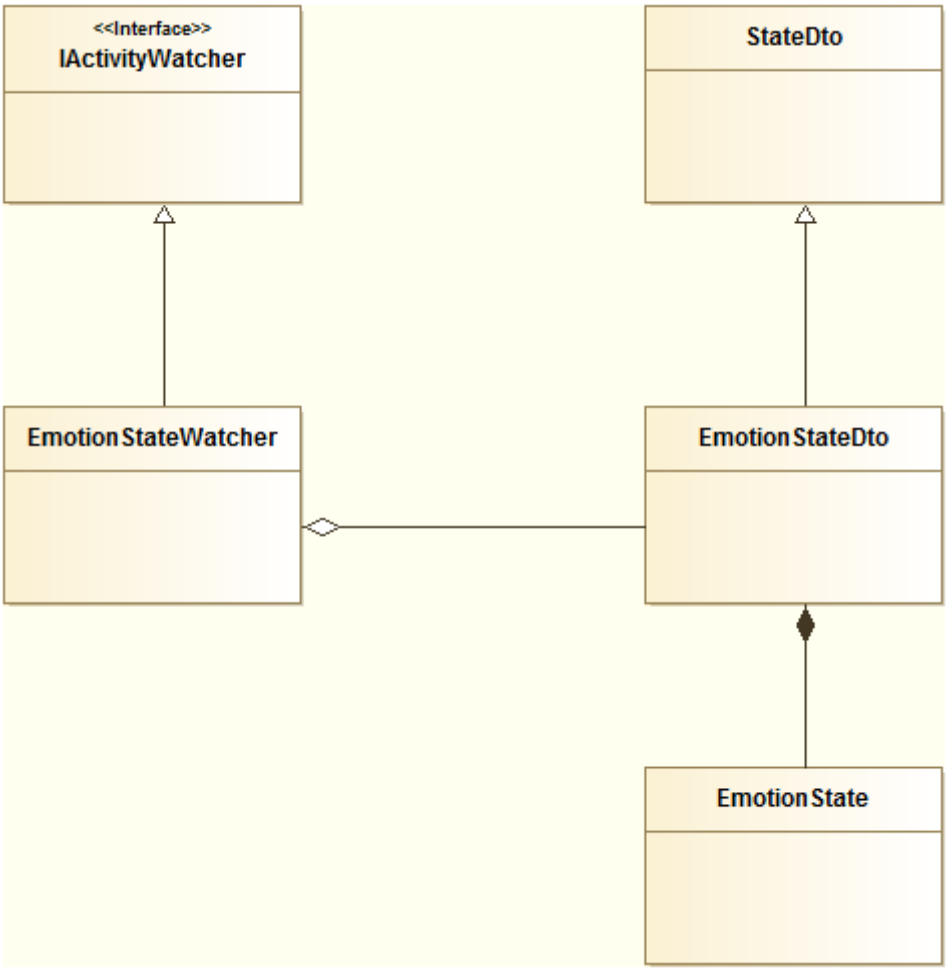
OBRÁZOK 9 - TOK ZÍSKANÝCH DÁT O POUŽÍVATEĽOVI

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

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.

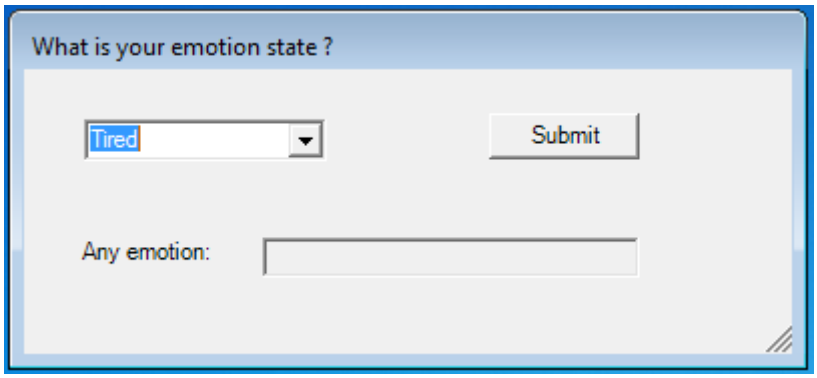
Pri oceneniach sa ľuďom vždy zvýši dobrá nálada nezávisle na počte ocenení. Pri zvyšných spúšťačoch šťastia je potrebné udržiavať širokú paletu týchto spúšťačov, aby sa predišlo efektu rýchlej straty emócie radosti a dobrej nálady z dôvodu príliš veľkej frekvencie prežitia konkrétneho spúšťača.

5 ŠPRINT 2 - HENRIETA

5.1 PRIDANIE FUNKCIONALITY DO PROGRAMU *PERCONIK*

5.1.1 LOGOVANIE EMÓCIÍ

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



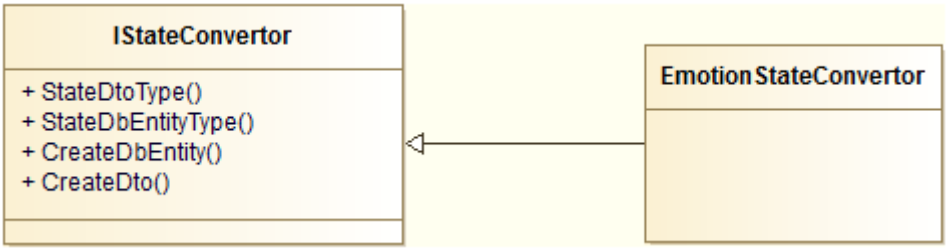
OBRÁZOK 11 - FORMULÁR PRE ZADÁVANIE EMOCIONÁLNEHO STAVU

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.

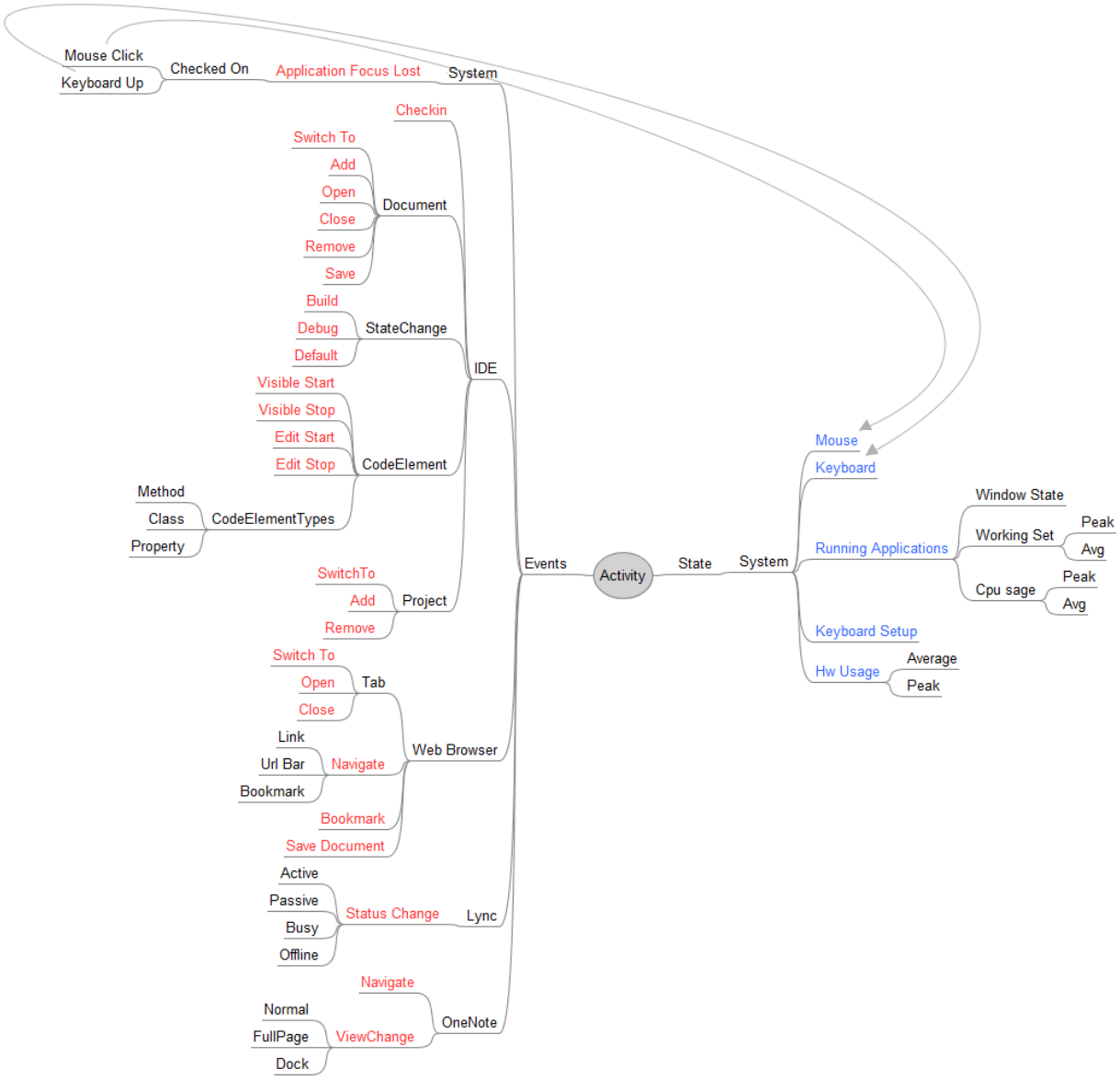


OBRÁZOK 12 - KOVERTOR PRE EMÓCIE

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.



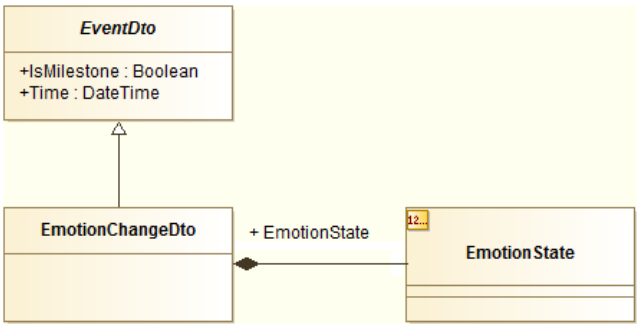
OBRÁZOK 13 - SLEDOVANÉ UDALOSTI A STAVY

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

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

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

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

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

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

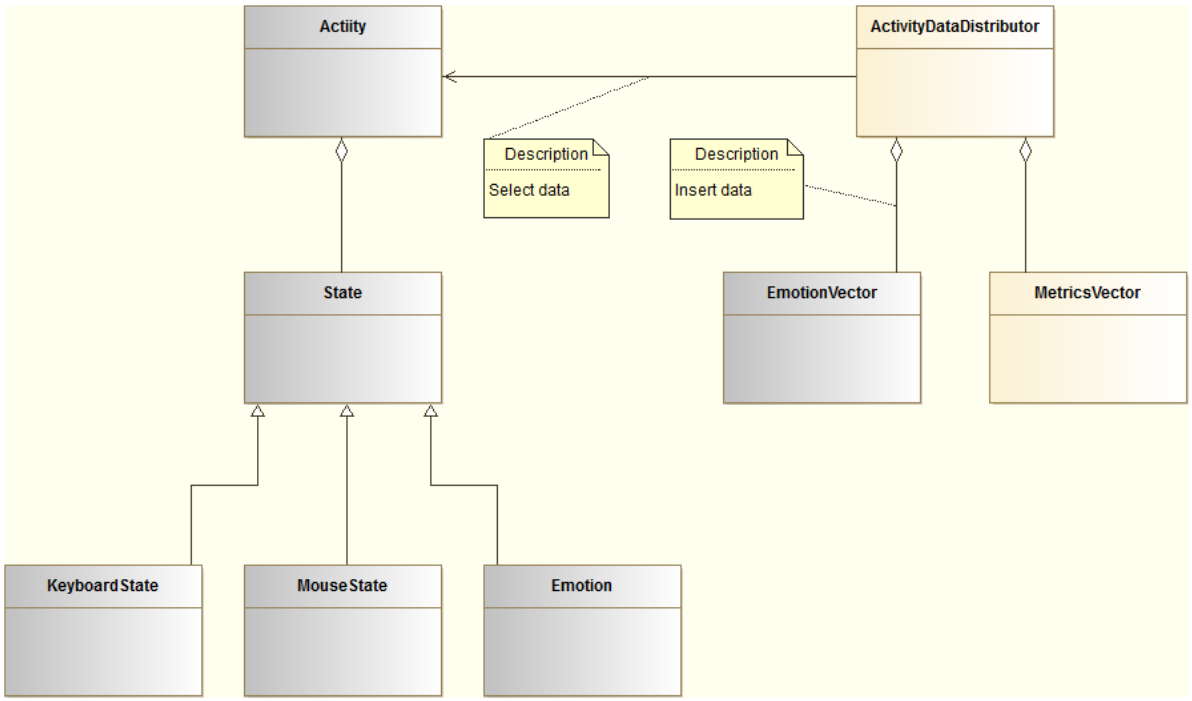
Entita v sebe ukladá emóciu zadanú používateľom a aj vypočítanú hodnotu, obe hodnoty v čase vzniku nemusia byť uvedené.

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

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

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:

$$\bar{e_i} = \frac{1}{|E|} \sum_{v \in E} e_{v,i}$$

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

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, $q \in E$ je vektor emócie z modelu používateľa a w je vektor váh vypočítaný z modelu používateľa, vypočítame:

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

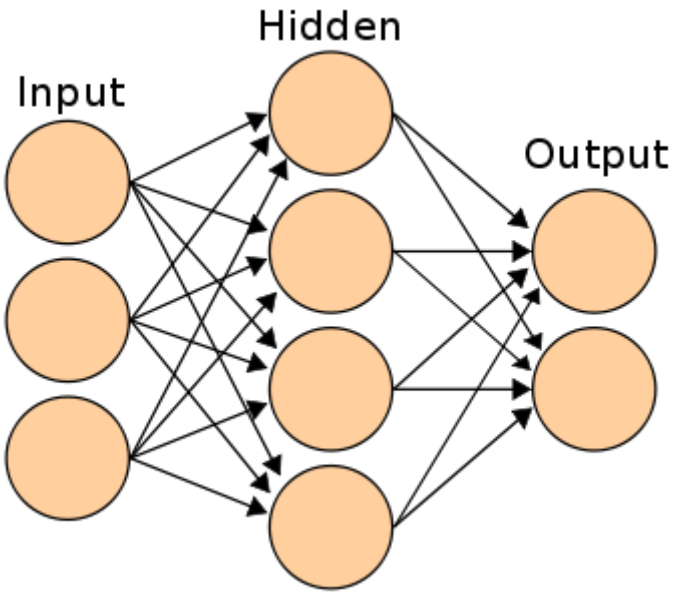
Vypočítaná podobnosť vektorov je v rozsahu <*-1,1*> preto ju je ešte potrebné normovať do rozsahu 0 až 1.

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

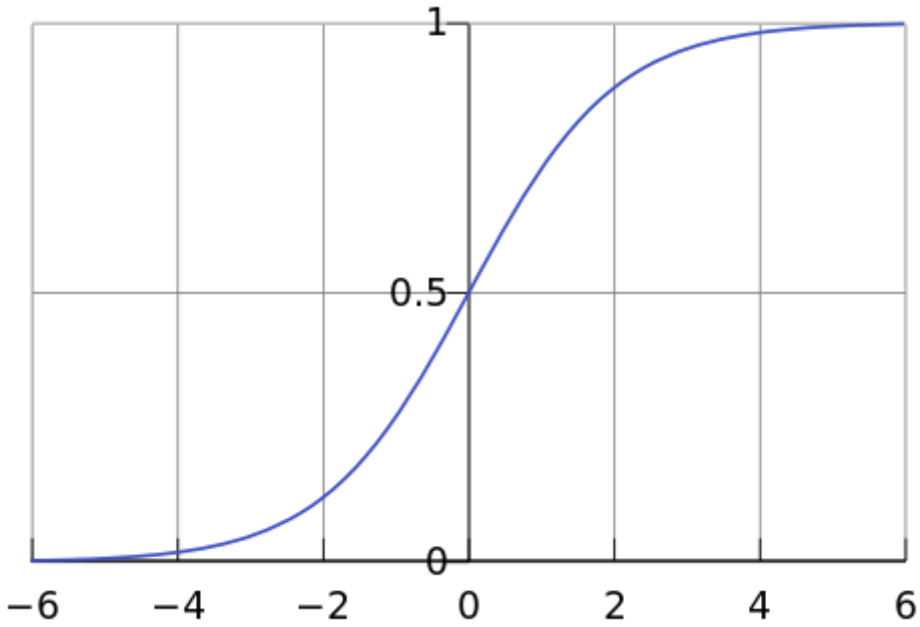
Všeobecne sa neurónová sieť skladá z troch typov vrstiev: vstupná vrstva, skrytá vrstva a výstupná vrstva. Skrytá vrstva sa ešte môže ďalej skladať z viacerých podvrstiev, čo je ale zvyčajne viac kontraproduktívne ako prínosné, preto sme v našej implementácii použili len jednu skrytú vrstvu.

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.

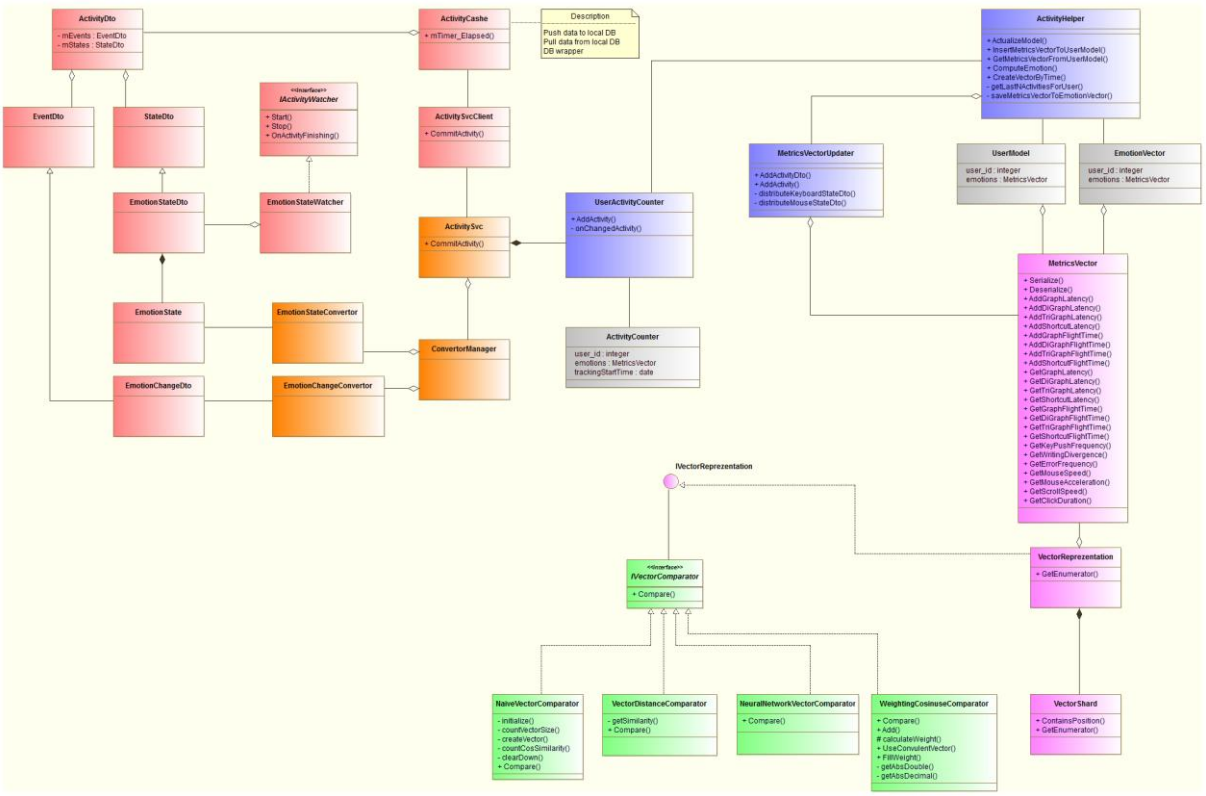
Čo sa týka implementácie, rozdelili sme kód ohľadne trénovania neurónovej siete na tri hlavné triedy

- *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



OBRÁZOK 18 - UML DIAGRAM TRIED AKTUÁLNEHO STAVU APLIKÁCIE

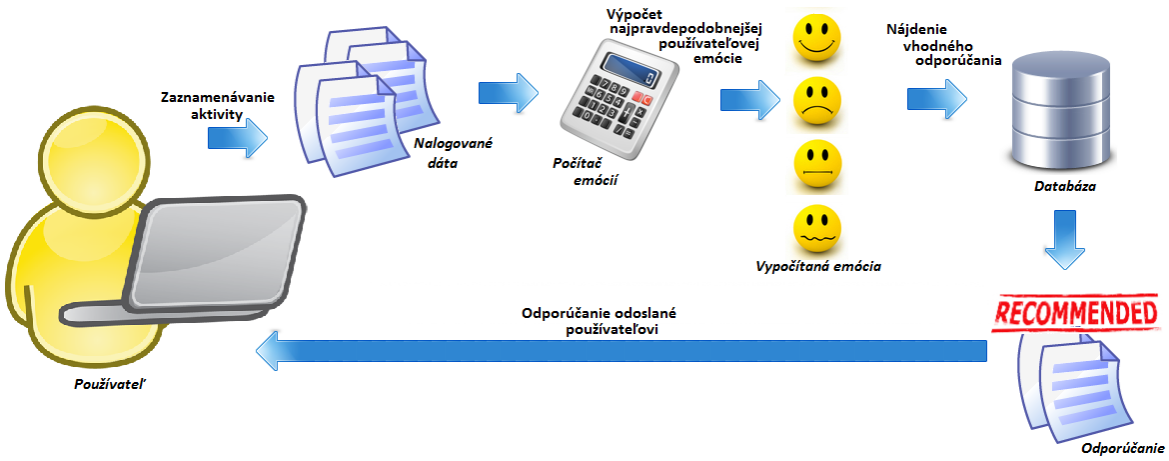
Popis farebného usporiadania diagramu:

- 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

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

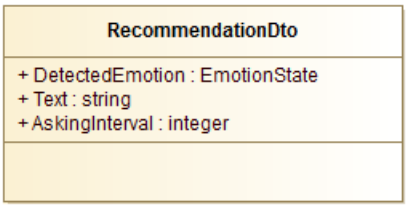
Po príchode dopytu od klientskej časti programu, ktorá si pýta odporúčanie sa požiadavka spracuje triedou *Recommender*. Táto trieda pozrie či sa v databáze nachádza vypočítaná hodnota emócie používateľa v nedávnom časovom období napr. pol hodiny. Ak áno, tak vyberie posledný takýto záznam a zistí, aká 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

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

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*

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.

8 SUMARIZÁCIA PO ZIMNOM SEMESTRI

8.1 SUMÁR VYKONANEJ PRÁCE

V tejto časti je uvedená sumarizácia práce na analýze, návrhu, implementácii a testovania zadania uvedeného v úvode dokumentu.

8.1.1 ANALÝZA

Na začiatku sme analyzovali emócie, ako sa prejavujú a ako ich dokážeme rozlišovať. V prvej fáze sme chceli zistiť aké emócie dokážeme rozpoznávať pomocou klávesnice a myši, preto sme vybrali širšiu paletu emócií pre skúmanie a to základných šesť emócií: strach, hnev, radosť, smútok, znechutenie a prekvapenie. Ďalej sme chceli otestovať rozpoznávanie týchto troch emócií (stavov používateľa): stres, nervozita a únava, nakoľko sme predpokladali ich možné zachytenie pomocou myši a klávesnice. Nakoniec sme pridali neutrálny stav, ktorý používateľ zadá v prípade, ak nepociťuje žiadnu z uvedených emócií tak, aby dokázal určiť jednu konkrétnu z nich.

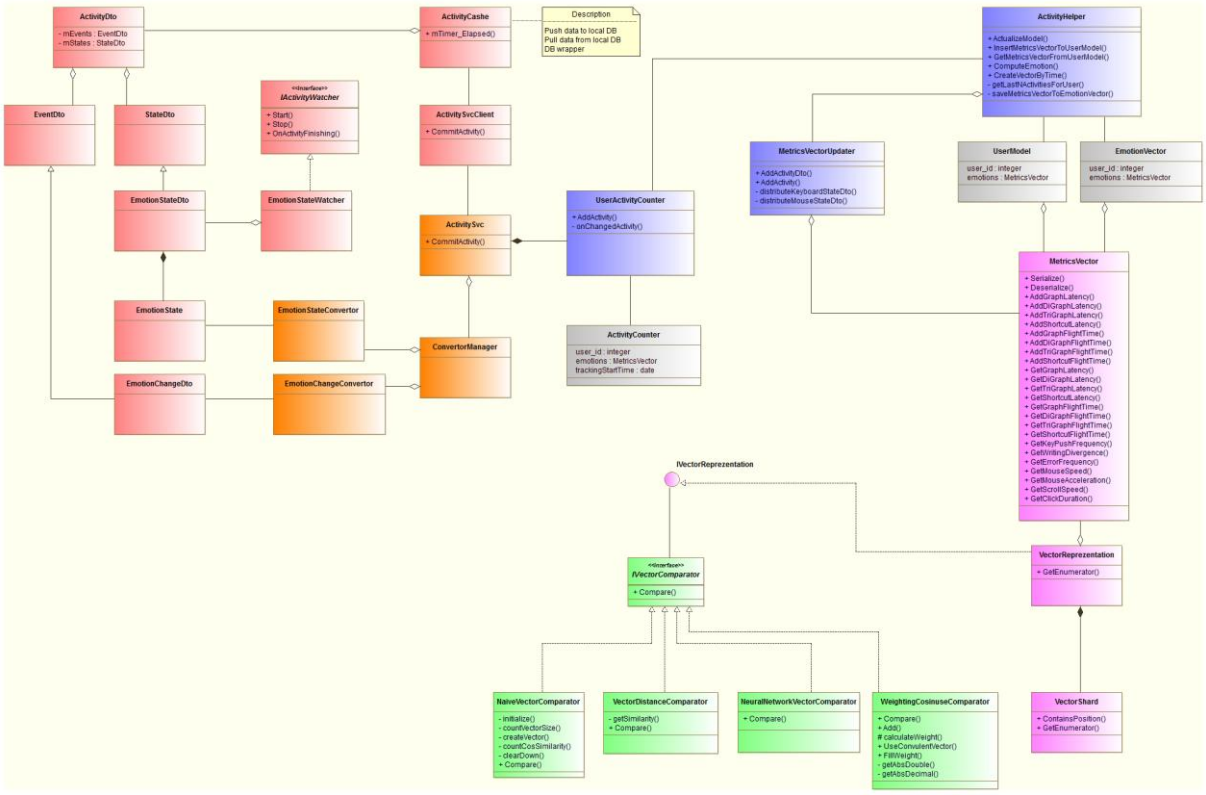
V ďalšom kroku sme analyzovali program *PerConIK*, nakoľko sme naň našou prácou nadväzovali bolo potrebné sa s ním oboznámiť, do takej miery aby sme vedeli k tomuto riešeniu pripojiť našu prácu.

8.1.2 NÁVRH

Na začiatku bolo potrebné navrhnúť spôsob logovania emocionálneho stavu používateľa. Následne bolo potrebné vytvoriť model používateľa z nalogovaných dát. Pre rozpoznanie sme navrhli štyri porovnávacie algoritmy:

- naivná metóda – porovnávanie vektorov kosínusovou metódou,
- metóda porovnávania vzdialeností vektorov – porovnávanie vektorov na základe *manhattanovskej* vzdialenosti,
- metóda automatického váhovania – metóda založená na váženej kosínusovej podobnosti,
- metóda neurónovej siete – metóda využívajúca natrénovanú neurónovú sieť použitú ako porovnávač vektorov.

Nasledovné komponenty sme navrhli zaintegrovať do serverovskej aplikácie programu *PerConIK* nasledujúcim spôsobom.



OBRÁZOK 21 NAVRHOVANÝ DIAGRAM TRIED SERVEROVSKEJ APLIKÁCIE

8.1.3 IMPLEMENTÁCIA

Implementácia nášho projektu bola realizovaná vo vývojovom prostredí *Visual Studio 2010*. Ako programovací jazyk sme použili *C#*, pretože sme nadviazali na existujúci projekt *PerConIK*, ktorého účelom je logovať aktivitu používateľa pri práci s počítačom.

Klientskú časť aplikácie *PerConIK* sme rozšírili o dialógové okno obsahujúce 12 tlačidiel popisujúcich emocionálny stav používateľa. Emocionálny stav sa následne ukladá do lokálnej databázy ku každému záznamu vytvoreného v časovom intervale platnosti zadanej emócie. Pomocou nich bol implementovaný model ako vektor polí pre každú emóciu.

Serverovská časť bola rozšírená o navrhované triedy, ktoré sú umiestnené v samostatných menných priestoroch. Navrhnuté metódy porovnávania emocionálnych vektorov sme implementovali do samostatných tried implementujúcich rozhranie *IVectorComparator* poskytujúce metódu *Compare* prijímajúcu dva vektory ako argumenty typu *IVectorReprezentation* a vracajúcu ich podobnosť.

Pre potrebu získavania odporúčania pre používateľa zo serverovskej aplikácie sme rozšírili rozhranie *IActivitySvc* metódou *GetRecommendation*. Rozhranie deklaruje metódy, ktoré je možné volať vzdialene, napr. cez sieť.

Týmto sme uzatvorili kruh – zaznamenávanie používateľa, analýza emocionálneho stavu a spätnej väzby pre používateľa v podobe odporúčania.

8.1.4 TESTOVANIE

Kvalitu implementácie komponentov sme testovali prostredníctvom *Unit* testov. Testy boli písané pre všetky triedy zakomponované do výslednej aplikácie.

Výsledky testov sme mohli pozorovať priamo v integrovanom vývojovom prostredí, ako je znázornené na nasledujúcom obrázku.

Result	Test Name	Project
Passed	TestAddGraphLatency	Fiit.Team10.Testing
Passed	TestNumbersInIVectorReprezentation	Fiit.Team10.Testing
Passed	TestNumbersInIvectorReprezentationShard	Fiit.Team10.Testing
Passed	TestCompareMethodNNV	Fiit.Team10.Testing
Passed	TestIVectorReprezentation	Fiit.Team10.Testing
Failed	TestInvertVectors	Fiit.Team10.Testing
Passed	TestSimpleNumbersInMetricsVector	Fiit.Team10.Testing
Passed	TestGetGraphLatency	Fiit.Team10.Testing
Passed	TestCompareMethodNVC	Fiit.Team10.Testing
Passed	TestGetKeyPushFrequency	Fiit.Team10.Testing
Passed	TestMetricsVectorConsistency	Fiit.Team10.Testing
Passed	TestCompareMethodVDC	Fiit.Team10.Testing
Passed	TestForNegativeValuesInIVectorReprezentation	Fiit.Team10.Testing
Passed	TestNotZeroInWCC	Fiit.Team10.Testing

OBRÁZOK 22 UNIT TESTY SERVEROVSKÝCH TRIED

8.2 CELKOVÝ OBRAZ PROJEKTU PO ZIMNOM SEMESTRI

Po zimnom semestri Tímového projektu máme hotový funkčný prototyp, ktorý je schopný vykonávať základné činnosti určené cieľmi projektu.

Úspešne sme rozšírili program *PerConIK* o to, aby sa v zadaných časových intervaloch pýtal používateľa na aktuálny emocionálny stav, a tak bol schopný, spolu s aktivitami, posielať na server aj túto informáciu. Na serveri sa používateľove aktivity uložia do databázy spolu s emocionálnym stavom. Každých desať minút sa uložené aktivity používateľa spracovávajú a počíta sa takzvaný emocionálny vektor. Ten je následne porovnaný s modelom používateľa, ktorý sa skladá z modelov jednotlivých emócií. Vypočítaná emócia je tá, ktorej model je najviac podobný aktuálnym emocionálnym vektorom. Na základe vypočítanej emócie sa vyberie najvhodnejšie odporúčanie. Klientská časť systému si toto odporúčanie preberie a ponúkne ho používateľovi.

V rámci implementovania prototypu sme museli vyriešiť mnoho problémov. V prvom rade sme museli navrhnúť spôsob, akým logovač zistí aktuálny stav používateľa. Ako reprezentovať v systéme emócie a ktoré sú najvhodnejšie na rozpoznávanie. Ako vytvárať

emocionálne vektory a čo majú obsahovať a ako ich reprezentovať. Ako zostaviť a udržiavať model používateľa.

Ďalej sme navrhli tri algoritmy porovnávania modlu používateľa a emocionálneho vektora, s čoho sa dva ukázali ako pomerne úspešné. A navrhnutý mechanizmus odovzdávania a voľby odporúčania.

Všetky tieto návrhy boli implementované a experimentálne overené.

8.3 Backlog – čo bolo v zimnom semestri urobené

Backlog projektu je vytvorený v tabuľkovom editore a je umiestnený na službe *GoogleDocs*. Každý záznam v *backlog*-u obsahuje:

- názov,
- popis,
- odkaz na *User Story* do ktorého logicky patrí.

Odkaz na *User Story* je v podobe čísla, ktoré sa prevedie na *User Story* podľa nasledujúceho obrázku.

User Story	Číslo
Ja ako používateľ som sledovaný	1
Ja ako používateľ som modelovaný	2
Modely sú rozpoznávané	3
Ja ako používateľ dostanem odporúčanie	4

Obrázok 23 Prevod User Story na číslo

Backlog obsahuje všetky navrhované vlastnosti projektu, tie ktoré už boli do projektu zapracované sú vyznačené oranžovou farbou.

User Story	Feature
1	Sledovať používateľa klávesnicou a myšou
1	Odoslať údaje na server
1	Zadať emocionálny stav
1	Analyzovať emócie
2	Analyzovať model emócie
4	Zobraziť odporúčanie
1	Nastavit emocionálny stav
1	Analyzovať formát získaných dát
2	Doplniť emotion vector do databázy
2	Prepísať objekt VectorRepresentation na "unsafe code"
2	Urobiť export získaných dát do formátu csv
2	Upraviť databázu pridaním stĺpcov "zadaná emócia" a "rozpoznaná emócia" do tabuľky MetricsVector
4	Vybrať odporúčanie na základe emocionálneho stavu

2	Vytvoriť schému databázy na serveri a spustiť skript
3	Získať spätnú väzbu od používateľa
3	Implementovať komparátor metódou neurónovej siete
3	Implementovať komparátor metódou porovnávania vzdialeností vektorov
3	Implementovať komparátor naivnou metódou
3	Implementovať komparátor metódou rozhodovacích stromov
3	Implementovať komparátor metódou support vector machine
3	Implementovať komparátor metódou automatického váhovania
4	Analyzovať a implementovať vzdialené volanie služby na serveri
3	Vytvoriť váhy pre komparátory
2	Doplniť model používateľa
2	Vytvoriť tabuľku ActivityCounter a CalculatedEmotion
2	Previesť vektor emocií do listu
1	Vynulovať časovač, keď používateľ manuálne nastaví emóciu
1	Implementovať možnosť nastavenia intervalu pýtania sa
1	Upraviť míľnik tak, aby sa generoval vždy pri zadaní emócie
4	Implementovať dotazovanie sa na server
1	Pridať atribút "verzia" do binárnych dát
1	Dorobiť vytváranie modelu po 10 minútových intervaloch po zmene emócie
4	Zobraziť vtip pri zlej nálade
2	Vytvoriť tabuľku pre testy komparátorov
4	Integrovať serverové komponenty
1	Implementovať logovanie spustených aplikácií
2	Zahrnúť nalogované dáta zo spustených aplikácií do modelu používateľa
4	Implementovať triedu Recommender
4	Naplniť tabuľku odporúčaní
4	Zobraziť štatistiky priebehu emócií za určité obdobie
1	Upraviť používateľské rozhranie pýtača
1	Vytvoriť nové GUI na zadávanie emócií
3	Do tabuľky TemporaryCalculatedEmotions dopísať ku 4 komparátorom 4 double (dokopy)
2	Skontrolovať aplikácie tak, aby sa dali použiť na serveri: generovanie modelu používateľa, generovanie emocionálnych vektorov
3	Implementovať aplikáciu – UserModel + EmotionVector => TemporaryCalculatedEmotion

Jednotlivé úlohy, ktoré mali byť zapracované do projektu, boli presunuté do *sprint backlogu*. V *sprint backlogu* bola farbou vyznačená ich priorita podľa nasledujúceho obrázku.

Priorita	Farba
Nízka priorita	
Stredná priorita	
Vysoká priorita	

Obrázok 24 Priorita úloh v Šprint backlogu

Sprint backlog pre prvý šprint s názvom „Frederika“ má nasledujúci obsah.

Pre prehľadnosť bola vynechaná časť popis.

Feature	Task	Story point
Zadať emocionálny stav	Zistiť, ako sa vytvára XML z dátového objektu a zistiť, ako sa používateľ autorizuje	5
	Zistiť, ako sa z XML vytvára dátový objekt na serveri	3
	Pridať zadanú emóciu do XML	3
	Pridať zadanú emóciu do SQLite	3
Sledovať používateľa klávesnicou a myšou	Zistiť, ako sa vkladá dátový objekt do MS SQL	5
	Zistiť, ako získať dátový objekt z SQLite	3
	Zistiť, čo trackuje PerConIK	5
Odoslať údaje na server	Zistiť, čo je milestone v PerConIKu a ako pridať vlastné	5
	Zistiť, ako hádzať logy z PerConIKa na server	5
Analyzovať emócie	Analyzovať databázu	13
	Analyzovať stres + čo odporučiť	5
	Analyzovať hnev + čo odporučiť	5
	Analyzovať únavu + čo odporučiť	5
	Analyzovať radosť + čo odporučiť	5

OBRÁZOK 25 SPRINT BACKLOG PRE ŠPRINT "FREDERIKA"

9 ŠPRINT 5 – LÝDIA

9.1 ZLEPŠENIE PRIESTOROVEJ EFEKTIVITY UKLADANIA DÁT DO DATABÁZY

Ukladanie emocionálnych vektorov do databázy je realizované serializáciou triedy *EmotionVector*. Táto trieda sa skladá zo štruktúr typu *Dictionary<string, int[]>*. Pri serializácii tak dochádza k ukladaniu nielen užitočných dát, ale aj názvov tlačidiel, dvoj-tlačidiel, troj-tlačidiel a klávesových skratiek. Reprezentácia dát prislúchajúcich k jednotlivým tlačidlám je realizovaná prostredníctvom početností, avšak mnohé prvky majú početnosť 0 a keďže táto hodnota sa dá chápať ako základná, tieto prvky nenesú žiadnu užitočnú informáciu.

Pre výhodnosť optimalizácie boli vypracované tabuľky opisujúce úsporu miesta.

Používateľ 1.

Emócia	*Anger*	*Any*	*Disgust*	*Fear*	*Happiness*	*Nervous*	*Normal*	*Sadness*	*Stressed*	*Surpirse*	*Tired*	Spolu
Pôvodná veľkosť	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	16714,84
Nová veľkosť	72,0625	100,414	22,5234	0,03906	11,35156	112,492	298,03	34,5937	48,6640	10,2187	142,632	853,031
Zmenšenie	21,0863	15,13265	67,46445	38900	133,861	13,50788	5,09843	43,92502	31,22492	148,7003	10,65345	19,59464

Používateľ 2.

Emócia	*Anger*	*Any*	*Disgust*	*Fear*	*Happiness*	*Nervous*	*Normal*	*Sadness*	*Stressed*	*Surpirse*	*Tired*	Spolu
Pôvodná veľkosť	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	16714,84
Nová veľkosť	17,64063	18,13281	77,03125	0,835938	78,1875	49,58594	321,0625	1,21875	13,16406	6,546875	135,0938	718,5
Zmenšenie	86,13818	83,80009	19,72617	1817,757	19,43445	30,6444	4,732821	1246,795	115,4303	232,1002	11,24798	23,26353

Používateľ 3.

Emócia	*Anger*	*Any*	*Disgust*	*Fear*	*Happiness*	*Nervous*	*Normal*	*Sadness*	*Stressed*	*Surpirse*	*Tired*	Spolu
Pôvodná veľkosť	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	1519,531	16714,84

Nová veľkosť	14,51563	0	16,625	7,125	23,53125	22,05469	68,03906	33,77344	30,97656	0,9375	28,04688	245,625
Zmenšenie	104,6825	inf	91,40038	213,2675	64,57503	68,89834	22,33322	44,9919	49,05422	1620,833	54,17827	68,05025

Pri optimalizácii ukladaných dát využívame prevod párov reťazec-početnosť, na sekvenčné pole počtu záznamov. Pole je vytvorené zreťazením všetkých zaznamenávaných tlačidiel, každému tlačidlu z pôvodnej reprezentácie prislúcha 1000 prvkové pole určujúce dĺžku stlačenia v milisekundách. Pole je zložené z dát s klávesnice,

Latency				*Fligtime*			
Graphs	*DiGraphs*	*TriGraphs*	*Shortcuts*	*Graphs*	*DiGraphs*	*TriGraphs*	*Shortcuts*

nasledované dátami z myši.

			Trvanie stlačenia			Trvanie dvojkliku		
Mouse speed	*Mouse acceleration*	*Scroll speed*	*Left button*	*Middle button*	*Right butotn*	*Left button*	*Middle button*	*Right butotn*

Výsledná reprezentácia v databáze je pole, ktorého párny prvok určuje index umiestnenia a nepárny prvok určuje počet záznamov.

9.2 IMPLEMENTÁCIA REPREZENTÁCIE V PAMÄTI POČÍTAČA

Z dôvodu efektívnosti sme navrhli 2 implementácie reprezentácie dát. Jedna reprezentácia určená na čítanie, ktorá je optimalizovaná na úsporu pamäte a rýchlosť čítania a druhá na zápis a čítanie, ktorá je optimalizovaná na rýchlosť pridávania záznamov. Obe reprezentácie implementujú rozhranie *IVectorRepresentation*.

9.2.1 ROZHRANIE IVECTORREPRESENTATION

IVectorReprezentation je rozhranie, ktoré reprezentuje emocionálne vektory. Poskytuje metódy na iterovanie nad všetkými hodnotami vektorov, na iterovanie len nad nenulovými hodnotami vektorov, metódu na pridávanie aktivít – aktualizáciu vektora, serializáciu na binárne dáta a metódu *JoinNormalizedContent*, ktorá vracia iterátor nad dvoma spojenými vektormi, pričom aspoň jedna z hodnôt na danej pozícii vo vektore je nenulová. Túto metódu využívame pri efektívnom porovnávaní vektorov.

Všetky implementácie rozhrania používajú spoločný formát pre serializáciu dát, ide o dvojice 32-bitových integerov, ktoré reprezentujú index a hodnotu. Tieto dvojice sú uložené za sebou.

9.2.2 READVECTORREPRESENTATION

ReadVectorRepresentation je implementácia rozhrania *IVectorReprezentation*, ktorá slúži iba na porovnávanie vektorov. Je optimalizovaná na rýchle čítanie dát z databázy a ich deserializáciu. Vektor reprezentuje pole dvojíc index-hodnota, ktoré je vzostupne usporiadané podľa indexov, čo umožňuje veľmi pohodlnú iteráciu a prácu s vektorom, pričom je minimalizovaná pamäťová náročnosť. Pre normalizáciu sa používa tabuľka súm, pre

jednotlivé časti vektora a normalizované hodnoty sa počítajú v čase behu. Táto implementácia neumožňuje pridávanie aktivít do vektora, je iba na čítanie (imutable).

9.2.3 WRITEVECTORREPRESENTATION

WriteVectorRepresentation je implementácia rozhrania *IVectorReprezentation*, ktorá slúži ako na čítanie, tak aj na aktualizáciu dát. Je optimalizovaná na rýchly zápis dát. Vektor je reprezentovaný hashovacou tabuľkou, kde kľúč je index do reprezentácie v podobe poľa. Použitím hashovacej tabuľky je možné dosiahnuť pridávanie v čase *O(1)*. Pre normalizáciu sa používa tabuľka súm vypočítavaná pri vkladaní aktivít.

9.3 ZAPUZDRENIE PRÍSTUPU K MODELU POUŽÍVATEĽA

K prístupu k modelu používateľa sa doteraz používali ad-hoc riešenia avšak pre rozšíriteľnosť riešenia je potrebné celý prístup zapuzdriť a ponúknuť iba pár verejných metód. Z tohto dôvodu bola vytvorená trieda *UserModelManager*. Trieda poskytuje metódy *Compare()*, ktorá porovná model používateľa so zadaným vektorom, *AddActivity()* na pridávanie aktivít do modelu a metódu *Save()*, ktorá uloží dáta do databázy.

9.4 ROZŠÍRENIE TABUĽKY USERMODEL

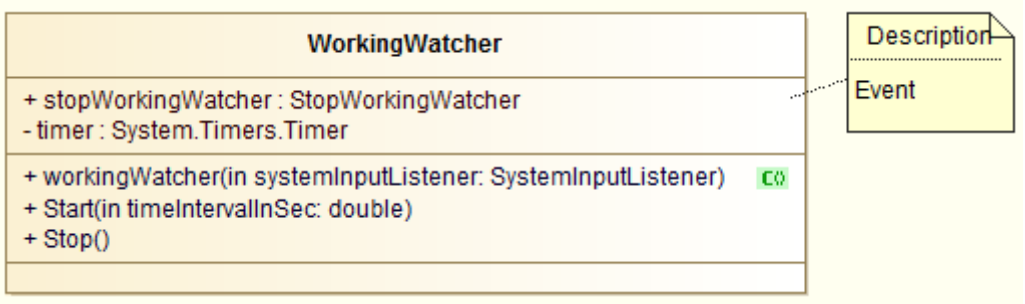
Tabuľka *UserModel* bola rozšírená o pole *UserModelSettingsManager*, ktoré je kontajnerom pre nastavenia jednotlivých porovnávacích metód. Obsahuje metódy pre serializáciu a deserializáciu dát a metódu *Update()*, ktorá sa volá z triedy *UserModelManager* pri každej zmene modelu používateľa. Metóda *Update* zabezpečuje aktualizáciu jednotlivých nastavení.

9.5 DYNAMICKÉ PÝTANIE SA POUŽÍVATEĽA NA EMOCIONÁLNY STAV

9.5.1 TRIEDA WORKINGWATCHER

Vzhľadom ku potrebe dynamického pýtania sa používateľa na jeho aktuálny emocionálny stav bolo potrebné vytvoriť triedu, ktorá slúži na sledovanie či používateľ aktívne pracuje s počítačom. To znamená či aktuálne píše na klávesnici (neskôr sa plánuje aj sledovanie práce s myšou).

Trieda vyzerá nasledovne:



OBRÁZOK 26 REPREZENTÁCIA TRIEDY *WORKINGWATCHER*

Princíp fungovania je taký, že ak používateľ pustí poslednú stlačenú klávesu na klávesnici, spustí sa časovač a ak do časového intervalu znova nestlačí klávesu aktivuje sa udalosť *stopWorkingWatcher.*

10 ŠPRINT 6 – MARÍNA

10.1 IMPLEMENTÁCIA DAILYUSERMODELMANAGER

Pre možnosť vytvárať model používateľa pre jednotlivé dni a nie len súhrnne pre celé obdobie používania, bola navrhnutá trieda *DailyUserModelManager*. Trieda je svojím rozhraním kompatibilná s pôvodnou triedou *UserModelManager* a pre zachovanie kompatibility bolo z triedy *UserModelManager* extrahované rozhranie *IUserModelManager*, ktorý obe triedy implementujú.

DailyUserModelManager používa lenivé načítavanie dát z databázy, takže k databáze sa pristupuje až v prípade, že sú dáta potrebné. V databáze bola vytvorená tabuľka *DailyUserRecord*, ktorá obsahuje dátum, z ktorého dňa boli emócie pridávané a jednotlivé vektory emócií. Pre každého používateľa je teda v databáze toľko tabuliek *DailyUserRecord,* koľko dní bol sledovaný.

Pre možnosť súčasnej práce s oboma reprezentáciami bola zmenená a rozšírená trieda *UserModelBuilder*, ktorá súčasne vytvára, resp. spravuje oba typy modelu.

10.2 REDUKCIA POČTU EMÓCIÍ

Po analýze dát boli zo všetkých emócií vybraté tieto emócie:

- radosť,
- normálny,
- stres,
- únava a
- frustrácia.

Pre možnosť pracovať s vybratými emóciami bola upravená trieda *EmotionStateWraper*, konkrétne metóda *GetEmotionValues*(), ktorá vracia už iba vybrané emócie. Na zamedzenie používania nepodporovaných emócií a pre možnosť ich filtrácie bola predchádzajúca trieda doplnená o metódu *IsSupportedEmotion*(), ktorá testuje podporu zadanej emócie ako parametra.

10.3 VYUŽITIE NAIVE BAYES KLASIFIKÁTORA PRI KLASIFIKÁCIÍ EMÓCIÍ

Cieľom klasifikátora je prideliť pravdepodobnosti, s ktorými element patrí do jednotlivých tried. Využíva pri tom Bayesovo pravidlo:

$$P(H|d) = \frac{P(H)\,P(d|H)}{P(d)}$$

$P(H)$ – apriórna pravdepodobnosť H (prior)

$P(d|H)$ – vierohodnosť (likelihood) – podmienená pravdepodobnosť hypotézy H za pozorovania d

$P(d)$ – pravdepodobnosť výskytu pozorovania d (evidence)

$P(H|d)$ – posteriórna pravdepodobnosť H ak pozorujeme d (posterior)

Jednotlivé pravdepodobnosti sú určené ako početnosti výskytov v trénovacej množine. Tým je formovaná hypotéza.

Klasifikátor *Naive Bayes* predpokladá, že všetky atribúty sú:

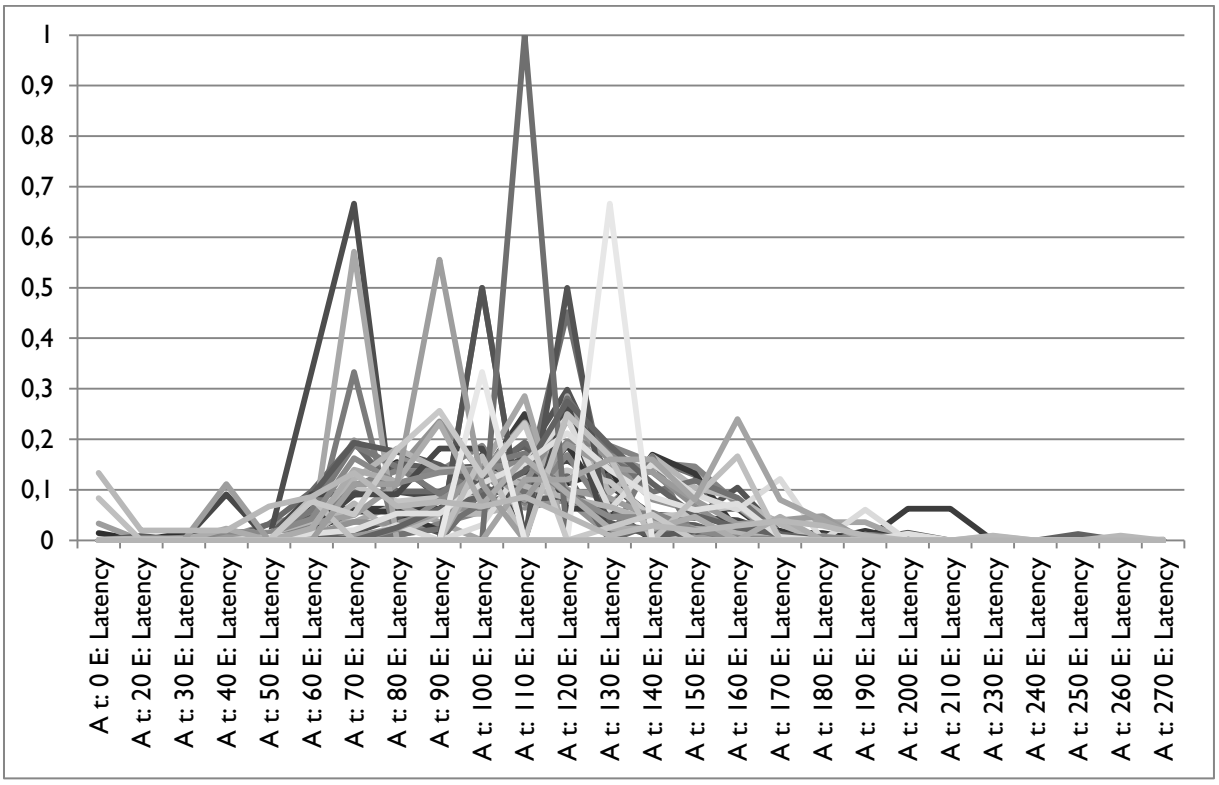
- Navzájom nezávislé
- Rovnako dôležité

Je preto dôležité analyzovať dostupné vzorky dát z hľadiska ich nezávislosti a dôležitosti.

10.4 ANALÝZA DÁT PRE POUŽITIE NAIVE BAYES KLASIFIKÁTORA

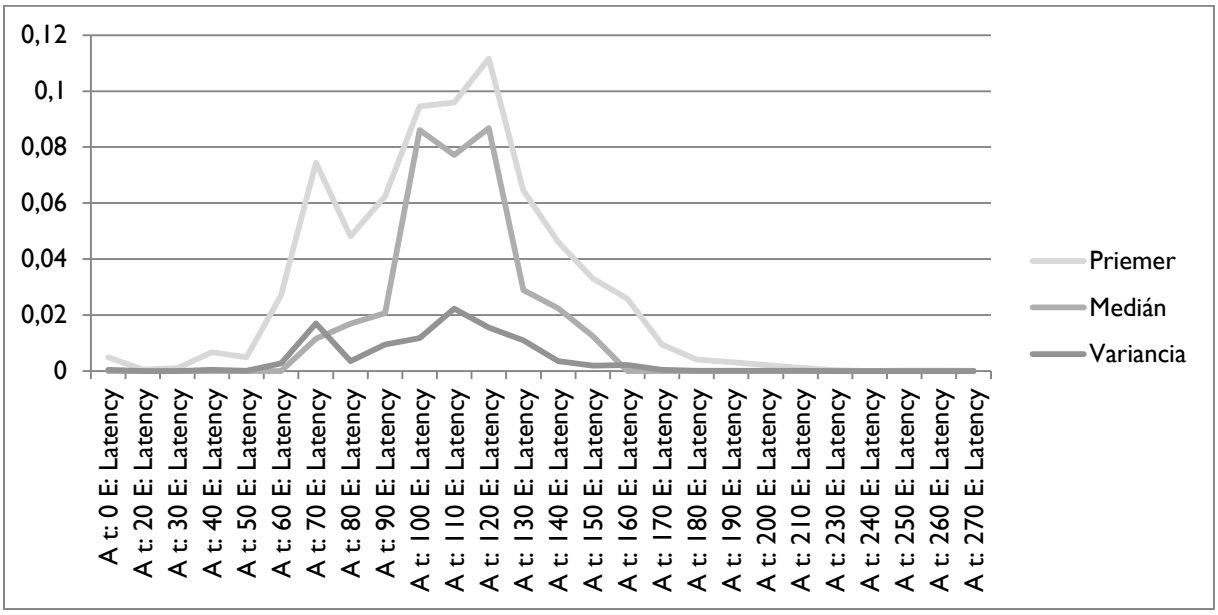
Dáta sú tvorené veľkým množstvom atribútov – pre každú klávesu, pre dvojice a trojice kláves sú počítané *Latency* a *FlightTime*, pre myš rýchlosť, akcelerácia, rýchlosť skrolovania kolieskom a latencia ľavého a pravého tlačidla a dvojklikov. Pre použitie klasifikátora je potrebné dáta redukovať. Analyzované dáta už boli zoskupené po 10ms intervaloch a atribúty, ktoré mali nulové hodnoty pri všetkých vzorkách, boli odstránené. Hodnoty boli normalizované tak, aby ich súčet pre jeden kláves a typ atribútu bol 1.

Na nasledovnom grafe možno vidieť 54 vzoriek duševného stavu *Normal* atribútov *Latency* pre kláves A.



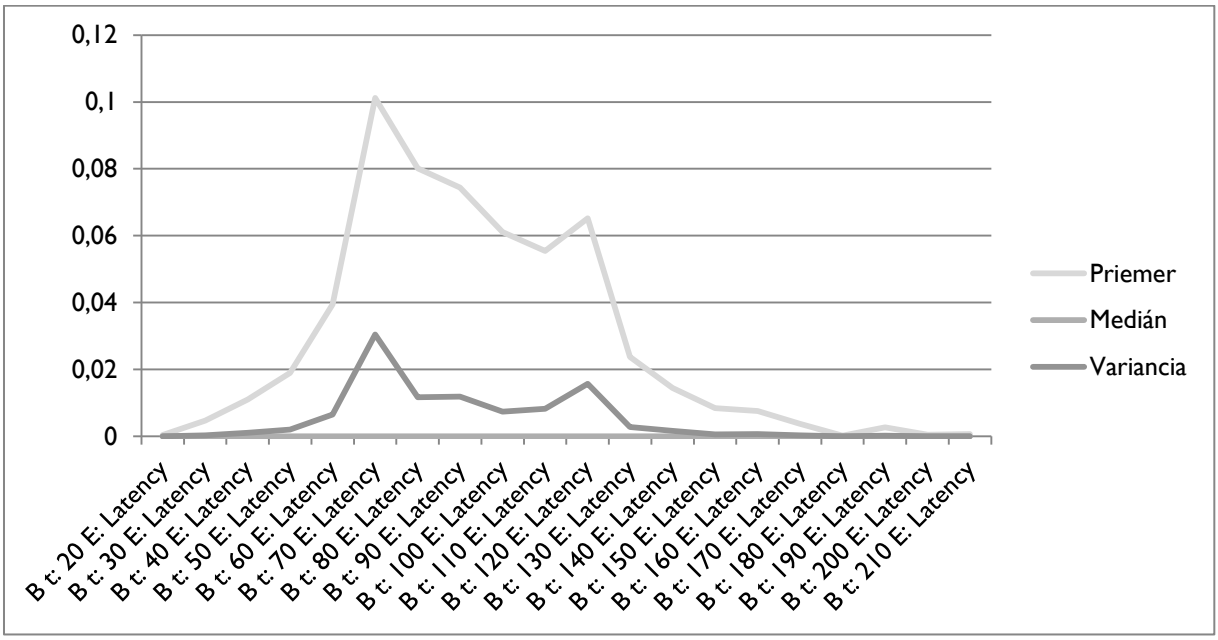
OBRÁZOK 27 – VZORKY LATENCY KLÁVES A PRE STAV NORMAL

Z grafu vidno určité rozdelenie vzoriek, ktoré lepšie zachytíme pomocou priemeru a variancie, prípadne mediánu, čo je znázornené na nasledovnom grafe.



OBRÁZOK 28 – PRIEMER, MEDIÁN A VARIANCIA LATENCY KLÁVES A PRE STAV NORMAL

Variancia je vyššia v oblasti, kde sú i namerané hodnoty vyššie. Jednotlivé vzorky sa priveľmi nelíšia. Aplikujeme obdobnú charakteristiku i na zvyšné atribúty. Nasledujúci graf zobrazuje priemer, medián a varianciu pre kláves B, čo je zobrazené na nasledovnom grafe.



OBRÁZOK 29 – PRIEMER, MEDIÁN A VARIANCIA LATENCY KLÁVES B PRE STAV NORMAL

Tu sa medián ukazuje ako nevhodná charakteristika – väčšina atribútov mala nulové hodnoty. Zato priemer a variancia ukazuje na podobné rozloženie hodnôt, ako pri klávesu A. Tieto

atribúty zrejme nie sú vzájomne nezávislé. Nízka variancia zároveň svedčí o redundantnosti vzoriek.

10.5 METÓDA ROZPOZNÁVANIA VYUŽÍVAJÚCA SVM

Trieda *SvmComparator* je implementácia porovnávača dočasných vektorov a modelu pomocou SVM a súčasne slúži aj na trénovanie SVM. Pri trénovaní treba správne nastaviť *kernel trick* a jeho parametre. Tieto nastavenia budeme musieť zistiť experimentálne.

10.6 NOVÉ GRAFICKÉ POUŽÍVATEĽSKÉ ROZHRANIE PÝTAČA

Bolo implementované nové grafické používateľské rozhranie okna zisťujúceho aktuálny emocionálny stav používateľa. Cieľom tohto nového používateľského rozhrania bola maximálna jednoduchosť a príťažlivosť pre používateľa. Keďže rôzni používatelia majú odlišný vkus, čo sa týka farebnej schémy rozhrania, v budúcnosti bude implementovaná možnosť nastavenie takej farebnej schémy používateľského rozhrania, ktoré používateľovi najviac vyhovuje. Ukážku nového používateľského rozhrania pýtača reprezentuje Obrázok 30.

OBRÁZOK 30 – NOVÉ POUŽÍVATEĽSKÉ ROZHRANIE PÝTAČA

10.7 ZMENŠENIE POČTU SLEDOVANÝCH EMÓCIÍ

	User 1	User 2	User 3	User 4	User 5	User 6	AVG
Normal	60,12	53,48	68,42	61,25	60,04	57,23	60,09
Tired	15,65	25,18	10,22	15,61	16,21	23,24	17,69
Disgust	12,86	0,51	1,3	9,95	12,49	6,03	7,19
Happiness	4,39	1,36	6,01	4,11	4,21	5,41	4,25
Stressed	1,51	2,63	7,08	5,35	4,96	3,84	4,23
Any	2,82	9,71	0	0,47	0,86	0	2,31
Nervous	0,31	2,93	1,14	1,22	0,31	4,25	1,69
Sadness	2,34	1,3	4,35	1,29	0,07	0	1,56
Anger	0	2,4	1,47	0,45	0,64	0	0,83
Surprise	0	0,5	0,01	0,3	0,21	0	0,17
Fear	0	0	0	0	0	0	0

TABUĽKA 1 - PERCENTUÁLNY PODIEL ZAZNAMENANÝCH AKTIVÍT POUŽÍVATEĽOVÝCH EMÓCIÍ

Na predchádzajúcej tabuľke je zobrazený podiel aktivít zaznamenaných s konkrétnou emóciou v období 3 mesiacov. Na základe tohto merania sme sa rozhodli zmenšiť počet sledovaných emócií v ďalšom priebehu vývoja aplikácie na nasledovné:

- *Normal*,
- *Tired*,
- *Disgust*,
- *Happiness*,
- *Stressed*,
- *Any* (táto emócia indikuje neprítomnosť používateľa pri PC).

Tieto emócie by zachytili 95,76% pôvodného rozsahu výberu emócií pri sledovaní používateľa, čo považujeme za postačujúce. Tieto emócie sú takisto odrazom toho, že pomocou klávesnice a myši dokážeme zachytávať skôr dlhodobé emocionálne stavy(normálny, unavený vystresovaný) ako krátkodobé (prekvapenie, hnev), nakoľko sa správanie pri písaní na klávesnici a pohybe myšou ťažko rozpozná na krátkom časovom intervale. Toto zúženie nám takisto pomôže pri vytváraní modelu používateľa a jeho rozpoznávania z hľadiska výpočtového výkonu a pamäťových nárokov.

Do ďalšej etapy projektu teda pokračujeme už len z uvedenými emóciami, kde sme sa po konzultácií s psychologičkou rozhodli premenovať emocionálny stav znechutený na frustrovaný.

10.8 ROZŠÍRENIE DYNAMICKÉHO PÝTANIA SA POUŽÍVATEĽA NA EMOCIONÁLNY STAV

10.8.1 TRIEDA ASKER

Pre zjednodušenie a sprehľadnenie funkcionality pýtania sa používateľa na emocionálny stav bola zavedená abstraktná trieda *Asker*, z ktorej dedia triedy *StaticAsker* a *DynamicAsker* reprezentujúce jednotlivú formu pýtania. Táto trieda obsahuje spoločnú funkcionalitu oboch prístupov pýtania.

10.8.2 TRIEDA STATICASKER

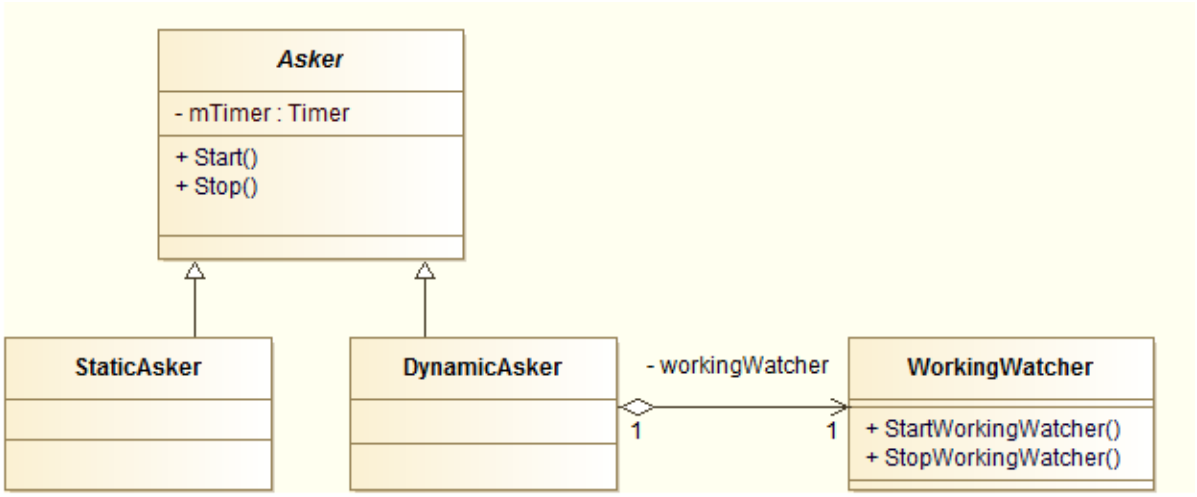
Trieda *StaticAsker* reprezentuje základný prístup pýtania sa používateľa na emocionálny stav. Po uplynutí *n*-minútového časového intervalu sa používateľovi zobrazí menu pre výber emócie. Po kliknutí na príslušnú emóciu sa časovač reštartuje. Časový interval je možné nastaviť v nastaveniach (*Asking Emotional State Interval*).

10.8.3 TRIEDA DYNAMICASKER

Dynamické pýtanie pomocou triedy *DynamicAsker* je oproti statickému komplikovanejšie. Po uplynutí základného *n*-minútového časového intervalu (nastavenia -> *Asking Emotional State Interval*) sa dostáva do jedného z dvoch stavov:

1. Používateľ nevykonáva žiadnu prácu pri počítači – nepíše na klávesnici (v súčasnosti sa berie do úvahy len klávesnica, do budúcna uvažujeme aj použitie myši). V tomto stave nevieme určiť, či používateľ je pri počítači alebo nie a čaká sa na vykonanie nejakej aktivity. Ak sa deteguje aktivita, prechádza sa do stavu 2.
2. Používateľ pracuje pri počítači – píše na klávesnici. V tomto stave sa počká, kým používateľ dokončí svoju činnosť. Hneď ako prestane písať na klávesnici na istú dobu (momentálne staticky 2 sekundy), predpokladáme, že dopísal myšlienku čo chcel a zobrazí sa mu okno pre výber emócie.

Sledovanie aktivity používateľa má na starosti trieda *WorkingWatcher*. Tá informuje o aktivite používateľa udalosťou *StartWorkingWatcher* a o ukončení aktivity udalosťou *StopWorkingWatcher*.



OBRÁZOK 31 - DYNAMICKÉ PÝTANIE SA EMÓCIE

11 ŠPRINT 7 – NIKITA

11.1 ODPORÚČANIA

11.1.1 KATEGÓRIE ODPORÚČANÍ

Ako odporúčania pre používateľa sme navrhli nasledovné kategórie:

- vtipy,
- povzbudenia,
- hudba,
- jedlo,
- nápoje,
- zlepšenie prostredia,
- prestávka
    - cvičenia,
    - prechádzka,
    - zmena aktivity pri počítači,
- obrázok z webu 9gag.com,
- príspevok s fóra lamer.cz.

Kategórie vtipy, povzbudenia, jedlo, nápoje, zlepšenie prostredia, prestávky sa budú používateľovi zobrazovať ako statický text, ktorý je uložený v databáze. Obrázky a príspevky z webu budú dynamicky vyberané ako náhodné z aktuálne najnovších prípadne najlepších príspevkov. Hudba sa bude vyberať ako náhodná pieseň z používateľovho priečinka s hudbou nastavenou v konfiguračnom súbore aplikácie(ako predvolený bude priečinok *MyDocuments\Music*).

11.1.2 VÝBER KATEGÓRIE

Výber kategórie spomedzi ktorej sa vyberie príspevok na zobrazenie odporúčania pre používateľa je robený podľa nasledovného vzorca – pravdepodobnosť výberu konkrétnej kategórie.

$$probability = \log(rank + 1) * weight$$

Váha(*weight*) je pevne nastavená pre každú kategóriu a pre hore uvedené kategórie sú váhy nasledovné:

- 1: jedlo, prestávky, povzbudenia, zlepšenie prostredia,
- 2: nápoje,
- 6: obrázky a príspevky z webu,
- 8: vtipy, hudba.

Hodnota poradia(*rank*) je pre každú kategóriu inicializovaná na hodnotu 50, ale táto hodnota sa dynamicky mení tým, že pri odporúčaní má používateľ možnosť zvoliť páči/nepáči sa dané odporúčanie a tým meniť hodnotu poradia – páči sa = +1, nepáči sa = -1.

11.2 Implementácia Ukončenia pýtania

Lokálna aplikácia PerConIK musí byť informovaná od serverovej časti, či je potrebné získavať ďalšie informácie o používateľovi. Tieto informácie sú prenášané návratovou hodnotou vzdialeného volania CommitActivity.

Server na detekciu dostatočného počtu dát používa dáta z tabuľky DailyUserRecord. Ak počet záznamov presiahne určitý počet pre aktuálneho používateľa klient je oboznámený s tým že dát je dostatok. Pre zamedzenie zastarávania dát a ich aktualizácii sa okrem počtu záznamov kontroluje aj vek najstaršieho záznamu. Ak bol najmladší DailyUserRecord vytvorený pred viac dňami ako je stanovené, server nariadi klientovi získavanie nových dát.

11.3 Implementácia n-minútového sledovania používateľa

Sledovanie používateľa prebieha neustále, jeho emocionálny stav sa v čase mení avšak mení sa častejšie ako je väčšinou nastavený interval pýtania sa. Preto bola do programu začlenená zmena, umožňujúca do modelu používateľa pridávať iba tie záznamy, ktoré vznikli v blízkom okolí zadania emócie.

Typ použitého vytvárania modelu je možné nastaviť zmenou typu objektu ktorý sa vytvára v metóde *createModelUpdater* v triede *UserModelBuilder*. Pre zachovanie kompatibility starého a nového prístupu, bol použitý vzor *Strategy*.

11.4 Aplikácia Fiit.Team10.CalculateResults

Táto aplikácia slúži na naplnenie databázovej tabuľky ComparatorResult. Napĺňa ju nasledujúcim spôsobom:

- Comparator – identifikátor použitého komparátora,
- Emotion – identifikátor emócie modelu, pre ktorú bola vypočítaná podobnosť,
- Result – výsledok porovnania,
- EvProperity – poradie v rámci výsledkov porovnania pre jednotlivý EmotionVector, komparátor a model,
- ModelProperity – identifikátor použitého modelu používateľa,

Výsledky sa počítajú pre každý emočný vektor, komparátor a model používateľa. Pre každé porovnanie pribudne toľko záznamov, koľko je modelov pre jednotlivé emócie v modeli používateľa.

Pri tomto porovnaní boli použite 4 modeli používateľa:

- UserModel
- DailyUserModel
    - bralo sa maximum s denného modelu
    - bral sa priemer s denného modelu
    - bral sa medián s denného modelu

11.5 Aplikácia Fiit.Team10.ResultPrezenter

Je aplikácia, ktorá v tabuľke zobrazí štatistiky jednotlivých modelov a komparátorov. Na porovnávanie je možné vybrať tri metriky:

- Pomer presných zásahov – pomer počtu prípadov, keď komparátor najlepšie ohodnotil emóciu, ktorá skutočne patrila emočnému vektoru ku počtu emočných vektorov
- Pomer zásahov na prvom alebo druhom mieste - pomer počtu prípadov, keď komparátor ohodnotil emóciu, ktorá skutočne patrila emočnému vektoru, na prom alebo druhom mieste ku počtu emočných vektorov
- Metrika na ohodnotenie správnosti – ide o číslo získané ako: $avg(\frac{1}{2^{EvProperity}})$, kde EvProperity poradie v rámci výsledku porovnania pri zhodnej emócii, vychádza s myšlienky, že ak sa komparátor trafí presne dá najlepší výsledok, ak sa trafí na druhé miesto, tak výsledok je dvakrát horší, ak sa trafí na tretie miesto, výsledok je dvakrát horší ako keď sa trafí na druhé miesto, atď.

11.6 Prehrávač hudby

Jeden typ odporúčania, ktoré môže byť ponúknuté používateľovi, je prehratie hudby z jeho lokálneho disku. Konkrétna pieseň sa vyberá náhodne zo zoznamu piesní, ktoré si používateľ predtým vytvoril. Po zvolení kladnej odpovede zo strany používateľa sa pieseň prehrá v predvolenom hudobnom prehrávači. Ak používateľ nemá tento program nainštalovaný na svojom počítači, zobrazí sa hlásenie, že hudby bude možné spustiť až vtedy, keď si používateľ doinštaluje daný program.

12 Šprint 8 – Olívia

12.1 Návrat dát pre odporúčanie klientovi

Po dopyte od klienta, keď si pýta odporúčanie od servera, server mu odpovie vo forme objektu triedy *RecommendationDto*, ktorá obsahuje nasledovné položky:

- rozpoznaná emócia,
- stav (*ok, not detected*),
- text,
- kategória,

- *rank id*.

U klienta sa odporúčanie zobrazí iba pri stave *ok,* stav *not detected* určuje, že emócia nebola rozpoznaná (napr. podobnosť k modelu nebola dostatočná podľa určenej prahovej hodnoty). V položke text sa nachádza samotný text odporúčania, v kategórii jedna z odporúčacích kategórii uvedených v predošlej kapitole. *Rank id* je číslo reprezentujúce položku databázy, ktorú je potrebné upraviť po tom, čo si klient zvolí, či sa mu odporúčanie páči alebo nie.

12.2 PRIDANIE STRATÉGIE VYHODNOTENIA EMÓCIE

Pri vyhodnocovaní neznámej emócie sa používa rozhranie *ICompareStrategy*, ktoré určuje stratégiu porovnávania. Predpisuje dve metódy *Compare* (porovnanie modelu s dočasným vektorom) a *Aggregate* (agregačná funkcia pre *DailyUserModelManager*). Dôvod pre jej zavedenie bol všeobecnejší prístup k vyhodnocovaniu neznámej emócie.

Rozhranie *ICompareStrategy* implementuje niekoľko tried:

- *DefaultCompareStrategy* – základná implementácia, využíva zadaný komparátor a agregačnú funkciu.
- *AvergageStrategy* – agregačná funkcia je priemer.
- *MedianStrategy* – agregačná funkcia je medián.
- *BestWccVdcStrategy* – kombinuje výhody *WeightCosineComparator* a *VectorDistanceComparator*, ako agregačnú funkciu používa medián.
- *RatioStrategy* – kombinuje dve stratégie v zadanom pomere.

Všetky stratégie aj *factory* trieda sa nachádzajú v mennom priestore *Fiit.Team10.EmotionEngine.Comparators.Strategy.*

12.3 AKCEPTAČNÉ TESTY

12.3.1 LOGOVANIE

Akceptačný test č.1

Názov: Funkčnosť logovacieho okna

Popis: Po spustení aplikácie sa spustí logovacie okno, do ktorého sa zadá emócia, ktorá sa následne uloží do lokálnej databázy.

Test vykonal: Peter Greguš

Test:

	Odkliknutá emócia	Emócia zapísaná do lokálnej db
Test 1	Šťastný	Šťastný
Test 2	Frustrovaný	Frustrovaný
Test 3	Unavený	Unavený
Test 4	Normálny	Normálny

Záver testu: Odkliknutej emócií zodpovedá daná emócia aj v logu lokálnej databázy. Test prešiel.

Akceptačný test č.2

Názov: Odchod dát na server

Popis: Pri logovaní sa napĺňa lokálna databáza. Z tejto databázy sa na server začnú odosielať dáta.

Test vykonal: Martin Geier

Test:

	Počet záznamov v lokálnej db	Počet záznamov v lokálnej db po uplynutí 1 min
Test 1	10740	10293
Test 2	10293	10147
Test 3	10147	9991
Test 4	9991	9836

Záver testu: Dáta sa z lokálnej databázy úspešne odosielali na server**.** Test prešiel.

12.3.2 MODELOVANIE

Akceptačný test č.1

Názov: Príchod dát na server

Popis: Pri logovaní sa na server odosielajú dáta. Do tabuľky activities tieto dáta prichádzajú a ukladajú sa.

Test vykonal: Peter Šinský

Test:

	Počet záznamov v tabuľke aktivít	Počet záznamov v tabuľke aktivít po 5 min
Test 1	3858	3863
Test 2	3863	3867
Test 3	3867	3874
Test 4	3874	3876

Záver testu: Test ukázal, že dáta prichádzajú na server ako majú. Každých 5 minút pribudlo priemerne 5 aktivít. Test prešiel.

Akceptačný test č.2

Názov: Vytvorenie UserModelu

Popis: Po logovaní prídu dáta na server a pri prvom príchode dát sa vytvorí nový UserModel pre daného používateľa.

Test vykonal: Martin Geier

Test:

	Vytvorený UserModel (T/F) pred príchodom dát	Vytvorený UserModel (T/F) po príchode dát
Test 1	F	T
Test 2	F	T
Test 3	F	T
Test 4	F	T

Záver testu: Po prijatí 20 aktivít jednej emócie sa úspešne vytvoril UserModel s dátami len pre danú emóciu. Test prešiel.

Akceptačný test č.3

Názov: Vytvorenie DailyUserModelu

Popis: Po logovaní prídu dáta na server a pri prvom príchode dát konkrétnej emócie sa vytvorí nový DailyUserModel pre danú emóciu.

Test vykonal: Martin Geier

Test:

	Vytvorený DailyUserModel (T/F)	Odkliknutá emócia v loggeri	Vytvorený DailyUserModel (T/F)
Test 1	F	Tired	T
Test 2	F	Normal	T
Test 3	F	Stressed	T
Test 4	F	Tired	T

Záver testu: Po prijatí 20 aktivít jednej emócie sa úspešne vytvoril UserModel s dátami len pre danú emóciu. Test prešiel.

Akceptačný test č.4

Názov: Porovnanie DailyUserModelu a UserModelu

Popis: Porovnajte či sa zhoduje počet dát UserModelu a súčtu jednotlivých DailyUserModelov pre jednotlivé emócie.

Test vykonal: Martin Geier

Test:

	Emócia	Zhoda v počte dát
Test 1	Normálny	Áno
Test 2	Šťastný	Áno
Test 3	Unavený	Áno
Test 5	Frustrovaný	Áno

Záver testu: Počty dát v UserModeli a súčty dát z DailyUserModelov sa zhodujú. Test prešiel.

12.3.3 ROZPOZNÁVANIE

Akceptačný test č.1

Názov: Kontrola pridania záznamu o výpočte používateľovej emócie

Popis: Každých 10 minút pri sledovaní používateľa v móde keď sa nepýtame používateľa na emóciu pribudne záznam v tabuľke CalculatedSimilarities o vypočítaných hodnotách komparátorov.

Test vykonal: Jozef Gajdoš

Test:

	Doba logovania	Počet pribudnutých záznamov v db
Test 1	12 minút	1
Test 2	25 minút	2
Test 3	30 minút	3
Test 4	1 hodinu	6

Záver testu: Výsledky testu potvrdili predpoklad, že každých 10 minút pribudne v tabuľke CalculatedSimilarities jeden záznam. Test prešiel.

12.3.4 ODPORÚČANIE

Akceptačný test č.1

Názov: Korektné zobrazenie odporúčania

Popis: Správne zobrazenie odporúčania v HTML okne.

Test vykonal:

Test:

	Kategória odporúčania	Korektné zobrazenie(T/F)
Test 1	Hudba	T
Test 2	Lamer	T
Test 3	9GAG	T
Test 4	Vtip	T

Záver testu: Odporúčania sa zobrazujú korektne. Test prešiel.

Akceptačný test č.2

Názov: Test zmeny hodnoty rank

Popis: Pri odporúčaní používateľ zaklikne páči sa/ nepáči sa tlačidlo, na základe, ktorého sa zmení hodnota ranku danej kategórie zmení o +1 alebo -1 na základe zakliknutého tlačidla (páči sa = +1, nepáči sa = -1).

Test vykonal: Peter Greguš

Test:

	Hodnota ranku pred odporučením	Hodnota ranku po odporučení	Páči sa/Nepáči sa
Test 1	50	51	Páči sa
Test 2	50	51	Páči sa
Test 3	50	49	Nepáči sa
Test 4	50	49	Nepáči sa

Záver testu: Hodnoty ranku sa po odkliknutí tlačidiel páči sa a nepáči sa upravujú správne. Test prešiel.

13 SUMARIZÁCIA PO LETNOM SEMESTRI

V tejto časti je uvedená sumarizácia práce na analýze, návrhu, implementácii a testovanie zadania uvedeného v úvode dokumentu.

13.1 ANALÝZA

Záverečná fáza aplikácie odporúča používateľovi na základe zistenej emócie. Preto bolo potrebné zistiť, aké typy odporúčaní je možné na ktoré emócie odporúčať. Odporúčania boli preto konzultované s odbornou psychologičkou. Bolo nám odporučené sústrediť sa na Maslovou model potrieb. Z modelu vychádza nutnosť sústrediť sa najskôr na základné potreby človeka, ako je pitie, jedlo a až keď sú vyriešené tieto potreby, začať uspokojovať vyššie potreby. Medzi vyššie potreby sme zahrnuli jednoduché odreagovanie, napr. vtipom, fyzickú aktivitu alebo okolité prostredie.

Z konzultácie so psychologičkou a analýzy nazbieraných dát sme tiež usúdili nutnosť redukcie počtu emócií. Medzi najvyužívanejšie emocionálne stavy sme zaradili:

- šťastný,
- normálny,
- unavený,
- vystresovaný,
- znechutený.

Pri konzultácii sme tiež odhalili vhodnosť zmeny názvu emócie zo znechutený na frustrovaný.

Pri skúšobnom behu aplikácie s väčším množstvom dát sme pozorovali priveľkú pamäťovú náročnosť dát v databáze. Zaťaženie spôsobovali vektory dát zachytávané v 10 minútových intervaloch. Pri bližšom skúmaní sme odhalili, že väčšina dát vo vektory je nezmenená a ich výpovedná hodnota je nulová, no stále zaberajú množstvo priestoru.

Po analýze sme zistili, že zmenou reprezentácie vektora dát zo poľa statickej veľkosti na tabuľku obsahujúcu kľúč a hodnotu, môžeme dosiahnuť 20 až 600 násobné zmenšenie vektora modelu emócie. Zmenšenie je samozrejme závislé na množstve dát vo vektory.

Pri zmene reprezentácie vektorov ale môžeme dosiahnuť oveľa väčšej úspory dát pri vektoroch vytváraných v 10 minútových intervaloch, kde zmenšenie pri takmer prázdnych vektoroch môže byť až nekonečné.

13.2 NÁVRH

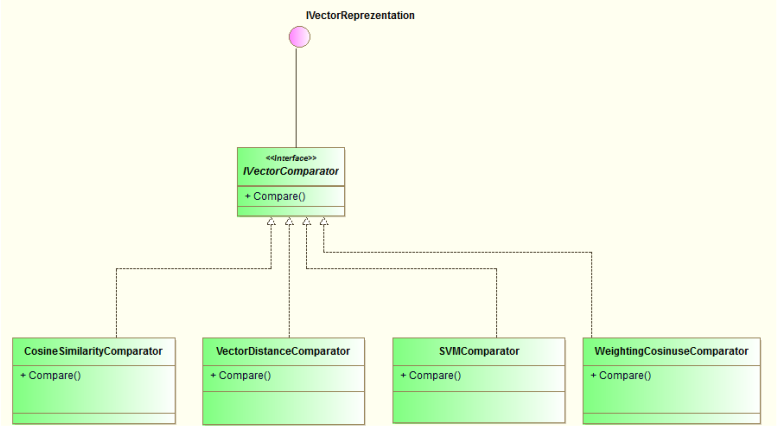
Pre reprezentáciu optimalizovaného vektora sme navrhli rozhranie IVectorRepresentation, ktoré majú implementovať konkrétne reprezentácie:

- WriteVectorRepresentation, optimalizovaný na pridávanie dát do vektora,
- ReadVectorRepresentation, optimalizovanú na rýchlosť čítania a pamäťovej náročnosti.

Pre zlepšovanie kvality vypočítanej emócie komparátorom sme navrhli nový model emócií používateľa. Model emócie nie je vytvorený ako súčet všetkých získaných dát, ale je

rozdelený na dni. Následne sme navrhli metódy porovnávania, kedy testovaný vektor testujeme zo všetkými dňami a na výsledok používame redukciu štatistickou metódou.

Po zmene implementácie a rozhrania vektora bolo potrebné upraviť existujúce metódy porovnávania. Súčasne sme navrhli dve nové metódy porovnávania vektorov. Prvá metóda je založená na algoritmoch podporných vektorov, druhá metóda využíva porovnávanie pomocou kosínusovej podobnosti.



Pre skvalitnenie práce používateľa s aplikáciou sme navrhli nový spôsob pýtania sa používateľa na emóciu. Metóda spočíva v periodickom vyskakovaní okna, avšak zohľadňujeme aktuálnu činnosť pri počítači. Ak používateľ píše, okno sa zobrazí až potom ako písať prestane. Ak používateľ je dlhodobejšie neaktívny, okno sa zobrazí až keď je zaznamenaná aktivita.

13.3 IMPLEMENTÁCIA

Na začiatku letného semestra sme pristúpili k optimalizácii vektorov emócií v pamäti aj databáze. Táto optimalizácia prebehla na dvoch úrovniach a to zmenou usporiadania dát (fly time, latency,...) vo vektore. A tým, že sa do databázy ukladá ako index - hodnota, pričom sú z neho vypustené nulové hodnoty. Tým došlo k výraznému zníženiu pamäťových nárokov. Pre vektory bolo vytvorené rozhranie *IVectorRepresentation* a jeho dve implementácie *ReadVectorRepresentation* (slúžiaca na rýchle načítanie dát) a *WriteWectorRepresentation* (slúžiaca na manipuláciu s vektorom).

Ďalej bol vytvorený ďalší typ modelu používateľa, ktorý využíva viacej modelov, každý pre jeden konkrétny deň, takýto typ modelu používateľa je využiteľný pri klasifikačných metódach.

Prístup k modelu používateľa bol zapuzdrený do tried UserModelManager (pre pôvodný model používateľa vyjadrený piatimi emočnými vektormi) a DailyUserModelManager (obsahuje model pre každý deň). Takisto bolo prerobené vytváranie a aktualizácia modelov.

K modelom používateľa bola pridaná stratégia porovnávania *ICompareStrategy*, ktorá určuje akým spôsobom sa bude model používateľa porovnávať s vektorom, ktorý má neznámu emóciu.

Boli pridané dva nové komparátory:

- *SvmComparator* – využíva algoritmy podporných vektorov (SVM), pre porovnávanie emočných vektorov
- *CosinusSimilarityComparator* – porovnáva vektory pomocou kosínusovej podobnosti a výsledky váži pomocou logaritmu metriky naplnenia ich častí

Do klientskej časti bola pridaná možnosť dynamického zisťovania emočného stavu používateľa (pomocou tried *StaticAsker* a *DynamicAsker*). Bolo pridané aj nové používateľské grafické rozhranie pre pytač (okno, ktoré sa spýta na emocionálny stav používateľa). Klient si vie zo servera zistiť, či sa má používateľa ďalej pýtať na emóciu, alebo mu poskytnúť odporúčanie.

Počas semestra bol úplne prerobený systém odporúčaní. Boli pridané kategórie odporúčaní: vtipy, povzbudenia, hudba, jedlo, nápoje, zlepšenie prostredia, prestávka, obrázky z webu a príspevky s fóra lamer.cz.

Do klientskej aplikácie bolo pridané okno s odporúčaním, ktoré získava aj spätnú väzbu od používateľa, či je dané odporúčanie vhodné. K oknu bola pridaná aj ďalšia funkcionalita, týkajúca sa odporúčaní, napríklad prehrávanie hudby. Na serveri sa odporúčanie vyberá aj pomocou spomínanej spätnej väzby a zistenej emócie.

13.4 TESTOVANIE

Pri komponovaní nových vlastností sme sa sústredili na ich pokrytie testami. Testami sme sa

Result	Test Name	Project
Passed	TriGraphGenerator	Fiit.Team10.Tests
Passed	Modus	Fiit.Team10.Tests
Passed	NormalizedContentIterator	Fiit.Team10.Tests
Passed	LoadTest	Fiit.Team10.Tests
Passed	Cmp	Fiit.Team10.Tests
Passed	MouseMetricsT	Fiit.Team10.Tests
Passed	Medain	Fiit.Team10.Tests
Passed	NormalizeTestExt	Fiit.Team10.Tests
Passed	NormalizedIterator	Fiit.Team10.Tests
Passed	Serialize	Fiit.Team10.Tests
Passed	NormalizeTestJoin	Fiit.Team10.Tests
Passed	ContentIterator	Fiit.Team10.Tests
Passed	Average	Fiit.Team10.Tests
Passed	DiGraphGenerator	Fiit.Team10.Tests
Passed	NormalizeTestSimple	Fiit.Team10.Tests
Passed	JoinNormalizedContents	Fiit.Team10.Tests
Passed	ShortcutGenerator	Fiit.Team10.Tests
Passed	GraphGenerator	Fiit.Team10.Tests
Passed	Iterator	Fiit.Team10.Tests
Passed	Variance	Fiit.Team10.Tests
Passed	MouseClicksT	Fiit.Team10.Tests

snažili skontrolovať správnosť častí, ktoré sú zložitejšie na implementáciu a logiku. Preto sme vytvorili testy hlavne na správnosť implementácie redukovaných vektorov. Algoritmicky náročne bolo aj vytvorenie štatistických hodnôt z vektorov a preto sa aj tomuto výstupu venovalo viacero testov.

Súčasťou záverečnej práce na projekte boli aj akceptačné testy, ktoré sú opísané v predchádzajúcej kapitole.