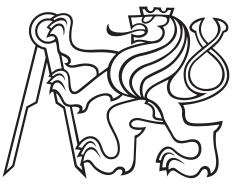
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Adaptace UI založena na emocích uživatele

Anastasiia Lunova
Studijní program: Softwarové technologie a management
Obor: Web a multimadia

Květen 2017
Vedoucí práce: Ing. Jiří Šebek

Poděkování / Prohlášení

Chtěla bych podekovat Ing. Jiřímu Šebkovi za vedení bakalářské práce, cenné rady a příjemnou spolupráci. Taky bych chtěla poděkovat celé své rodině a přítelovi za veškerou podporu, chápání a trpělivost.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 22. 05. 2017

.....

Abstrakt / Abstract

Bakalářská práce se zabývá úlohou vytvoření knihovny pro Android, která umožní přizpůsobení ozdobení rozhraní aplikace v závislosti na emocích uživatele. Návrh počítá s oběma stranami: uživatelem, náladu kterého má za cíl pozitivně ovlivnit, a vývojářem, pro kterého poskytuje jednoduchou integraci a následovně využití. Řešení umožňuje flexibilní použití knihovny a zpětnou vazbu s programátorem ve tvaru exportu dat z databázi, kde je uložená statistická informace o vnitřních dojmech uživatele. Při řešení úlohy byl použít programovací jazyk Java, prostředí Android Studio 2.2 od JetBrains' IntelliJ IDEA na operačním Windows 10 a SDK Affdex od Affectiva Emotion API pro snímání mimických výrazů.

Klíčová slova: Android, dynamická adaptace, rozhraní, emoce, ozdobení, design.

This bachelor thesis deals with the task of creating a library for Android that will allow customization of the application interface design depending on user's emotions. The design counts on both sides: the user, whom mood will be positively influenced by, and the developer, for whom it provides simple integration and usage. The solution enables flexible library use and feedback to the programmer in the form of exporting data from a database where the statistical information about users' internal impressions is stored. Task solving was to use the Java programming language, the JetBrains 'IntelliJ IDEA' s Android Studio 2.2 on Windows 10 and the Affectiva Emotion API Affdex SDI for scanning mimic expressions.

Keywords: Android, dynamic adaptation, interface, emotions, decor, design.

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

1.1 Motivace

Již v minulém století lidstvo vstoupilo do nové etapy rozvoje vědy, kdy se manuální práce začala nahrazovat inteligentními mechanismy, stroje jsou stále více soběstační, a to všechno proto, že tyto stroje se víc a víc začaly vybavovat elektronikou. Od té doby se všechna zařízení jen silněji spojuje s naším životem a teď mají za cíl nejenom pomáhat člověku v práci, ale také poskytovat pohodlí, odpočinek a usnadnit každodenní rutinu. Zároveň se staly součástí každodenního života i komunikační zařízení. Těžko si sebe můžeme představit bez takových pomocníků, jako počítač a mobilní telefon – dnes to jsou standardní atributy, jako oděv a obuv. Velké procento používání mobilů připadá na chytré telefony (v roce 2013 jejich tržby poprvé překročily prodej běžných telefonů, a toto číslo se nadále roste[1]). Takže, když takové zjednodušující život nástroje jsou vynalezeny, můžeme je zlepšovat, aby byly snadnější v použití, atraktivnější, ergonomičtější. Zároveň se za cíl předpokládá nejen zlepšení samotného zařízení, ale i aplikací, které běží na něm. Uživatel očekává nejen fungující aplikaci (samotné efektivity už nestačí) ale pěkně zdobený produkt, který je schopen "rozumět" svému majitelovi a odpovídat jeho pojetí komfortu a příjemnému vzhledu.

1.2 Cíle

Cílem práce je navrhnout možnou variantu vylepšení: vytváření aplikace, které upravuje svůj vzhled podle emoce uživatele. Realizace dané myšlenky poskytne následující výhody:

- Za prvé, zajistí takový zevnějšek rozhraní, který by s větší pravděpodobností vyhovoval uživatelovi.
  - Protože nebude statické šablony výzdoby a design se bude měnit plynulé a zřídka, pravděpodobnost znepokojení nebo nepřijetí barevné palety zákazníkem je menší, než kdyby byla fixovaná nebo s prudkými změnami.
- Za druhé, psychologické normování stavu člověka.
  - Adaptace bude navřena tak, aby pozitivně ovlivnit člověka – povzbudit smutného, uklidnit rozezleného (více v kapitole Zpracování emocí).
- Za třetí, to je jedna atraktivní věc navíc, která zvýší zájem o aplikaci a přispěje k její popularizaci.
- Za čtvrté, aplikace bude organizována takovým způsobem, že bude užitečná i pro vývojáře.

[1] Data jsou vzata ze statistiky, dostupné z `http://www.gartner.com/newsroom/id/3323017`

- V případě, že se negativní emoce detekuji trvalé, a to na stejném místě, - už se těžko dá považovat za náhodu. Nalezené výsledky by mohly poskytnout developerům signál, že pravděpodobně jsou nějaké problémy ve fungování. Výhodou je, že okamžitě je známá ta část aplikace, kde se tyto problémy vyskytují.

Pro vývoj byl vybrán operační systém Android, protože je nejpopulárnější mezi používaných chytrých telefonů[1].

[1] Data jsou vzata ze statistiky, dostupné z `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`

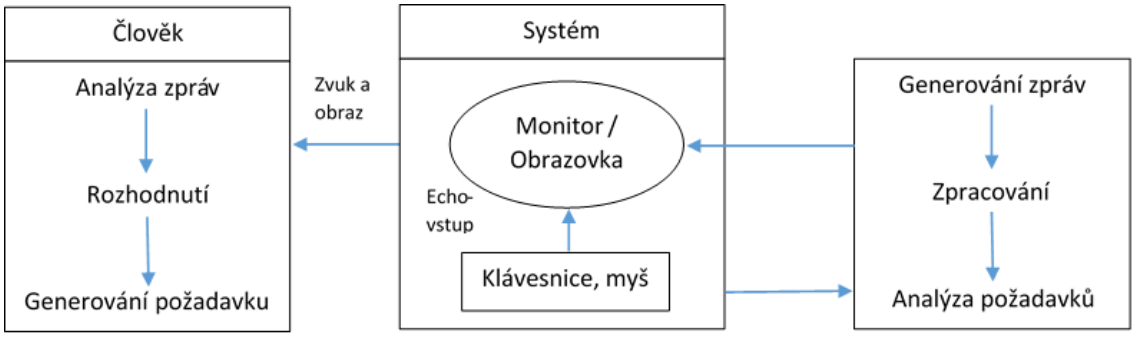
Kapitola 2
Rešerše

2.1 Uživatelské rozhraní

Uživatelské rozhraní (angl. User Interface, UI) je druh rozhraní, který obsahuje sadu nástrojů, technik a pravidel pro interakci jakéhokoliv systému a člověka, který jim řídí. Sadami nástrojů, technik a pravidel se rozumí [1]:

Nástroje:

- Výstupní data ze systému uživateli – veškerý spektrum účinků na lidský organismus (vizuální, sluchové, dotykové, čichové atd.).
- Vstupní data od uživatele do systému – různé druhy zařízení jako tlačítka, spínače, senzory polohy a pohybu, určité gesty, a dokonce i detekování mozkové aktivity člověka. Prostřednictvím nástrojů pro vstup a výstup dat se realizuje komunikace mezi dvěma strany „dialogu“ – člověkem s jedné strany a systémem s jiné. Proto UI se taky nazývá rozhraní „člověk-systém“.



Obrázek 2.1. Znázornění komunikace člověka a systému [2].

Mezi tyto prostředky by nemělo být nic zbytečného, ale zároveň nemá chybět nutné; měly by být pohodlné a praktické, umístěné přiměřeně a logicky (tím se zajišťuje ergonomie).

Techniky:

- sada pravidel stanovených developerem, podle které sada uživatelských akcí by měla vést k žádoucí reakci systému a plnění požadovaných úkolů.

Tato pravidla by měla být jasná natolik, aby byla snadně pochopitelná, přirozeně a jednoduše zapamatovatelná (to vše je zahrnuto v pojmu použitelnosti (usability)).

Jsou dva způsoby návrhu UI [3]:

- Fixní (statický) – rozhraní zachovává stálé schéma chování ve vztahu k uživateli.
- Dynamické (adaptivní) – rozhraní se přizpůsobuje uživatelskému modelu chovaní.

2.1.1 Adaptivní rozhraní a User Modeling

Adaptivní rozhraní umožňuje efektivnější použití možností systému pomocí automaticky laděného vzhledu podle preferencí nebo zvyků určitého uživatele. AUI je založené na statické části, která podporuje její fungování, inicializuje ji a ze které se dá získat potřebná data. Základem adaptivních systémů je uživatelský model (User Modeling).

2.1.2 User Modeling

Uživatelský model je souhrn informací o uživateli, díky kterému se systém přizpůsobí uživateli, předpovědí jeho cíle, preference nebo identifikuje vzor chování. Sběr informace o uživateli se koná buď explicitně prostřednictvím dotazníků, testy, nebo implicitně – monitorováním činnosti uživatele.

Při vytváření uživatelského modelu mohou být použity různé vzory, které se mohou i míchat [4]:

- Statický: je základním druhem uživatelských modelů. Jednou sehnaná základní data se už nemění. Posuny v preferencích uživatelů nejsou registrovány a žádné učení algoritmů není použité ke změně modelu.
- Dynamický: umožňuje více aktuální reprezentaci uživatele. Změny v jejich zájmu, jejich pokrok nebo interakce se systémem jsou zjistitelné a ovlivňují uživatelský model. Tento model tak může být aktualizován a aktuální potřeby a cíle jednotlivých uživatelů vzatý na zřetel.
- Stereotypní: založené na demografických statistikách. Na základě shromážděných informací uživatele jsou zařazení do stereotypů. Systém se pak přizpůsobí tohoto stereotypu. Aplikace tak může učinit předpoklady o uživateli, i když nemá o něm žádné údaje (neboť vychází z hypotézy, že všechny uživatelé v tomto stereotypu mají stejné vlastnosti).
- Vysoce adaptivní uživatelské modely (individuální): snaží se reprezentovat jednoho konkrétního uživatele, a proto umožňují velmi vysokou přizpůsobivost systému. Na rozdíl od modelů založených na stereotypů nespoléhají na demografické statistiky, ale mají za cíl nalézt konkrétní řešení pro každého uživatele, ale za cenu zjištění velkého množství informací při prvním použití aplikace.

Současné použití stereotypního přístupu a individuálního je dobrý příklad kombinování uvedených druhů modelování. V takovém případě při zahájení první komunikace se nový uživatel inicializuje podle nejvhodnějšího stereotypu a dál podrobnější informace bude postupně zjištěná algoritmy vysoce adaptivních modelů.

2.1.3 Vývoj uživatelského modelu

Obvykle je uživatelský model postaven přiřazením vah klíčovým slovům, která zastupují zájmy, definice chování uživatele. Ke klíčovému slovu se přičítá váha pokaždé co bylo to slovo zmíněno ve kontextu používání systému a tím pádem čím zajímavější je téma pro uživatele, tím větší celkovou váhu má slovo (v daném případě, čím je častěji emoce rozeznávána, tím větší váhu má stav mezi ostatními).

Modely založené na klíčových slovech jsou jednoduché v implementaci, ale vyžadují velké množství informací o uživateli.

Poté, co model je postaven, začíná samotný proces přizpůsobování. V závislosti na systému algoritmy mohou být velmi odlišné od sebe navzájem. Společné mezi nimi je to, že výsledek jejich práce je vždy zaměřen na zlepšení účinnosti interakce uživatele a informačního systému. Posledním krokem procesu adaptace rozhraní je zavedení výsledků přizpůsobení do fungujícího systému. [5]

2.2 Adaptace

Na problematiku adaptace uživatelského rozhraní se dá dívat s různých stran:

- adaptace může být zapříčiněná nutností zajištění požadované úrovni bezpečnosti systému (například, v závislosti na kategorii, do které uživatel patří, může mu být zakázáno, dovoleno nebo povoleno s určitými omezeními vykonání některých funkcí nebo přístup k určitým částem systému);
- přizpůsobení na základě relativitě použití objektů aplikace (například, změna pořadí položek menu, složení tlačítek na panelu nástrojů);
- změny kosmetických komponent rozhraní (barev, tvarů, písmem).

Jelikož knihovna pracuje s emocemi, což je subjektivním projevem, systém se může vyjádřit ve stejném směru pomocí kosmetických úprav. Dá se využit následující druhy změn:

- palety barev,
- druhy, barvy a rozměr písma,
- tvar UI-objektu (zaokrouhlení rámečku políček, když je uživatel zlí),
- zvukový doprovod.

V rámci dotyčné práce budou využity metody proměny barevných palet a tvaru objektů. Jak a v jakých případech se použijí je vysvětleno v kapitole “Zpracování emocí a následující adaptace”.

2.3 Emoce

Podle Paula Ekmana[1] existuje 7 základních emocí:

1) hnus,
2) zlost,
3) strach,
4) smutek,
5) štěstí,
6) překvapení,
7) despekt.

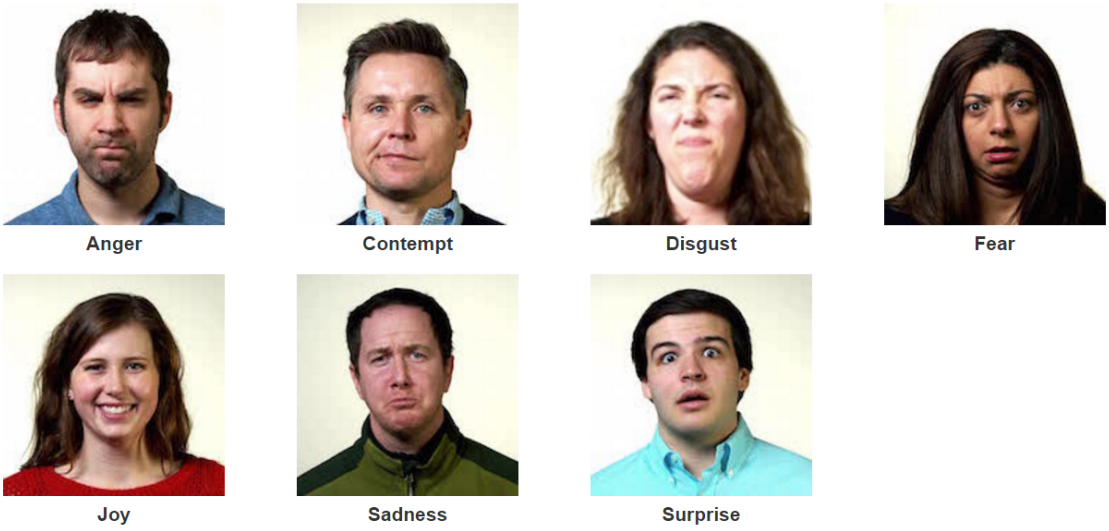
V rámci této práce nás nebudou zajímat strach, opovržení a překvapení. Takže zaměřím se na detekování a zpracování projevů:

1) štěstí,
2) smutku,
3) hnusu
4) a zlosti.

Se změnou stavů se bude příslušně měnit i barevná úprava aplikace. Základním je neutrální stav s vlastní paletou pro vzhled rozhraní.

2.3.1 Měření emocí. API

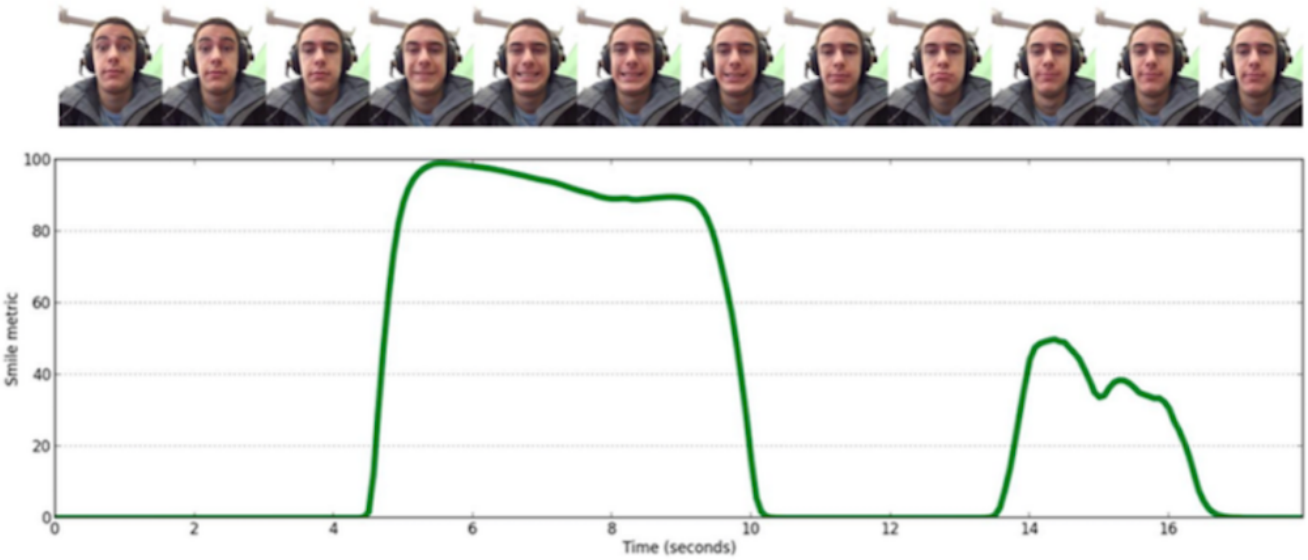
Pro získání emočního stavu uživatele budu používat Affectiva Emotion SDK [7], neboť se podporuji Androidem a je použitelný mobilními telefony s tímto operačním systémem, nepotřebuje mnoho místa, navíc nepotřebuje připojení k Internetu (analýza se bude konat přímo na přístroji). Affectiva[7] umí odhalit všechny 7 základních emocí [8], takže i ty, které jsou vybrané pro účely projektu.



Obrázek 2.2. Affectiva, Metriky [8].

[1] Paul Ekman je americký psycholog, 59. nejcitovanější psycholog 20. století. Proslavil se studiem emocí, a především jejich vztahu k mimice. Tvrdí, že v lidské mimice jsou mikroprojevy (trvající 1/25-1/15 sekundy), které není člověk schopen ovládnout, a které prozrazují emoci, jíž se člověk snaží skrýt. Definoval sedm základních emocí, které se projevují skrze mimické mikroprojevy. Jemu patří koncepce odhalování lže z mimiky [6].

Metriky, které využívá SDK, ukazují, kdy uživatel demonstruje určitou emoci nebo výraz tváře (například, úsměv) spolu se stupněm vyjádření: od 0 do 100 podle toho, jak silně je vyjádřen dotyčný projev.



Obrázek 2.3. Výstupní data z práce Affectiva [7].

2.3.2 Rozeznávání

Pomocí dat zachycených z kamery knihovna rozeznává určité výrazy obličeje, ze kterých se dá zjistit emoce. Odhalení emocí se uskutečňuje mapováním mimiky do emocí, která se staví na mapování EMFACS vyvinutém Friesen & Ekman[1].

Affectiva detekuje pět pohybů svalů tváře a jejich kombinace (AU = action unit), viz obrázek 2.4.

Pro vyhodnocení pravděpodobnosti vyjádřené emoce se ověřuje přítomnost každého pohybu. Pomocí počítačového vidění a strojového učení, lícní kódovací technologie Affectivy – Affdex [7] – automatizuje rozeznávání emoce. Video se z webové kamery přenáší v reálném čase do cloudu Affectivy, kde je automaticky analyzováno (nebo se zpracovává přímo na přístroji, pokud se nevyužívá internet připojení). Za-prvé, Affdex extrahuje důležité části obličeje (například, rty, obočí), potom analyzuje pohyb, tvar a kombinaci těchto vlastností, aby definovat určitý AU (obrázek 2.5).

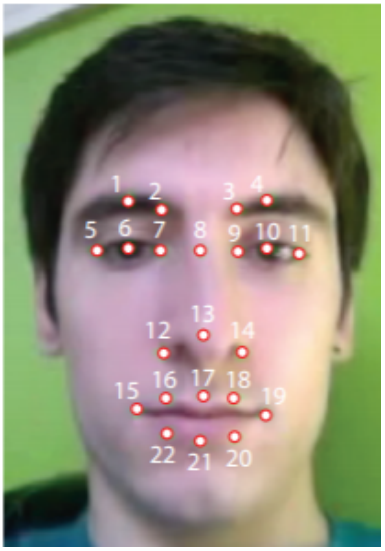
Každý AU má svou „oblast zájmu“ (ROI = region of interest), která se nachází kolem bodu v příslušné části tváře. Obraz (konkrétní frame videa) je pak oříznout dle ROI a pro každou oblast je následovně vypočítán histogram orientovaných gradientů (HOG = histogram of orientated gradients), přičemž nezávislé jedna na druhé.

Výstupem pak je spočítaná váha každé ze sedmi emocí, co rozeznává Affectiva.

[1] EMFACS (Emotional Facial Action Coding System) je systém taxonomie mikro pohybů svalů lidské tváře, založený na systému původně vyvinutému švédským anatomem Carl-Hermanem Hjortsjö a později adoptován pod koncepci Paula Ekmana a Wallace V. Friesena.

AU	Description	Example
AU02	Outer corner of eyebrow raised.	
AU04	Eyebrows drawn medially and down.	
AU09	Upper lip raised and inverted; superior part of the nasolabial furrow deepened; nostril dilated by the medial slip of the muscle.	
AU10	Upper lip puller up (either unilaterally or bilaterally).	
AU12/ smile	Lateral lip corner pull without AU04 or AU09.	
AU14	Dimpler (dimples in the cheeks).	

Obrázek 2.4. Definice a obrázky s příklady pohybů tváře, které detekuji Affektiva (převzaté z oficiální specifikace [9]).



Obrázek 2.5. Detekování důležitých části tváře a označení funkčními body [9].

2.3.3 Chybovost detekce emocí

Emoce se nedá detekovat absolutně správně. Je to práce s lidským faktorem, a proto odchylky jsou nezbytné. Tak třeba někteří lidé mají specifickou mimiku pro výraz nějakého pocitu, což podle „šablony“ ukazuje často na opačnou věc. Z toho důvodu říkáme o pravděpodobnosti emoce a vlivu výrazu obličeje na tuto pravděpodobnost, který může být buď pozitivní nebo negativní. Následující tabulka ukazuje vztah mezi výrazy obličeje a predikaci emocí [??]. Znázornění mimických výrazů jsou v Příloze B.

Emoce	Pozitivní vliv	Negativní vliv
Štěstí	■ Úsměv	■ Svraštěné obočí ■ Zvednuté obočí
Smutek	■ Zvednuté vnitřní konce obočí ■ Svraštěné obočí ■ Stlačené rohy rtů	■ Zvednuté obočí ■ Otevřená pusa ■ Stisknuté rty ■ Sevřené rty ■ Úsměv
Hnus	■ Pokrčený nos ■ Zvednutý horní ret	■ Stisknuté rty ■ Úsměv
Zlost	■ Svraštěné obočí ■ Otevřená pusa ■ Svraštěná brada ■ Sevřené rty ■ Široce otevřené oči ■ Stisknuté rty	■ Zvednuté obočí ■ Zvednuté vnitřní konce obočí ■ Pokrčený nos ■ Úsměv
Překvapení	■ Zvednuté vnitřní konce obočí ■ Zvednuté obočí ■ Široce otevřené oči ■ Pokleslá čelist	■Svraštěné obočí
Strach	■ Zvednuté vnitřní konce obočí ■Svraštěné obočí ■Široce otevřené oči ■Protáhnutý ret	■ Zvednuté obočí ■Stlačené rohy rtů ■Pokleslá čelist ■Úsměv
Despekt	■ Svraštěné obočí ■ Samolibý úsměšek	■ Samolibý úsměšek

Tabulka 2.1. Vztah mezi výrazy obličeje a predikaci emocí [10]

2.4 Zpracování emocí a následující adaptace

2.4.1 Provázanost barev a emocí člověka

Alexandr Etkind (1979, 1980-1985) provedl řadu studií barevno-emocionální propojenosti u dospělých. Ve svém pokuse od roku 1979 zkoumal spojení 8 barev z testu M. Luschera s 9 základními emocí podle K. Izardu (1980). Následující tabulka procentuálně ukazuje frekvenci barevných asociací s emocionálními faktory Izarda [11].

Barva	Zájem	Radost	Překvapení	Smutek	Zlost	Hnus	Ostuda	Strach	Únava
ŠEDÁ	6	4	2	27	1	15	18	12	53
MODRÁ	27	4	2	27	5	7	13	15	8
ZELENÁ	26	10	26	13	8	7	19	8	7
ČERVENÁ	16	52	23	4	55	4	4	17	2
ŽLUTÁ	20	24	56	1	9	19	12	15	1
FIÁLOVÁ	5	12	14	12	6	22	16	7	12
HNĚDÁ	10	8	3	14	4	27	17	3	23
ČERNÁ	10	2	2	22	38	18	13	43	24

Tabulka 2.2. Asociace barev s emocemi

V této studii si pokusné měli představovat emoce ve formě barvy, tj. volit si takové barvy, které by nejvíc, podle vlastního mínění, jejím odpovídaly. Dále všechny názory pro určitou barvou k dané emoci byly vypočítány a prezentovány v podobě procentuálních hodnot.

Z výzkumu je vidět, že lidé mají silnou vazbu `emoce-barva`, kteráž stanovena na hluboké úrovni, protože i, aniž by člověk byl v určitém stavu (jak to bylo během testu), je i nadále schopen barevně nadefinovat emoci. Samozřejmě, existují i nejednoznačný výklad (například, jak to je se zájmem), a to z důvodu, že jednak emoce je složitější, a za druhé, hraji roli individuální preference a vnímání. Zatím v celku jsou výsledky poměrně homogenní. Pro cíle práce je důležité, je-li schopná barva donutit člověka cítit určitou emoci. Proto neposkytuji fakta o pravděpodobnosti opačného vlivu.

„S ohledem na všudypřítomný výskyt barvy v životě a kultuře, by se dalo očekávat, že psychologie barev je velmi dobře rozvinutá oblast," - označuji výzkumníci Endryu Elliot a Marcus Mayer. „Je to překvapivě, ale do dnešního dne málo bylo provedeno teoretických nebo empirických výzkumů o vlivu barvy na psychologické fungování." [12].

Ale to, že je jich málo neznamená, že neexistují vůbec.

Tak, například, díky výzkumů Rose Kivi, Dana Tuffelmire [13], a skupiny Birren, Kinney, Schaie a Woodruff (1981) [14] se zjistilo, že i když děti zatím nemají naučeny tradiční významy, přisuzovaný emocím, podvědomě provazují studené barvy se smutkem, neboť kreslí smutné tváře, a teplé – s radostí, a kreslí postavy s úsměvem. Kendra Cherry, psycholog, která má nejvyšší počet citací svého blogu za požadavkem `colors impact moods` na Google Scholar, uvádí několik dalších neobvyklých příkladů důsledků vlivu barev na člověka [12]:

- Jeden výzkum [15] ukázal, že placebo, obarvené v barvy teplého spektra, byly zaznamenané jako účinnější než tablety v chladných tónech.
- Instalace modrého pouličního osvětlení může přivést ke snížení kriminality v těchto oblastech [16].
- Zcela nedávno vědci zjistili [17], že červená barva nutí lidí reagovat s větší rychlostí a silou, což by mohlo být užitečné během sportovních akcí.
- Jedna studie [18], založená na historických údajích, ukázala, že sportovní týmy oblečený v černé uniformy se častěji dostávají trest.
- Průzkum [19] rovněž ukázal, že některé barvy mohou mít vliv na výkon. Například, ovlivnění studentů červenou před zkouškou mělo negativní dopad na výsledky zkoušek. Každého ze studentů zastupovalo členské číslo buď červené, zelené nebo černé barvy. Pak se prováděl pětiminutový test. Výsledkem je, že studenti s červenými čísly ukázali výsledky o 20 % horší než studenti se zelenými nebo černými.

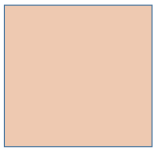
Takže, i když jednoznačného důkazu vlivu zbarvení na kondici lidí zatím není, tato oblast je vyvíjí, i přesto na základě provedených experimentů nelze alespoň popřít, že barva hraje určitou roli ve formování lidské nálady a pocitů.

2.4.2 Výběr palet adaptačního rozhraní

Na základě údajů uvedených v tabulce 2.2 a z předpokladu pozitivního vlivu na emocionální stav uživatele byly vytvořeny palety designu rozhraní pro každý případ uvedený ve druhém odstavci kapitoly ‘Emoce’. Hlavním tónem úpravy pro konkrétní emoci je vzatá taková barva, která má minimální hodnotu v příslušném sloupci (tedy současný negativní stav je vyvážený pomocí opačné barvy podle asociace). V případě nalezení pozitivní emoce – radosti, – výše uvedené pravidlo bude mírně korigováno: namísto ostře opačné hodnoty (černé) budou využity barvy odpovídající náladě. Pokaždé bude vybrána kombinace tří barev: barva pozadí, písma a zdůrazňující (bude hrát roli barvy orámování u objektů). Spolu s tím se bude měnit tvar objektů – zaokrouhlení (rámečků tlačítek, textových polí).

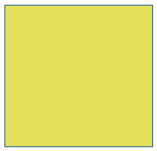
2.4.3 Neutrální stav

Neutrální stavu je základem, který se použije, pokud není zjištěna žádná emoce. Jak říká samotný název, paleta bude vytvořena neutrálními odstíny. Jako báze byl vybrán teplý hnědý, a jeho světlejší a tmavší varianta – jako doplňující.

Položka adaptace	Ilustrace	RGB			HEX
		R:	G:	B:	
Pozadí		238	201	177	EEC9B1
Písmo		71	87	111	47576F
Akcent		162	127	104	A27F68
Zaokrouhlení	–				

Obrázek 2.6. Paleta pro stav Neutrál.

2.4.4 Potěšení, štěstí

Položka adaptace	Ilustrace	RGB			HEX
		R:	G:	B:	
Pozadí		227	224	89	E3E059
Písmo		39	55	114	273772
Akcent		150	148	44	96942C
Zaokrouhlení	–				

Obrázek 2.7. Paleta pro stav Štěstí.

2.4.5 Smutek

Položka adaptace	Ilustrace	RGB			HEX
		R:	G:	B:	
Pozadí		242	124	56	F27C38
Písmo		248	248	248	F8F8F8
Akcent		242	29	29	F21D1D
Zaokrouhlení	–				

Obrázek 2.8. Paleta pro stav Smutek.

2.4.6 Zlost

Položka adaptace	Ilustrace	RGB			HEX
		R:	G:	B:	
Pozadí		114	189	232	72BDE8
Písmo		71	83	181	4753B5
Akcent		116	117	130	747582
Zaokrouhlení	+				

Obrázek 2.9. Paleta pro stav Zlost.

2.4.7 Hnus

Položka adaptace	Ilustrace	RGB			HEX
		R:	G:	B:	
Pozadí		255	199	201	FFC7C9
Písmo		122	40	52	7A2834
Akcent		216	143	109	D88F6D
Zaokrouhlení	+				

Obrázek 2.10. Paleta pro stav Hnus.

2.4.8 Alternativní vývoj

Jiná možnost, jak lze provádět adaptaci, týkající se barevného ozdobení, je místo použití natvrdo založených palet provádět korigování existující, nastavené programátorem, palety: například, když je detekován smutný stav – celý vzhled se zesvětlí, když stav je zlý – přidá se zelená složka, smutný – zvýrazní, barvy stanou jasnější atd. V případě uvedeného postupu neutrální stav bude originální vzhled rozhraní beze změn čili programátorská volba palet.

Kapitola 3
Související práce

Tato bakalářská práce ve své implementační části používá pro vybudování modelu aplikace koncept stromu, myšlenka kterého je popsána v bakalářské práci N.Mishchenko "Aspektové orientovaný vývoj adaptivní struktury aplikace pro Java EE aplikace" [20].

Kapitola 4
Analýza a design knihovny

Pro vývoj projektu, zaměřeného na přizpůsobení k náladě uživatele, mám řešit dané otázky:

- Vytvoření meta-modelu aplikace
- Zjištění předmětů adaptace (co se bude měnit)
- Definice detekovaných stavů a popis chování knihovny (jak se bude měnit)
- Ukládání zjištěných dat pro potřeby vývojáře.

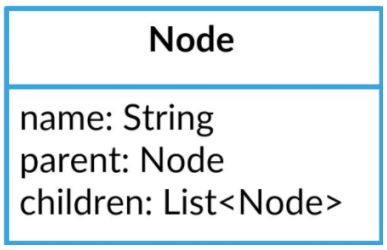
4.1 Krok 1. Vytvoření metamodelu aplikace

Aplikace obvykle řeší několik druhů aktivit, a proto se může skládat z několika okének a mít trochu odlišné vzhledy v závislosti na účelu. Aby se adaptace aplikovala ne jenom na jednu stránku aplikace, ale i na ostatní, je zapotřebí před startem knihovny zjistit celkovou představu o struktuře základného projektu.

Hlavní okno aplikace je kontejnerem pro prvořadné opce (seznam v panelu menu), kteréž případně mohou obsahovat v sobě další (položky, co jsou vidět po rozklikávání jedné z možnosti ve zmíněném seznamu) atd. Datová struktura strom vystihuje formátu zjištěných informací.

Tj. každý bod menu lze reprezentovat jako uzel – čili je zaváděna třída Node. Třída Node, jako normální část stromu, má uchovávané znalosti o rodiči, dětech a vlastní identifikaci – v tomto případě jméno.

Sumarizace kroku: zavádění třídy Node a instančních proměnných, odpovídajících za stromovou strukturu.



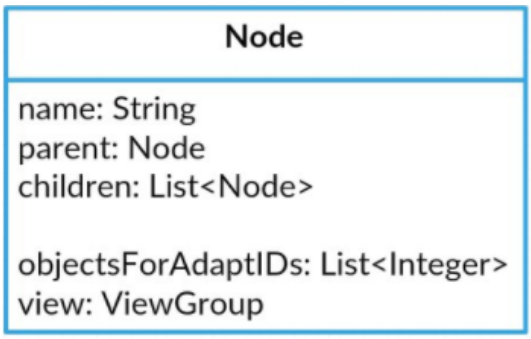
Obrázek 4.1. Třída Node pro sestavení stromu aplikace

4.2 Krok 2. Zjištění předmětů adaptace

Každý node, jinými slovy každá stránka aplikace, má svou náplň a liší se objekty rozhraní.

Aby umět pracovat s nimi je nutné je uložit. Stačí vědět id objektu. Na tento účel pohodlným bude seznam, z toho důvodu, že umožňují rychlý přístup k datům.

Sumarizace kroku: zavádění další instanční proměnné, týkající se náplně uzlu, a důležitá poznámka pro budoucí implementaci je obdržení proměnné typu View pro možnost zjištění výš uvedených dat.

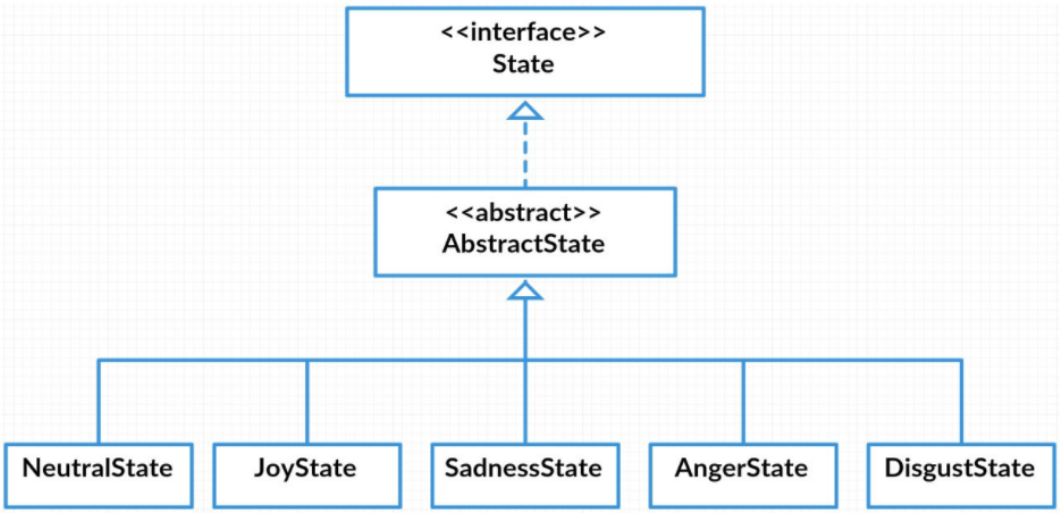


Obrázek 4.2. Třída Node s přidanými proměnnými pro ukládání objektů

4.3 Krok 3. Definice detekovaných stavů a popis chování knihovny

Na každou z měřených emocí by bylo zapotřebí reagovat různými způsoby. Pro popis chování budou nutné třídy, odpovídající za stavy. Jak bylo zmíněno ve druhém odstavci kapitoly "Emoce", bude se pro příklad uvažovat pět základních emocí.

Sumarizace kroku: naplánování tříd stavu.



Obrázek 4.3. Plánovaný diagram tříd pro definici stavů

4.4 Krok 4. Ukládání dat

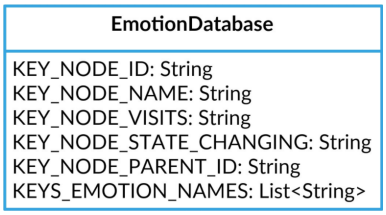
Data se budou ukládat do databázi SQLite, výhodou kteréž je pro mobilní aplikace to, že zapisuje primo do souboru. Pro připojení a veškerou práci s databází má být dodána samostatná třída.

Předpokládá se, že údaje, uložené v tabulce, budou využité pro analýzu aplikace vývojářem: odhalení slabých míst a zachycení možností úprav. Na to by se hodilo vědět kolikrát uživatel celkem navštěvoval konkrétní stránku, jak často se během visitu mu měnila nálada a jaké emoce většinou cítil (přehledněji by to bylo ve tvaru poměru).

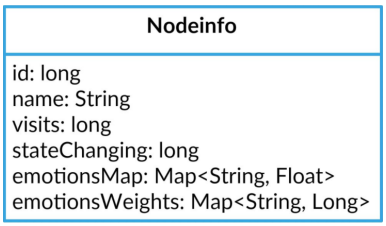
Může se stát, že detekování negativních reakcí v uzlech je způsobeno rodičem, takže by bylo dobře mít v databázi i záznam o rodiči, který je uložen ve třídě Node.

Práce s databází má svou specifičnost: při dodání, vytěžování nebo kopírování dat s, resp. do databáze, vytváří se nová třída, do které se nasýpá to, co se zjistí z databáze, resp. to, co by se chtělo tam uložit. Proto není vhodné míchat informace o "fyzické" části uzlu s tím, co by se chtělo uložit do tabulky, a uchovávat všechno dohromady v jedné třídě. Z tohoto důvodu pro držení dat "emocionální" části uzlu, by měla být zavedena jednotlivá třída – NodeInfo a zakomponovaná do třídy Node.

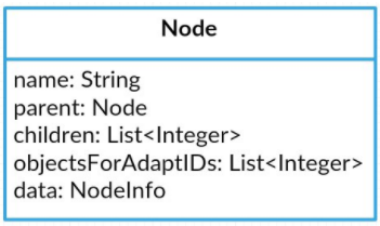
Sumarizace kroku: vytváření třídy pro práci s databázi EmotionDatabase, zavádění třídy NodeInfo pro její účely a vožení poslední do třídy Node jako instanční proměnné.



Obrázek 4.4. Třída EmotionDatabase pro práce s databází



Obrázek 4.5. Třída NodeInfo pro ukládání informace pro databázi

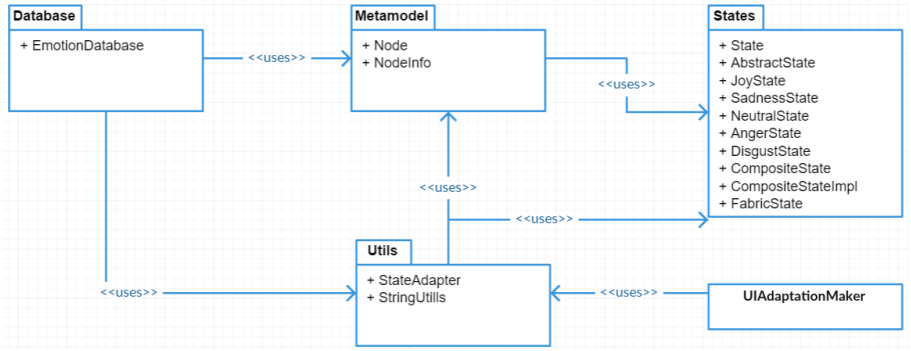


Obrázek 4.6. Třída Node s vestavěnou proměnnou typu NodeInfo

Kapitola 5
Implementace

5.1 Struktura projektu

Probírání kroků, popsaných v analýze, ukázalo, jaké části podle účelu má připravována knihovna. Aby najednou porozumět celému obsahu libovolného projektu, uvidět jeho modularitu a vysledovat vazby mezi jeho částmi, hodí se Package Diagram. Jak je vidět z obrázku 4.1, implementace projektu se skládá ze tří logicky rozdílených modulu a jednoho pomocného – Utils.



Obrázek 5.1. UML Package Diagram

Balík "Database" obsahuje třídu, určenou pro vytváření databázi a veškerou práci s ní.

Balík "Metamodel" chrání informaci o struktuře aplikace ve tvaru stromu. Každá instance třídy Node je skladištěm dat o jednotlivých položkách menu (čili aktivitách) aplikace.

Balík "States" obsahuje rozhraní a třídy, které řeší, co se má dít v případě určité emoce.

V balíku "Utils" jsou popsané třídy, vykonávající pomáhající funkcí a které nemají příčin být instanciovanými.

UIAdaptationMaker je rozhraní, přes kterou aplikace bude komunikovat s knihovnou.

5.2 Třídy pro popis metamodelu

S ohledem na to, že aplikace může být i více okénková, jak bylo zmíněno v Kapitole 4.1, je zapotřebí ze začátku zjistit celkovou představu o struktuře základného projektu. Přitom výhodou je, co nejednoduše pro samotného programátora, zjistit nezbytné pro implementaci poznatky. Z tohoto důvodu struktura bude získaná pomocí vlastního menu, neboť každá z jeho položek je zodpovědná za určitou aktivitu, resp. náhled. Tímto způsobem, jediné, co bude nutné od vývojáře, který bude chtít použit knihovnu, je to sdílet s ní proměnnou menu.

Jakmile se vyvolá metoda createApplicationStructure() a objekt třídy Menu je pro knihovnu známý, vytvoří se Node reprezentující tu hlavní stránku, která se objevuje při startu aplikace. Ta je vnímána jako root. Dál se využiji vestavěná možnost Javy pro Android číst příslušný xml soubor, kde je popsáno, jak to celé menu vypadá, a získat z toho jak samotné položky, tak i submenu v případě, že je má.

Podle Kapitoly 4.2, je nutné zjistit objektovou náplň každého Nodu. Aby moct ovlivnit objekty stránky, za pravidly programování na Androidu, je nutné vědět jejich id. Ty se dá získat z proměnné typu ViewGroup příslušné stránky a to požadavkem getChildAt(int id). Metoda vrátí objekt typu View. V Androidu od View se dědí tlačítky (Button, CheckBox, RadioButton, FloatingActionButton, ToggleButton, …), různé vzhledy (ImageView, CalendarVeiw, TextView), TextField, Toolbar, ProgressBar a mnoho dalších tříd, což je popsáno v oficiální dokumentaci[1].

Z toho důvodu metoda createApplicationStructure() žádá o další argument – objekt třídy ViewGroup, který se dá vytáhnout z platné aktivity pomocí:

```
getLayoutInflater().inflate(R.layout.activity_main, null);
```

Výpis 5.1. Způsob obdržení ViewGroup z aktivity

Vývojář může dle vlastního uvážení vybrat druhy objektů pro modifikaci. Tím pádem může sám rozhodnout, jaké prvky budou podléhat změnám. Na to má při zavolání metody createApplicationStructure() hlavního rozhraní UIAdaptationMaker přidat ji jako poslední argument dynamické pole typu String, které obsahuje názvy tříd objektů. Pro změnu určitého typu objektů stačí uvést třídu jenom jednoho. Pokud programátor nechá dynamické pole prázdným, jinými slovy poslední argument metody bude chybět, defaultně se budou měnit všechny druhy objektů, které má stránka.

```
// example 1: change only Buttons and TextViews
uiAdaptation.createApplicationStructure(this, menu,
getLayoutInflater().inflate(R.layout.activity_main, null),
button.getClass().toString(), textView.getClass().toString());

// example 2: change everything
uiAdaptation.createApplicationStructure(this, menu,
getLayoutInflater().inflate(R.layout.activity_main, null));
```

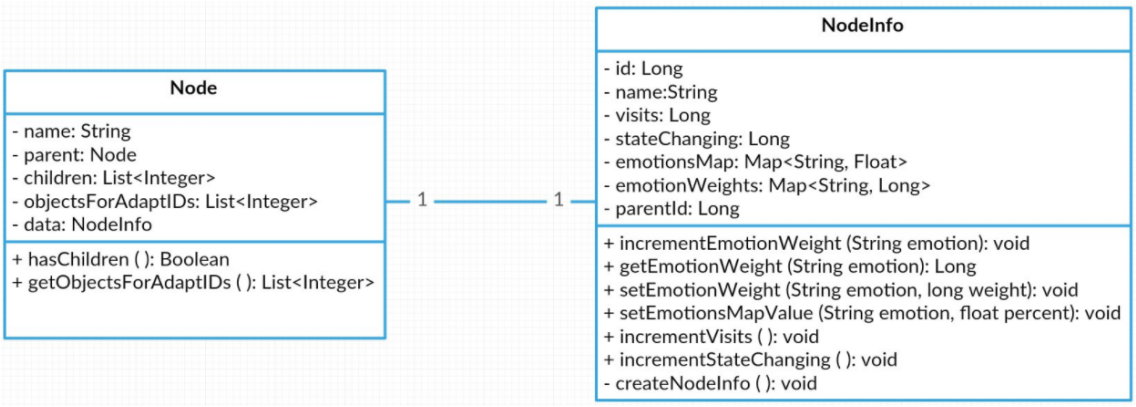
Výpis 5.2. Příklady volání metody createAplicationStructure()

V Kapitole 4.4 bylo vysvětleno, proč je zavedena další třída NodeInfo – ve skutečnosti je skoro reprezentací řádku databázi:

- long id – id uzlu aplikace (to, co mu přidělila databáze);
- String name – název uzlu;
- long visits – počet návštěv uzlu;
- long stateChanging – počítá, kolik krát došlo ke změně stavu;
- Map¡String, Long¿ emotionWeights – uchovává kolik krát se vyskytla každá emoce;
- Map¡String, Float¿ emotionsMap – uchovává výskyt každé emoce v procentech (je vypočteno ze stateChanging a emotionWeights).

[1] Dostupné z `http://www.gartner.com/newsroom/id/3323017`

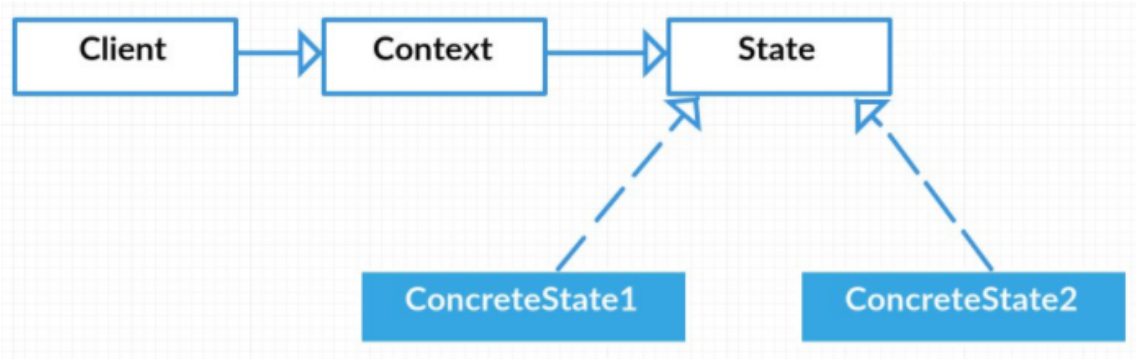
Ve finále, třídy pro vytvoření metamodelu z balíčku “Metamodel” vypadají takovým způsobem:



Obrázek 5.2. Třídy pro sestavení metamodelu

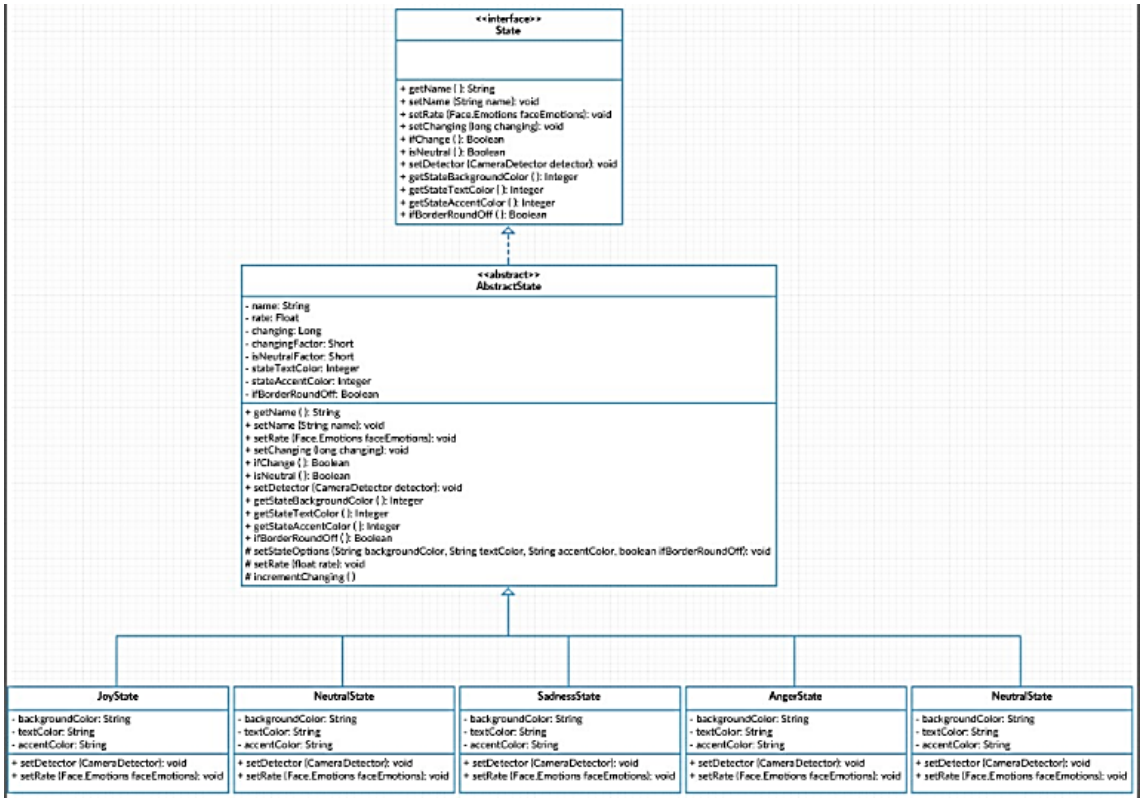
5.3 Třídy pro popis stavů

Implementace chování knihovny dle každé detekované emoce je vybudovaná návrhovým vzorem State. Podle definice, návrhový vzor State [21] je vhodným řešením v případě, kdy je objekt, jež během své existence mění své vnitřní chování tak, že nabývá různých stavů. Přičemž chování objektu v různých stavech se výrazně liší.



Obrázek 5.3. Obecný diagram návrhového vzoru State

Při změně vnitřního stavu objektu se pak objekt reprezentující původní stav zamění za objekt jiný, jež odpovídá stavu novému. A to je přesně to, co se děje při určení emoce.



Obrázek 5.4. Třídy pro popis stavů

Třída AbstractState má proměnné:

- String name – název smaotného stavu;
- float rate – je hodnota od 0 až 100, což znamená míru vyjádření emoce; stanoví SDK Affectiva;
- long changing – je pro fixování faktu změny emoce: Affectiva měří body obličeje několik krát za vteřinu. Proto kdyby se, například, uživatel jen na okamžik přestal usmívat, náhle se by mu vyměnilo rozhraní. Taková častá a rychlá transformace by neovlivňovala náladu uživatele v pozitivním směru, proto je nutné dát čas, aby vyhnout takovému problému a přizpůsobení pod náhodnou náladu.
- final short changingFactor – určuje právě ten `čas`, o kterém se jednalo výš. Adaptace bude provedena jen tehdy, kdy proměnná changing dosáhne hodnoty changingFactor.
- final int defaultBackgroundColor, defaultTextColor, defaultAccentColor – jsou defaultní hodnoty, nadefinované pro ten případ, kdy pří dodání nové emoce nebude uvedená příslušná barva.
- final int stateBackgroundColor, stateTextColor, stateAccentColor – uložiště pro hodnoty barev, které se budou ve skutečnosti použité za adaptaci.
- boolean ifBorderRounfOff – nastavená na true bude znamenat změnu formy UI-objektů do zaokrouhleného tvaru.

Pro detekování výrazu obličeje knihovna importuje třídy SDK Affektiva: CameraDetector, Detector, Face.Emotions.

Aby nástroj věděl, co má detekovat, musí se za nastartování aplikace nastavit zmíněný Detector tak, že se předá true-hodnota jako argument do setteru s názvem příslušné emoce , například:

```
detector.setDetectJoy(true); // to enable joy detection
detector.setDetectSadness(true); // the same for sadness
detector.setDetectAnger(true); // and anger
// … and so on…
```

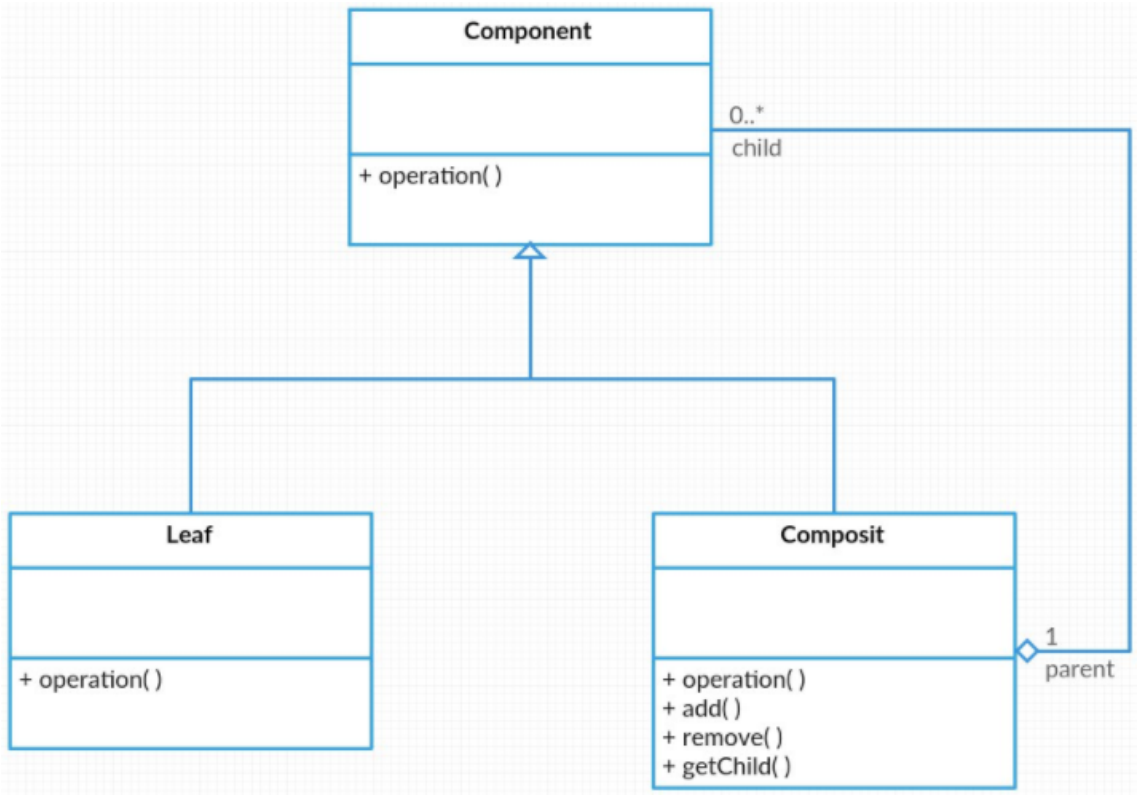
Výpis 5.3. Demonstrace postupného volání metod pro nastavení detektoru

Stejný postup platí i když se sbírají naměřena data:

```
faceEmotions.getJoy();
faceEmotions.getSadness();
faceEmotions.getAnger();
```

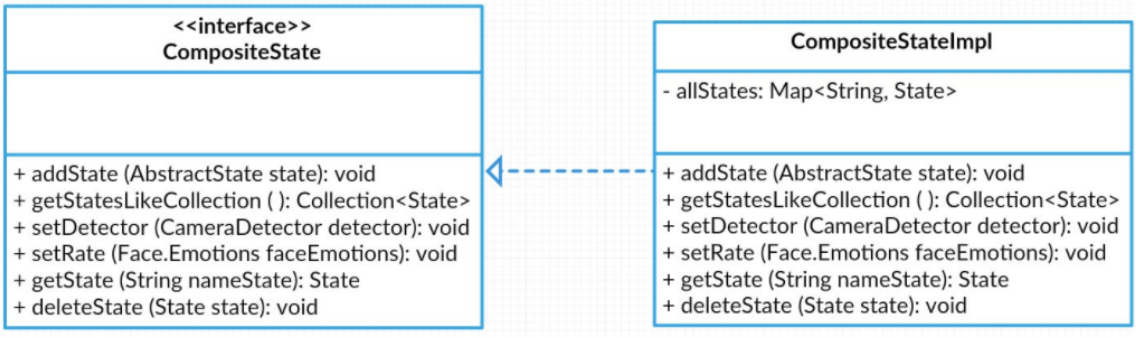
Výpis 5.4. Demonstrace postupného sbírání dat

Aby se nemuselo takovým způsobem `ručně` nastavovat detektor a vytahovat výsledky, je možné sebrat všechny stavy dohromady, jak to třeba dovolí vzor Composite [21].



Obrázek 5.5. Obecný diagram návrhového vzoru Composite

Proto byla vytvořena třída CompositeState s tím rozdílem od standardu vzoru, že se nedědí ani neimplementuje stejné rozhraní s výš uvedenou třídou State (což je pro Composite Leaf-em) z důvodu komplikací realizace. Nejedná se tedy o přesný pattern Composite, ale jeho myšlenka je základem.



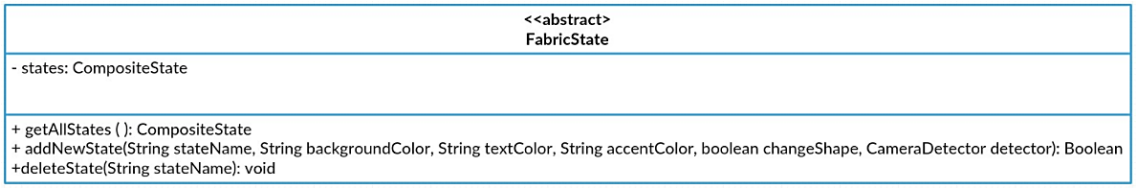
Obrázek 5.6. Interface a implimentující ho třída pro Composite

Aby byla knihovna flexibilní vůči potřebám těch, kdo ji bude používat, umožňuje i přidání dalších emocí stejně jako i odebrání ze seznamu zpracování. Vývojář může operovat emocemi v rámci, stanovené Affectivou, tj.:

- Joy (potěšení, štěstí)
- Sadness (smutek)
- Anger (zlost)
- Disgust (hnus)
- Surprise (překvapení)
- Fear (strach)
- Contempt (despekt)

Pro to potřebuje zavolat metodu addNewState() a jako argument předat název ty emoce, jakou chce doplnit program, nepovinně ty barvy, co by chtěl vidět (v případě null bude použitá defaultní hodnota, nadefinovaná ve třídě AbstractState), a hodnotu typu boolean, která znamená požadavek na zaokrouhlování tvaru UI-objektů. Veškeré nastavení se provedou pomocí Java Reflection. Pro smazání emoce stačí do metody deleteState() předat název.

Zmíněné opce poskytuje třída FabricState. Kromě toho je hlavním uchovatelem stavů, které se budou zpracovávat během činnosti aplikace. Každá třída knihovny, která pro svou funkčnost potřebuje vědět o nich, bere znalosti právě z FabricState. To dává výhodu, proto že po změně seznamu zpracovávajících emocích není potřeba měnit ten výčet v několika místech – aktuální informace je v jednom místě.



Obrázek 5.7. Třída FabricState

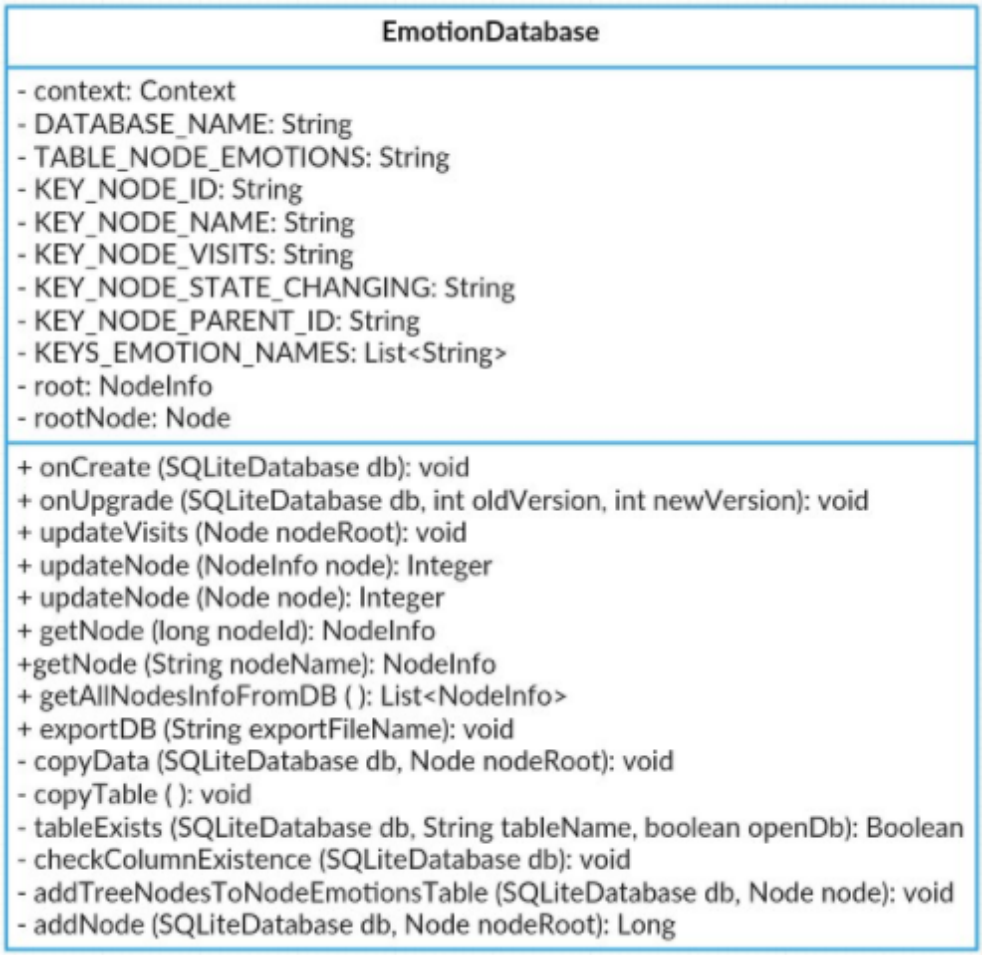
5.4 Třída pro ukládání dat

Na účel udržení informace je použita databáze SQLite, pro práci s kterou je zavedená třída EmotionDatabase, dědící třídu SQLiteOpenHepler. Kvůli tomu se mají být přepsány metody onCreate(SQLiteDatabase db) a onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion). Metoda onCreate(SQLiteDatabase db) se volá, když je databáze poprvé vytvořena. Zde se vytvoří tabulky a jejich počáteční nastavení.

```
@Override
public void onCreate(SQLiteDatabase db) {

if(!tableExists(db, TABLE_NODE_EMOTIONS, db.isOpen())){

//if table doesn't exist make creating command and execute

StringBuilder creatingCommand = new StringBuilder();

creatingCommand.append("CREATE TABLE IF NOT EXISTS")
.append(TABLE_NODE_EMOTIONS).append(" (");

int index = 0;

for(Map.Entry keyPair : allKeys.entrySet()){
index++;
creatingCommand.append(keyPair.getKey()).append(keyPair.getValue));
if(index == allKeys.size()) creatingCommand.append(")");
else creatingCommand.append(", ");
}

db.execSQL(creatingCommand.toString());

//Add nodes to table in conformity with them tree structure
this.addTreeNodesToNodeEmotionsTable(db, rootNode);
}
else {
//if table already exists { copy table header and then copy content
copyTable();
copyData(db, rootNode);
}

}
```

Výpis 5.5. Předepsána metoda onCreate() třídy EmotionDatabase

První, co je pro potřeby projektu bakalářské práce nutné – zkontrolovat, existuje-li už tabulka s daty. Pokud ano, a to by se nekontrolovalo, tak po každém spuštění aplikace by se mazaly nasbírané z minula informace. Jinak se bude pouze aktualizovávat, resp. doplňovat. Metoda onUpdate(SQLiteDatabase db, int oldVersion, int newVersion) se volá v tom případě, kdy databáze má být aktualizována. V této metodě se bude kontrolovat existence všech sloupečků, co má argument db – kdyby vývojář pak chtěl rozšířit tabulku o další informaci, tak požadovány sloupeček se vytvoří, aniž by smazal existující tabulku. Třída EmotionDatabase po tom vypadá následovně:



Obrázek 5.8. Třída pro práce s databází

5.5 Analýza dat detektoru

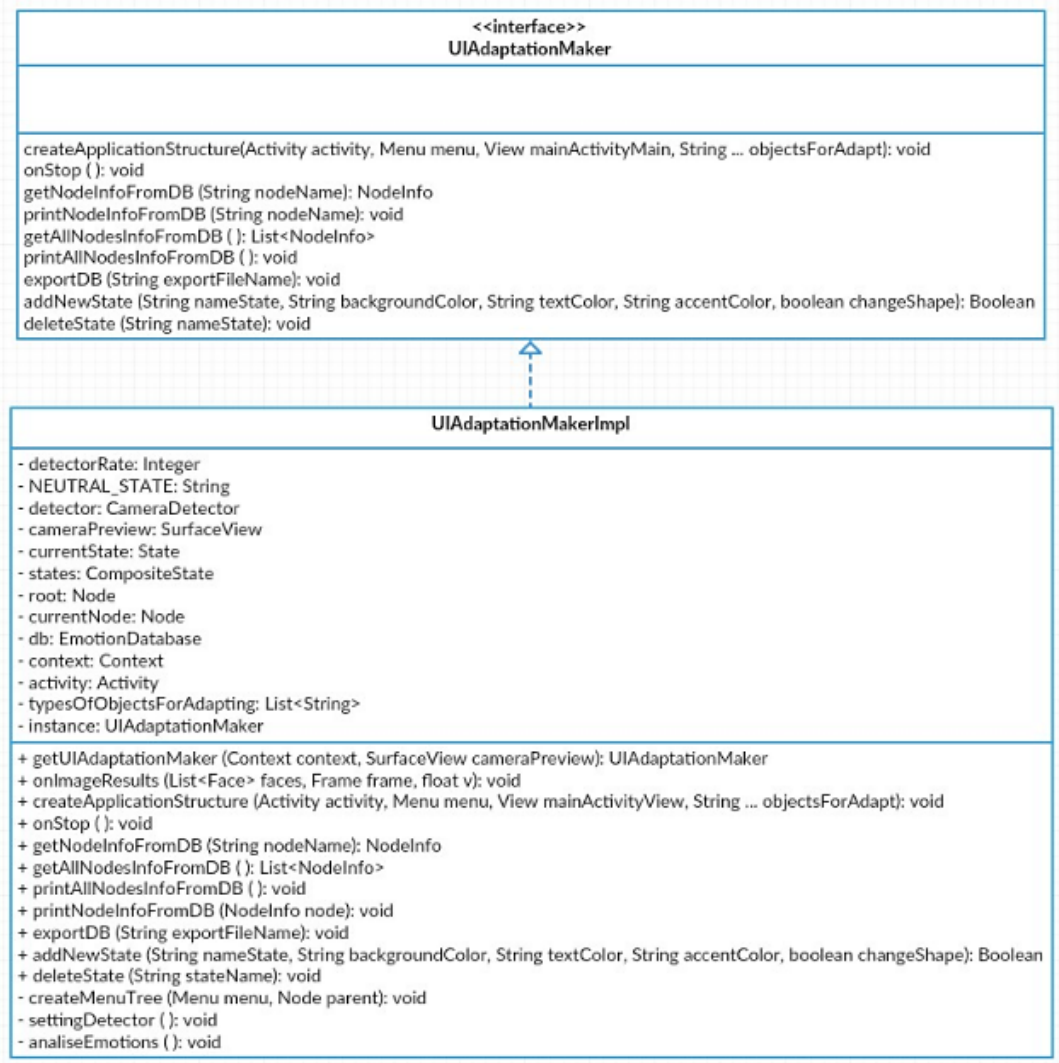
Snímání a sběr dat provádí SDK Affectiva. Pro možnost práce s ní má být třída, která implementuje rozhraní Detector.ImageListener a přepisuje metodu:

```
public void onImageResults(List<Face> faces, Frame frame, float v){ … }
```

Výpis 5.6. Deklarace metody onImageResults(), určená pro přepsání

Právě v ni se popisuje, co se má dít s obdrženými čísly. Odsaď se po detekci změny emoce vyvolává požadavek na přizpůsobení a zápis příslušného záznamu do databáze.

Tuto důležitou metodu by bylo nebezpečně nechávat přístupnou, ale i udělat ji private taky nelze, neboť je součásti rozhraní. Z této příčiny ukryjeme hlavní třídu UIAdaptationMakerImpl za rozhraním UIAdaptationMaker.



Obrázek 5.9. Hlavní třída knihovny a její rozhraní

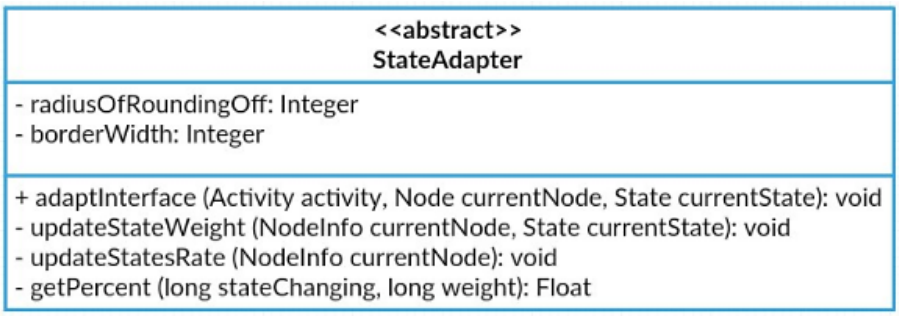
5.6 Průběh adaptace

Průběh adaptace je popsán ve třídě balíčku Utils – StateAdapter. Když pro vytvoření kompletu stavů byl využit návrhový vzor State, tak by se mohlo očekávat, že by třídy sady měly metodu na provedení změn rozhraní ve společném interfacu a přepisováním by konkretizovaly chování. Jak už je ale známé z analýzy, kvůli podmínkám programování na Android, pokud někdo chce ovlivnit cokoliv (minimálně ze strany ozdobení), pak potřebuje vědět jeho id (předepsané v xml souborech). Z toho vyplývá, že by pak každý stav měl vědět všechny UI-objekty uzlů programu, což by mělo špatný vliv na čistotu kódu. Aby oddělit realizace a držet zvlášť se za jiných okolnosti nepřekrývající informace, byla vytvořena samostatná třída, která se zabývá právě adaptací. Zveřejněná je jediná statická metoda adaptInterface(), která pro svou funkčnost potřebuje znalosti o aktuálním stavu a místě programu, a taky objekt typu Activity pro hledání objektů podle uložených id.

```
  public static void adaptInterface(Activity activity, Node currentNode,
State currentState){...}
```

Výpis 5.7. Deklarace metody adaptInterface()

Třída má i privátní metody, cílené na doprovodné úkoly, takové, jak zafixovat změnu nálady a obnovit váhy stavů.



Obrázek 5.10. Třída pro uskutečnění adaptace

5.7 Obdržení dat z databáze

Jedním z úkolů návrhu je umožnění zpětné vazby, tj. poskytování dat z uložiště na požadavek programátora. Vývojář má k dispozici možnost obdržet informaci o uzlu podle jména ve formě objektu třídy NodeInfo nebo její vypsat. Stejně pro celý obsah tabulky, tj. dostane seznam NodeInfo nebo se vypíšou data o všech uzlů aplikace. Taky lze obsah tabulky exportovat do souboru s příponou ".txt". Znázornění je vidět zde:

```
1:
{"Parent_id":"1","Quantity_of_state_changing":"282",
"NEUTRAL":"32.6241","Visits":"188","SADNESS":"6.73759",
"JOY":"30.1418","node_id":"1","Name":"MAIN_ROOT_APP_NODE",
"DISGUST":"15.2482","ANGER":"16.6667"}
2:
{"Parent_id":"1","Quantity_of_state_changing":"0",
"NEUTRAL":"0","Visits":"3","SADNESS":"0",
"JOY":"0","node_id":"2","Name":"File",
"DISGUST":"0","ANGER":"0"}
3:
{"Parent_id":"2","Quantity_of_state_changing":"1",
"NEUTRAL":"100","Visits":"1","SADNESS":"0",
"JOY":"0","node_id":"3","Name":"Create New",
"DISGUST":"0","ANGER":"0"}
4:
{"Parent_id":"2","Quantity_of_state_changing":"2",
"NEUTRAL":"50","Visits":"2","SADNESS":"0",
"JOY":"50","node_id":"4","Name":"Open",
"DISGUST":"0","ANGER":"0"}
```

Obrázek 5.11. Ukázka exportovaných dat

Soubor je standardně uložen složky telefonu "Internal storage".

5.8 Komunikace projektu a knihovny

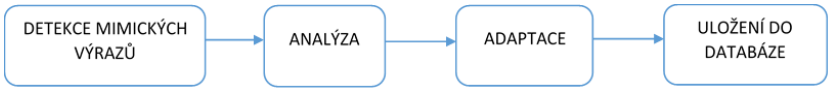
Dostupné pro vývojáře metody knihovny lze uvidět na Use Case Diagramu níž:

Obrázek 5.12. Use Case Diagram, přístupné metody

- createApplicationStructure(Activity activity, Menu menu, View mainActivityView) o pro vytváření stromu aplikace
- List¡NodeInfo¿ getAllNodesInfoFromDB() o pro získání veškeré informace, uložené v tabulce database
- NodeInfo getNodeInfoFromDB(String nodeName) o pro získání informace z databázi dle jména uzlu aplikace
- void printAllNodesInfoFromDB() o pro výpis veškeré informace, uložené v tabulce database
- void printNodeInfoFromDB(String nodeName) o pro výpis informace jednoho uzlu aplikace dle jeho jména
- void exportDB(String exportFileName) o pro uložení dat z tabulky databáze do .txt souboru s uvedeným názvem, které může být i null (v takovém případě název souboru bude vzít z názvu aplikace)
- boolean addNewState(String nameState, String backgroundColor, String textColor, String accentColor, boolean changeShape) o pro přídání nové emoce na detekování
- void deleteState(String nameState) o pro odebrání emoce
- onStop() o velmi se doporučuji dodat do vlastní přepsané metody onStop()

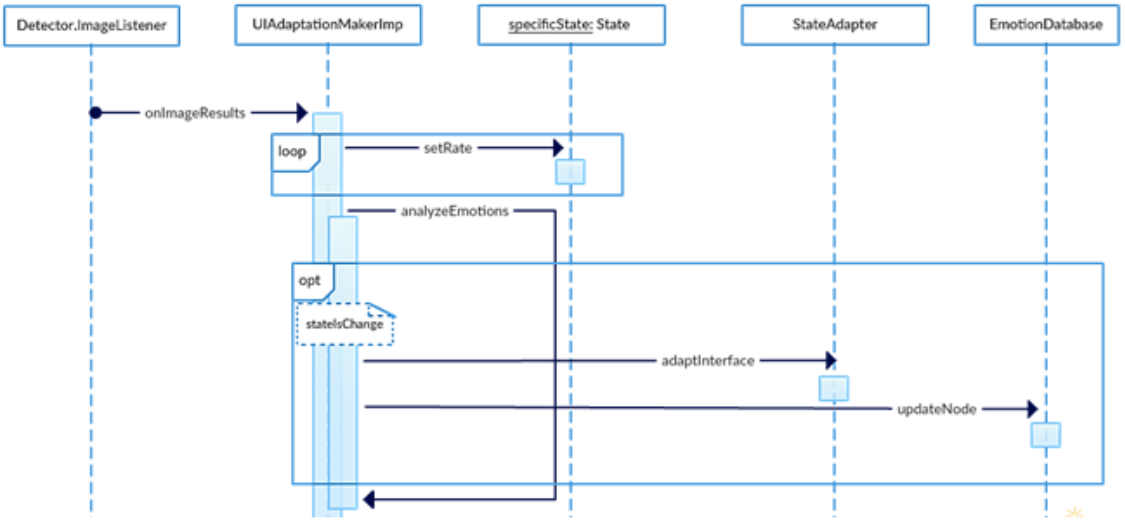
Kapitola 6
Vysvětlení toků hlavních procesů

Tato kapitola znázorňuje a ujasňuje běh hlavních funkcí knihovny. Obecný přehled je schematicky zobrazen níže:



Obrázek 6.1. Obecný přehled běhu knihovny

Ve tvaru sekvenčního diagramu to vypadá následovně:



Obrázek 6.2. Sekvenční diagram

6.1 Detekce mimických výrazů

Dotyčný krok je uskutečněn pomocí SDK Affdex od Affectivy. Automaticky se volá metoda frameworku onImageResults(), která mezi argumenty má seznam typu Face. Odtud lze obdržet zachycený obličej a vytáhnout informaci o emocích, které jsou na něm detekovány. Zmíněná metoda je přepsána, v ní se volá funkce setRate(), což nastaví v každém stavu knihovny příslušnou hodnotu z detektoru.

6.2 Analýza

Když data o měřených emocích jsou obdržené a nastavené, knihovna analyzuje, jestli se nastala změna stavu.

Jak bylo zmíněno v minulých kapitolách, Affectiva detekuje zobrazení z kamery několik krát za vteřinu. Proto je potřeba ošetřit podmínky zahájení adaptace z dvou příčin:

1. aby se vyhnulo zbytečným změnám při přečtení náhodné emoce;
2. aby nevolat metodu pro přizpůsobení, jestli byla nalezena stejná emoce, co i okamžik před tím. V tom pomůže zavedení lokální proměnné oldCurrentState, kam se při spuštění metody uloží platný stav.

```
// save the current state to temp variable
State oldCurrentState = currentState;
```

Výpis 6.1. Součást metody analiseEmotions(). č.1

Pak bude probrán každý stav na předmět toho, jestli byl nalezen detektorem nebo ne, voláním metody isNeutral() – pozitivní odpověď metody znamená, že ptána emoce je v neutrálním stavu čili nedetekována.

Specifický případ je neutrální stav. Affectiva odhaluji emoce na základě mimických vlastností (víz Kapitola 2.3.3), ale neutrální stav se nedoprovází žádným speciálním výrazem obličeje. Jedině, dá se ho definovat absencí rys emocí.

Z toho důvodu volání isNeutral() na neutrálním stavu přivede k chybným výsledkům, neboť vždy bude vracet pravdivý výrok. Proto za kontroly změnil-li se stav, neutrální se vynechává.

```
for (State state : stateCollection) {

// avoid to check neutral state if is valid
if (!state.getName().equalsIgnoreCase(neutralString) && !state.isNeutral()){
...
}
}
```

Výpis 6.2. Součást metodu analiseEmotions(). č.2

Skutečnost, že nastal právě on, bude zajištěna tím, že se nastaví od začátku a pokud žádný jiný stav se neobjeví, zůstane jako platný.

Takže celkem etapa analýzy změny stavu vypadá takovým způsobem:

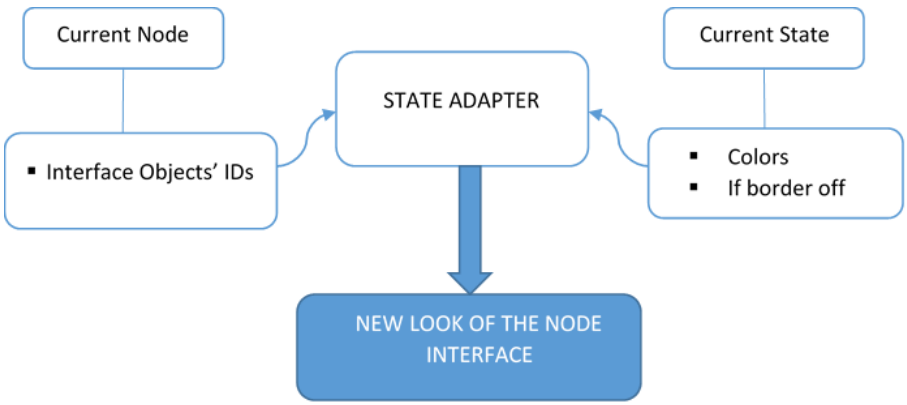
```
private void analiseEmotions() {
// get states' names only
Collection<State> stateCollection = states.getStatesLikeCollection();

// save the current state to temp variable
State oldCurrentState = currentState;
```

```
// set current state like neutral
currentState = states.getState(NEUTRAL_STATE);

if(states != null) {
String neutralString = currentState.getName();

for (State state : stateCollection) {

// avoid to check neutral state if is valid
if (!state.getName().equalsIgnoreCase(neutralString) && !state.isNeutral()){
currentState = state;
break;
}
}
}

//reset counting of detected emotion changing (reason 1)
if(!oldCurrentState.equals(currentState)){
currentState.setChanging(0);
}
//verify if current emotion and detected emotion are different (reason 2)
else if(currentState.ifChange()) {
StateAdapter.adaptInterface(activity, currentNode, currentState);
db.updateNode(currentNode);
}
}
```

Výpis 6.3. Součást metody analiseEmotions()

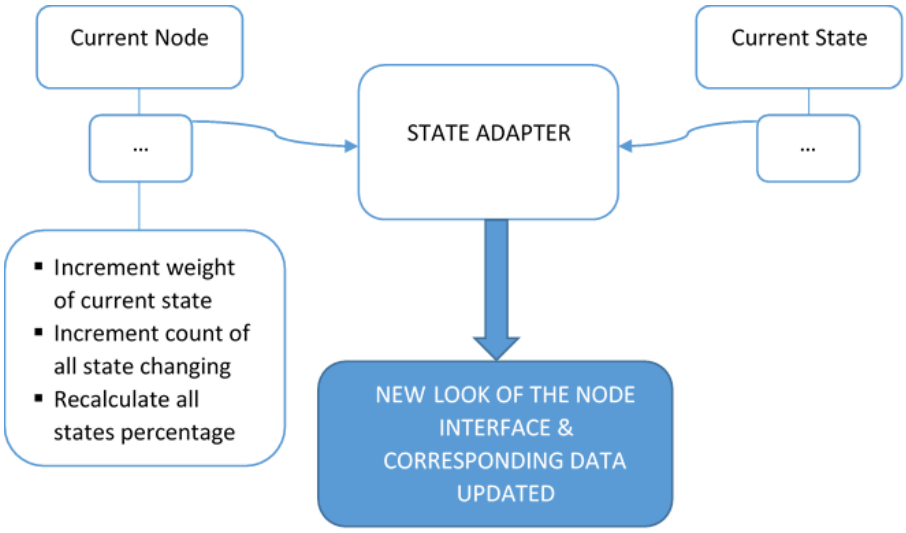
6.3 Adaptace rozhraní a aktualizace dat

Jestli došlo ke změně, zavolá se statická metoda adaptInterface() třídy StateAdapter. Jako argumenty přijímá aktuální stav a uzel aplikace. Na základě obdržených dat (víz obrázek níž) provede potřebné změny.



Obrázek 6.3. Znázornění etapy adaptace

Spolu s tím se metoda stará i o aktualizaci dat uvnitř instancí třídy NodeInfo, která jsou určená pro uložení do databáze.



Obrázek 6.4. Znázornění etapy adaptace, pokračování

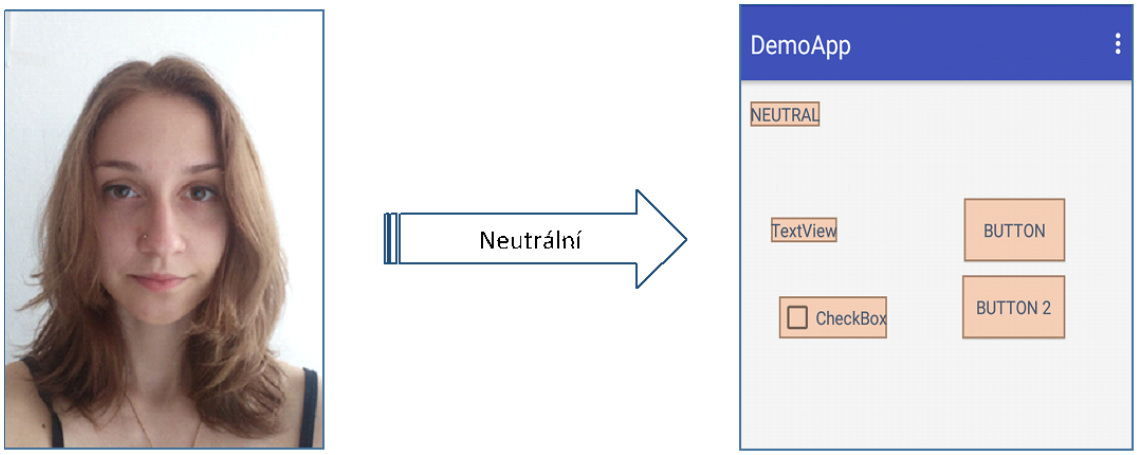
6.4 Uložení do databáze

Etapa se provede tím, že se v tabulce zaktualizuje záznam o nodu.

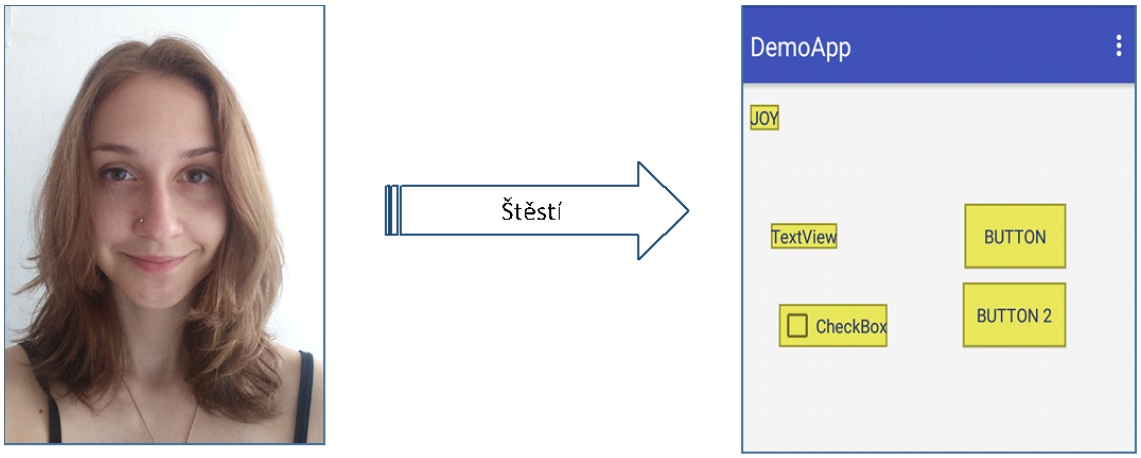
Kapitola 7
Ukázka knihovny v provozu

7.1 Demonstrace adaptačních procesů

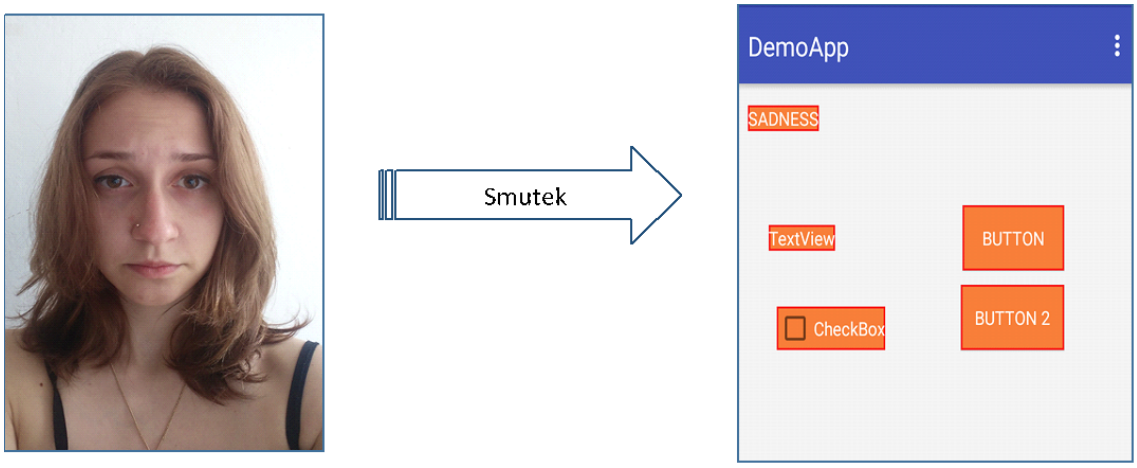
Níže lze vidět, na jaké výrazy obličeje a jak reaguje projekt v realitě:



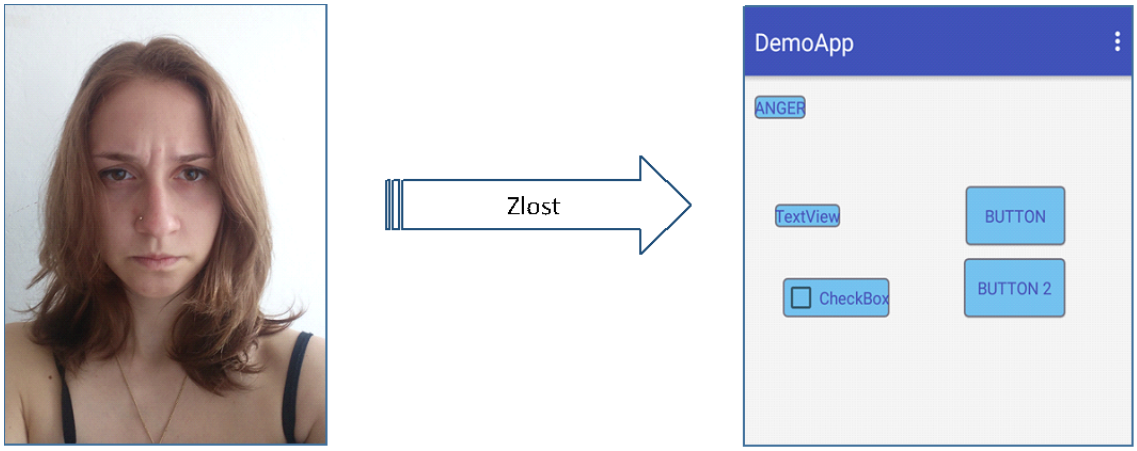
Obrázek 7.1. Demonstrace adaptace rozhrání pod Neutrální stav



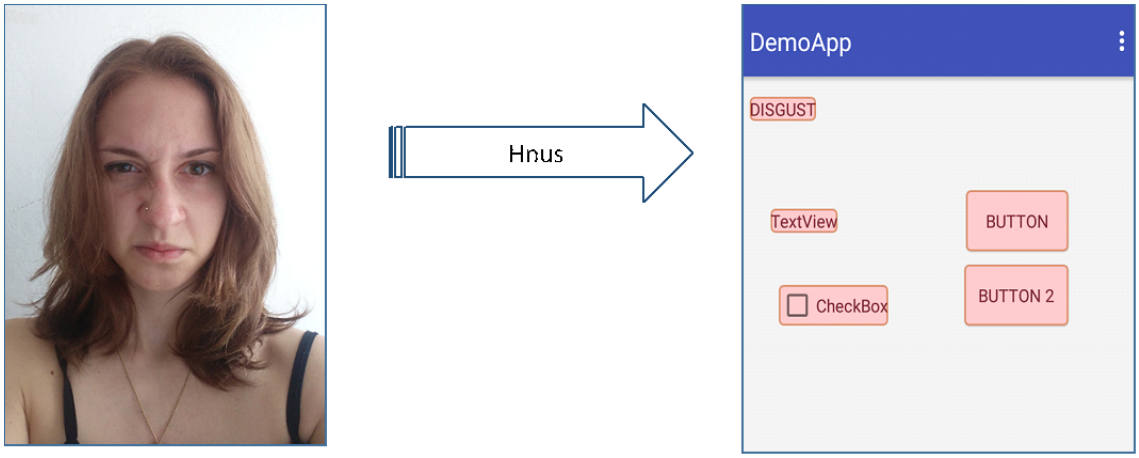
Obrázek 7.2. Demonstrace adaptace rozhrání pod stav Štěstí



Obrázek 7.3. Demonstrace adaptace rozhrání pod stav Smutek



Obrázek 7.4. Demonstrace adaptace rozhrání pod stav Zlost



Obrázek 7.5. Demonstrace adaptace rozhrání pod pod stav Hnus

7.2 Využití dat z databáze

Reakce uživatele, jeho emoce, mohou posloužit jako zpětná vazba pro vývojáře. Na jejich základě se dá posoudit, jak dotyční vnímají aplikaci. V této sekci jsou uvedené několik příkladů rozboru dat.

- Stejný stav se projevuje v uzlů jedné větví stromu menu.
  - To by s velkou pravděpodobností mohlo znamenat, že problém se schovává v rodičovském uzlu.
- Celý strom menu má pozitivní nebo neutrální reakce, ale nějaký uzel má negativní. V tomto případě příčinou může být:
  - Nepohodlné rozmísťování elementů, což vede k delšímu a složitějšímu hledání potřebného;
  - Pomalá odezva;
  - Nepříjemné ozdobení;
  - Neočekávané chování aplikace;
  - Nesrozumitelný kontent.
- Celý strom stránek zobrazuje negativní zpětnou vazbu.
  - Jestli statistika ukazuje pravidelnou převahu negativních reakcí, může se to považovat za nástin programátorovi přehodnotit strukturu a náplň aplikace.

Případy se neomezují jenom probranými body. Při rozboru dat je třeba dávat pozor na zákonitosti a náhodnosti. Taky je třeba brát v úvahu to, že ne všechno v rozpoložení uživatele záleží na aplikaci. Příčinou může být okolní hluk, nepohodlná obuv, osobní problémy, únava anebo jednoduše špatná nálada atd.

Kapitola 8
Požadavky

- mobilní telefon s operačním systémem Android OS verze minimálně 4.4 (KitKat);
- přítomnost přední kamery;
- aplikace používá Java verze 8;
- v hlavním layout aplikace (standardně `activity_main.xml`) mít vytvořeny Camera-Preview:

```
<SurfaceView
android:id="@+id/cameraPreview"
android:layout_width="1px"
android:layout_height="1px"
android:background="@android:color/transparent"
/>
```

Výpis 8.1. Součást xml souboru

Kapitola 9
Instalace

1. Dodat knihovnu do složky aplikace “libs”. Cesta je: `Application_Name->app->libs`. Pokud složka neexistuje – vytvořit.

2. V souboru build.gradle projektu dodat do repositories:

```
maven {
url "http://maven.affectiva.com"
}
```

Výpis 9.1. Instalace: součást soubiru build.gradle projektu

3. V souboru build.gradle modulu aplikace dodat do dependences:

```
compile fileTree(dir: 'libs', include: ['*.jar'])
compile 'com.affectiva.android:affdexsdk:3.1.2'
```

Výpis 9.2. Instalace: součást build.gradle modulu

4. Do AndroidManifest.xml dodat následující záznamy:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
…
<application
tools:replace="android:allowBackup,android:label"
android:allowBackup="false"
…
```

Výpis 9.3. Instalace: součást souboru “AndroidManifest.xml”

5. V MainActivity deklarovat objekt typu UIAdaptationMaker:

```
public class MainActivity extends AppCompatActivity {
private TextView textView;
/*
other class variables
*/
private UIAdaptationMaker uiAdaptation;
}
```

Výpis 9.4. Instalace: součást třídy MainActivity

6. V metodě onCreate() nadefinovat výš označenou proměnnou:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.setContentView(R.layout.activity_main);

textView = (TextView) findViewById(R.id.textView);

/*
other variable definitions
*/

uiAdaptation = UIAdaptationMakerImpl.getUIAdaptationMaker(this,
(SurfaceView)findViewById(R.id.cameraPreview));

}
```

Výpis 9.5. Instalace: součást metody onCreate() třídy MainActivity

7. V metodě onCreateOptionsMenu(Menu menu) zavolat metodu createApplication-Structures(), i když aplikace menu nemá (v tomto případě dát null).

```
@Override
public boolean onCreateOptionsMenu(Menu menu){
uiAdaptationMaker.createApplicationStructure(
this, null, getLayoutInflater().inflate(R.layout.activity_main, null));
return true;
}
```

Výpis 9.6. Instalace: součást metody onCreateOptionsMenu() třídy MainActivity

Kapitola 10
Závěr

Cílem bakalářské práce bylo vymyslet a probrat problematiku adaptace rozhraní, založené na emocích uživatele, v souvislosti s tím prostudovat vliv barevné charakteristiky rozhraní na dojem člověka a navrhnout implementaci myšlenky. Realizace by měla přinášet užitek jak uživateli, tak i vývojáři: pro prvního to znamená snahu pozitivně ovlivnit jeho stav, pro druhého – umožnit analýzu vlastní aplikace, opírající se na data o pocitech uživatelů.

V první části práce byla prozkoumána teorie, která se týká adaptace celkově, vybudování uživatelského modelu aplikace, způsoby vyjádření člověkem vlastních emocí, rozeznávání jich na základě mimiky a taky jaký vliv má koloristika na náladu. Druhá část je věnována návrhu implementačního řešení.

Výsledkem je rozpracována knihovna, založená na výš uvedených studiích, která dovolí rozšířit chápání pojmu ozdobení rozhraní a je doprovázena statistickou informaci o vnitřních dojmech uživatele.

Literatura

[1] Definition of User Interface. *InterfyeisBlogspot* [online]. 2012 cit.[2016-11-15]. Dostupné z:
http://interfyeis.blogspot.cz/2012/03/blog-post.html.

[2] Representation of communication between human and system. In: Bricks Narod [online]. Russia [cit. 2017-05-14]. Dostupné z:
http://bricks.narod.ru/tp/31_pon.htm.

[3] MCGRENERE, Joanna a Leah FINDLATER *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: A Comparison of Static, Adaptive, and Adaptable Menus* [online]. Vienna, Austria: ACM, 2004, 89-96 [cit. 2016-12-18]. DOI: 10.1145/985692.985704. ISSN: 1-58113-702-8.

[4] HOTHI, Jatinder a Wendy HALL. *An Evaluation of Adapted Hypermedia Techniques Using Static User Modelling.* Electronics and Computer Science University Road, Southampton, Hampshire, UK, 1998.

[5] SHUMKINA, Vera. *Methods of User Interface Adaptation* Novosibirsk, 2013. Disertace. Novosibirsk State University. Leadership D. Palchunov.

[6] Dr. Paul Ekman. *Paul Ekman Group* [online]. Washington, D.C., 2017 [cit. 2017-04-30]. Dostupné z:

[7] Domovská stránka frameworku.*Affectiva* [online]. Boston, USA: Affectiva, 2017 [cit. 2016-11-10]. Dostupné z:
https://developer.affectiva.com.

[8] Metrics. *Affectiva* [online]. Boston, USA: Affectiva, 2017 [cit. 2016-11-14]. Dostupné z:
developer.affectiva.com/metrics/emotions.

[9] *Affectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected In-the-Wild* [online]. 2017, 8 [cit. 2017-05-14]. Dostupné z:
http://www.affectiva.com/wp-content/uploads/2017/03/Affectiva-MIT-Facial-Expression-Dataset-AMFED-Naturalistic-and-Spontaneous-Facial-Expressions-Collected-In-the-Wild.pdf

[10] Mapping Expressions to Emotions. *Affectiva* [online]. Boston, USA, 2017 [cit. 2016-11-14]. Dostupné z:
http://developer.affectiva.com/emotion_mapping/.

[11] BAZYMA, Boris. *Color psychology: theory and practice.* 1. Harkov: Ryech, 2005. ISBN 5-9268-0363-2.

[12] Color Psychology. *Kendra Cherry, VeryWell* [online]. [cit. 2016-11-04]. Dostupné z:
https://www.verywell.com/color-psychology-2795824.

[13] D. TUFFELMIRE Do Colors Affect Children's Learning?
http://www.ehow.com/facts_7358881_do-colors-affect-children_slearning_.htm.

[14] BUCKALEW, W. a BELL. Effects A. of Colors on Mood in the Drawings of Young Children. *Perceptual and Motor Skills* [online]. 1985, 3 [cit. 2017-05-14]. DOI: 10.2466/pms.1985.61.3.689. Dostupné z: http://journals.sagepub.com/doi/abs/10.2466/pms.1985.61.3.689?journalCode=pmsb.

[15] ERNST, E. a A. HERXHEIMER. Effect of colour of drugs: systematic review of perceived effect of drugs and of their effectiveness.*PubMed.* Netherlands: Department of Clinical Epidemiology, Academic Medical Centre, University of Amsterdam, 1996, (7072), 21-28.

[16] SHIMBUN, Yomiuri Blue streetlights believed to prevent suicides, street crime. *The Seattle Times* [online]. 2008, 1 [cit. 2016-12-18]. Dostupné z: http://www.seattletimes.com/nation-world/blue-streetlights-believed-to-prevent-suicides-street-crime/.

[17] ELLIOT, A.J. a H. AARTS. Perception of the color red enhances the force and velocity of motor output. *Journal of Personality and Social Psychology.* New York: Department of Clinical and Social Sciences in Psychology, University of Rochester, Rochester, 2011, **14627**. DOI: 10.1037/a0022599.

[18] FRANK, M.G. a T. GILOVICH. The dark side of self- and social perception: Black uniforms and aggression in professional sports. *Journal of Personality and Social Psychology.* New York: Department of Psychology, Cornell University, Ithaca, 1988, **14853**(Vol 54(1)), 74-85.

[19] ELLIOT, Andrew J., Markus A. MAIER, Arlen C. MOLLER a Ron FRIEDMAN.Color and psychological functioning: The effect of red on performance attainment. *Jörg Journal of Experimental Psychology: General.*, 2007, Vol 136(1), 154-168. DOI: 10.1037/0096-3445.136.1.154.

[20] MISHCHENKO, Nikita. *Aspektově orientovaný vývoj adaptivní struktury aplikace pro Java EE aplikace.* Prague, 2016. Diploma thesis. Department of Computer Science, CTU FEE. Leadership J. Šebek.

[21] GAMMA, Erich, John VLISSIDES, Ralph JOHNSON a Richard HELM. *Design Patterns: Elements of Reusable Object-Oriented Software.*, St. Petersburg: Addison-Wesley, 2015. ISBN: 978-5-496-00389-6.

Příloha A
Zadání práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Anastasiia Lunova

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Adaptace UI založena na emocích uživatele

Pokyny pro vypracování:

Prostudujte možnosti adaptivních uživatelských rozhraní a nástrojů pro zachycení emocí uživatele. Vytvořte framework pro Android zařízení s možností adaptací UI dle emocí. Důležitá část práce je správně navrhnout model ukládání kontextových informací. Proveďte testy funkčnosti. Demonstrujte použití na modelovém příkladě. Výsledná implementace musí být snadno rozšiřitelná. Zhodnoťte výhody a možná omezení řešení.

Seznam odborné literatury:

1. ŠEBEK, J. and K. RICHTA. Aspect-oriented User Interface Design for Android Applications [online]. In: DATESO 2015. Databases, Texts, Specifications, and Objects 2015, Nepřívěc u Sobotky, Jičín, 2015-04-14/2015-04-16. Praha: MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty UK, 2015, pp. 121-130. CEUR Workshop Proceedings. vol. 1343. ISSN 1613-0073. ISBN 9788073782856. Available from: http://www.cs.vsb.cz/dateso/2015/
2. HODAKOV, Viktor. Adaptive User Interface: Problems Of Building. Automation Electrotechnical Complexes and Systems. 2003, 1(11), 45-57. ISBN 5-7763-8361-7.
3. Šebek, J. - Richta, K.: Usage of Aspect-Oriented programming in Adaptive Application Structure. New Trends in Databases and Information Systems: ADBIS 2016 Short Papers and Workshops, BigDap, DCSA, DC, Prague, Czech Republic, August 28-31, 2016, Proceedings

Vedoucí: Ing. Jiří Šebek

Platnost zadání: do konce zimního semestru 2018/2019

L.S

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 4.4.2017

Příloha B
Znázornění a popis mimických výrazů

Attention – Measure of focus based on the head orientation

Brow Furrow – Both eyebrows moved lower and closer together

Brow Raise – Both eyebrows moved upwards

Cheek Raise – Lifting of the cheeks, often accompanied by "crow's feet" wrinkles at the eye corners

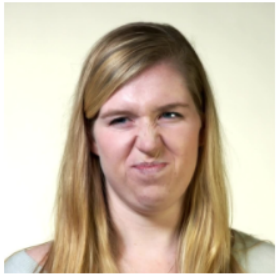
Chin Raise – The chin boss and the lower lip pushed upwards

Dimpler – The lip corners tightened and pulled inwards

Eye Closure – Both eyelids closed

Eye Widen – The upper lid raised sufficient to expose the entire iris

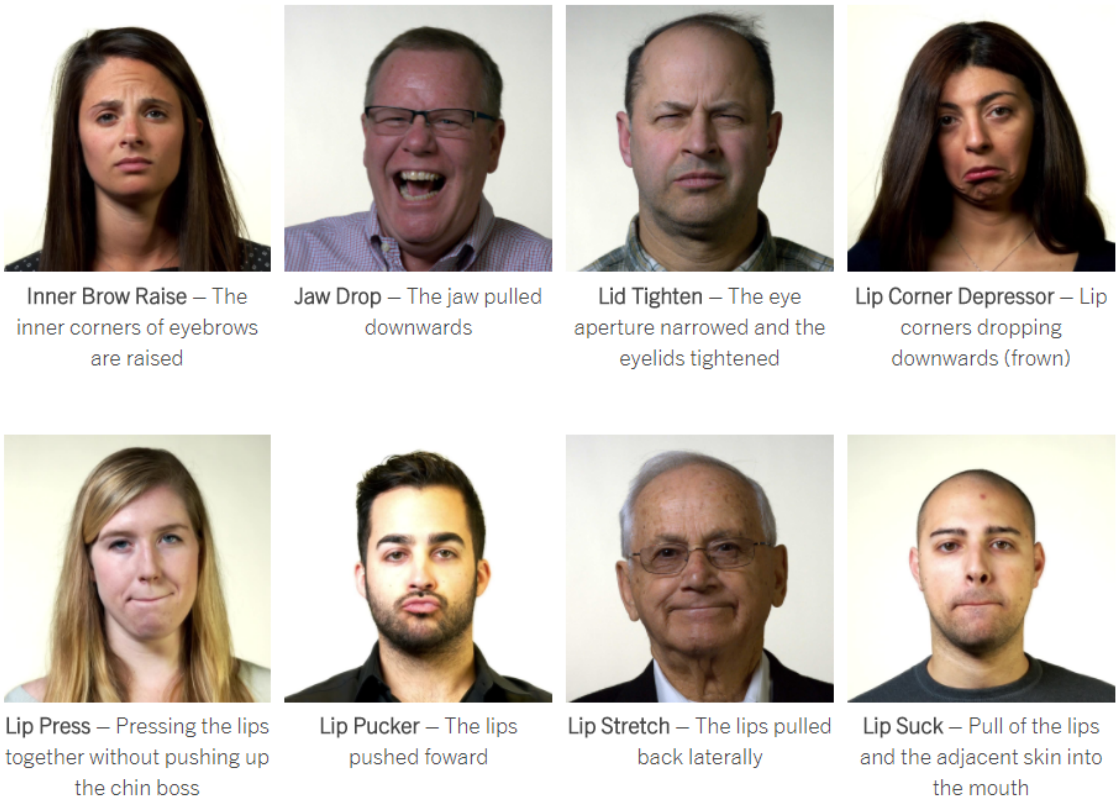
Mouth Open – Lower lip dropped downwards



Nose Wrinkle – Wrinkles appear along the sides and across the root of the nose due to skin pulled upwards

Smile – Lip corners pulling outwards and upwards towards the ears, combined with other indicators from around the face

Smirk – Left or right lip corner pulled upwards and outwards



Obrázek B.1. Znázornění a popis mimických výrazů.

Příloha C
Príklady kódu

Příloha D
Obsah priloženého CD

- EmotionAdaptationLibrary.zip . . . . . Kódy implementované knihovny
- DemoApp.zip . . . . . Ukázková aplikace s připojenou knihovnou
- images.zip . . . . . Použité obrázky
- diagrams.zip . . . . . Použité diagramy
- bp_lunovana.pdf . . . . . Text bakalárské práce
- zadani_bp.pdf . . . . . Zadaní bakalárské práce
- bp_lunovana_tex.zip . . . . . Zdrojové soubory k textu bakalárské práce

Příloha E
Použité zkratky

UI	User Interface / Uživatelské rozhraní
AUI	Adaptive User Interface / Adaptivní uživatelské rozhraní
SDK	Software development kit / Sada vývojových nástrojů
AU	Action Unit / Pohybová jednotka
EMFACS	Emotional Facial Action Coding System
ROI	Region of Interest / Oblast zájmu
HOG	Histogram of Orientated Gradients / Histogram Orientovaných Gradientů