FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Přihlašování obličejem pro KDE
Student:	Petr Dušek
Vedoucí:	Ing. Ondřej Guth, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Proveďte rešeršní rozbor projektů implementujících rozpoznávání obličeje, které jsou použitelné pro autentizaci v operačním systému GNU/Linux. Prozkoumejte možnosti, jakými lze rozšířit o rozpoznávání obličeje přihlašování a odemykání plochy v prostředí KDE. Na základě těchto průzkumů realizujte rozšíření přihlašování a odemykání plochy v KDE o rozpoznávání obličeje. Tato realizace bude vyžadovat co nejméně změn v kódu stávajících projektů a bude uživatelem snadno nastavitelná, stejně tak jednoduchým způsobem umožní jiný způsob přihlašování. Možnost nastavení proveďte v souladu se zvyklostmi v prostředí KDE, v nastavení umožněte i správu samotného rozpoznávání obličejů. Vytvořte instalátory (balíčky) pro distribuce GNU/Linux minimálně v rozsahu Mint, OpenSUSE, Gentoo. Celé řešení zhodnoťte z hlediska bezpečnosti a vhodnými prostředky otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Přihlašování obličejem pro KDE

Petr Dušek

Katedra softwarového inženýrství
Vedoucí práce: Ing. Ondřej Guth, Ph.D.

27. června 2019

Poděkování

Chtěl bych tímto poděkovat vedoucímu Ing. Ondřeji Guthovi, Ph.D. za pomoc, cenné rady a připomínky při tvorbě této práce. Dále bych chtěl poděkovat celé své rodině a přátelům za jejich trpělivost a morální podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. června 2019 ……

Odkaz na tuto práci

Dušek, Petr. *Přihlašování obličejem pro KDE*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato bakalářská práce popisuje vznik aplikace implementované jako systémový modul pro prostředí KDE Plasma 5. Aplikace využívá jako svůj základ projekt Howdy. Hlavní náplní práce je popis vzniku aplikace a příslušných instalačních balíčků pro distribuce Linux Mint, Gentoo a OpenSUSE. Práce obsahuje analýzu, návrh, postup řešení jednotlivých problémů a konečné otestování.

Klíčová slova rozpoznání obličeje, KDE, Plasma 5, systémový modul, linux přihlašování, instalační balíčky, PAM

Abstract

This bachelor thesis describes the creation of an application implemented as a system module for KDE Plasma 5. The application uses the Howdy project as its basis. The main task of the thesis is to describe the creation of the application and installation packages for distributions Linux Mint, Gentoo and OpenSUSE. The thesis contains analysis, design, implementation, solution individual problems and final testing.

Keywords face recognition, KDE, Plasma 5, system module, linux authentication, install packages, PAM

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Operační systém Windows nabízí několik funkcí, které se dají použít pro přihlášení. Tyto funkce využívají biometrických technologií k rozpoznání uživatelů. Patří mezi ně přihlášení pomocí otisků prstů, skenováním očí a nebo pomocí obličeje. Odemykání obličejem a pomocí otisků se zabývá funkce Windows Hello, která pro identifikaci lidského vzhledu využívá infračervené kamery kvůli vyšší bezpečnosti.

Rozpoznávání obličeje, stejně tak jako otisků prstů, vyžaduje určité modely. Bez těchto modelů by nebylo možné určit, jak má vypadat správná hodnota. Je to podobné jako při ověřování hesla. Zadané heslo do systému musí být totožné s heslem uloženým v databázi. Pokud se hesla neshodují, autentizační systém přístup odmítne. Rozpoznání obličeje funguje na stejném principu, s vyjímkou toho, že místo hesla se používá model obličeje. Tento model je již provedený sken obličeje. Avšak zde většinou neplatí pravidlo přímé totožnost porovnávaných modelů. Vyžaduje to totiž velmi přesná a spolehlivá zařízení, která zajistí sken obličeje na vysoké úrovni. Proto se u méně kvalitnějších zařízení definuje přesnost. Znamená to, že pokud při porovnání dvou modelů nastane odchylka, která je menší než požadovaná přesnost, jsou vyhodnocené jako stejné.

V operačním systému GNU/Linux existují podobná řešení na tento způsob odemykání. Většina z nich ovšem není téměř použitelná (nebo ve velmi omezené míře) a běžný uživatel si je nedokáže sám jednoduše nastavit nebo zprovoznit.

GNU/Linux je velmi zajímavý díky své variabilitě, každý si může vybrat distribuci, která mu nejvíce vyhovuje, a také desktopové prostředí, kterých je k dispozici velké množství. Mezi nejpopulárnější patří Gnome, Unity, Xfce, Cinnamon, KDE a další.

KDE je desktopové prostředí pro Windows, GNU/Linux a další unixové operační systémy, které je vystavěné nad knihovnou Qt. Je zvykem, že každý program, kterým se nastavují nebo upravují funkce systému nebo funkce pří-

davných zařízení, má svůj modul obsažený v systémovém nastavení. Žádné z objevených řešení není na takové úrovni, že by se dalo ovládat jinak než z příkazové řádky. Proto bude třeba vytvořit systémový modul, který by jednoduše s uživatelsky přívětivým vzhledem zobrazoval různé informace a umožňoval uživatelům nastavit si program podle svých potřeb.

Pro operační systém Linux se obvykle při instalaci využívají instalační balíčky. Každá distribuce má svůj preferovaný formát balíčku. Bude tedy potřeba vytvořit instalační balíčky pro jednotlivé distribuce linuxu a celé řešení otestovat.

Kapitola 1

Cíl práce

Cílem práce je provedení rešerše stávajících projektů zabývajících se odemykáním pomocí obličeje pro systém GNU/Linux a nejlépe fungující řešení využít pro přihlášení a odemykání plochy v systému KDE Plasma 5.

Dalším cílem je analýza, návrh a implementace systémového modulu v prostředí KDE Plasma 5. Systémový modul bude mít GUI a umožní uživateli snadné nastavení fungování vybraného projektu.

Důležitou součástí je také vytvoření instalačních balíčků pro distribuce Gentoo, Linux Mint a OpenSUSE. Nakonec bude nutné aplikaci zhodnotit z hlediska bezpečnosti a otestovat vhodnými prostředky.

Kapitola 2

Současná řešení pro systém Linux

Tato kapitola popisuje nalezené projekty, které se zabývají přihlašováním pomocí obličeje pro systém Linux. Současná řešení jsem hledal na webu. Zjistil jsem, že zatím neexistuje aplikace, která by se zabývala odemykáním pomocí obličeje a byla implementována přímo jako systémový modul pro prostředí KDE plasma 5.

Narazil jsem však na několik řešení, které se zabývají rozpoznáním obličeje a odemykáním Linuxu s využitím modulů pro ověřování nazývané PAM. PAM je sada sdílených knihoven, která umožňuje správci systému zvolit způsob, jakým aplikace ověřují uživatele. Jinými slovy, je možné přepínat mezi autentizačními mechanismy a autentizační systém zcela měnit, bez úpravy samotných aplikací [1].

2.1 Pam-facial-auth

Projekt, který je napsaný v jazyce C++. Zprovoznění projektu se skládá z několika kroků. Prvním krokem je spuštění skriptu *setup.sh*. Tím se vytvoří složka build, ve které proběhne sestavení aplikace nástrojem CMake a vzniklé spustitelné soubory se překopírují do předchozího adresáře. Spustitelnými soubory jsou *run_test* a *run_training*. Je také vytvořen sdílený objekt *facialauth.so*, který je následně zkopírován do adresáře */lib/security*, kde se nachází PAM moduly. Dalším krokem je spuštění souboru *run_training* s parametrem, obsahujícím jméno složky s fotografiemi uživatele. Ta musí mít přesně definovanou strukturu, viz obrázek 2.1. Spuštěním skriptu se pomocí knihovny OpenCV detekuje obličej s potřebnými parametry a model se uloží do souboru *model-facerec.xml* v adresáři */etc/pam-facial-auth* .

Posledním krokem je zprovoznění PAM modulu. Pokud chceme, aby se modul spouštěl při přihlášení, je třeba ho přidat do příslušného PAM konfigu-

```
data_dir ..... složka předávaná trénovacímu algoritmu
├── username1 ..... složka s názvem uživatele
│   ├── image1 ..... obrázek s obličejem uživatele
│   ├── ..... obrázek s obličejem uživatele
│   └── imageN ..... obrázek s obličejem uživatele
└── username2 ..... složka s názvem jiného uživatele
    └── ..... ..... složka s obličejem uživatele
```

Obrázek 2.1: Struktura složky s modely

račního souboru v adresáři */etc/pam.d*. Jelikož se při spouštění skriptu v tomto adresáři vytvořil také PAM konfigurační soubor *pam_test*, stačí jednoduše přidat text **@include pam_test**. Pokud chceme například v distribuci Ubuntu ověřit uživatele při přihlášení, přidáme text **@include pam_test** do souboru */etc/pam.d/common-auth*.

Ve srovnání s ostatními projekty je složitější na rozběhnutí a není možné si nijak upravit chování programu bez zásahů do kódu. Při ověřování uživatele u příkazu **sudo -i** vrací modul chybu a není možné uživatele ověřit. Projekt je dostupný veřejně v github repozitáři, viz [2].

2.2 PAM Face

Jedná se o funkční projekt v jazyce Python, který využívá PAM k autentizaci a o detekci obličeje se stará knihovna OpenCV, viz [3].

Oproti předchozímu projektu není třeba vytvářet fotografie uživatele. Program lze spouštět pouze z příkazové řádky a obsahuje příkazy pro přidání a odebrání modelů obličeje.

Tento projekt je možný si pro distribuce založené na Debianu nainstalovat pomocí instalačního balíčku, který si však uživatel musí vytvořit z připravených souborů dostupných na githubu projektu, viz [4]. Pro ostatní distribuce je nutné si program nainstalovat manuálně.

2.2.1 Struktura

Soubory, se kterými tento projekt pracuje se instalují do několika adresářů. Jsou to adresáře:

- */usr/share/pam-configs*
- */lib/security*
- */usr/bin*
- */etc/pam-face*

/usr/share/pam-configs

Tento adresář obsahuje definice PAM modulů, které jsou v distribucích Debian instalovány pomocí nástroje **pam-auth-update**. Je sem zkopírován soubor *face*, kterým se přidá PAM modul tohoto projektu.

/lib/security

Tento adresář je obecně využíván jako adresář obsahující PAM moduly. V tomto adresáři by se po instalaci měl nacházet soubor *pam_face.py*. Jedná se o PAM modul napsaný v jazyce Python.

/usr/bin

V tomto adresáři se v operačním systému GNU/Linux nacházejí spustitelné soubory. Na toto místo je zkopírován soubor *pamface-conf*, který spouští celý projekt.

/etc/pam-face

Poslední využitý adresář má v sobě uložené modely uživatelů a konfigurační soubor, ve kterém jsou definovaná jména uživatelů. Všechny modely uživatelů se nachází v XML souboru *models.xml*, kde jsou uložená pole dat obsahující informace o jednotlivém modelu a id uživatele. Jméno uživatele se získá z konfiguračního souboru podle definovaného id.

2.2.2 Rozpoznání a přidání obličeje

Knihovna OpenCV nabízí algoritmy trénované k detekci různých objektů. K detekci obličeje, připadně očí, nebo úsměvů slouží XML klasifikátory. Projekt PAM Face detekuje obličej pouze na základě celého obličeje. Neřeší detailněji pozici očí, nosu, nebo úst.

Klasifikátor *haarcascade_frontalface_default.xml* knihovny OpenCV se stará o zjištění pozice obličeje a vrací jeho lokaci na fotografii. Poté se z těchto lokací rozpoznává obličej metodou LBPH nabízenou knihovnou OpenCV. Tato metoda využívá porovnávání jednotlivých pixelů s okolím. Výsledná data z rozpoznání obličejů jsou uložena do souboru *models.xml*.

Přidávání obličeje může uživatel zároveň sledovat v okně, které se zavoláním příkazu pro přidání objeví. Zobrazí se výstup z webkamery a detekované obličeje jsou ohraničené čtvercem, aby bylo vidět, které obličeje byly detekovány.

Informace v této sekci jsou získané na základě zkoumání zdrojového kódu.

2.2.3 Porovnání obličejů

Porovnání obličejů je řešeno na základě uložených dat a výstupu z webkamery. Provádí se stejný postup jako při rozpoznání a přidání obličeje, s tím rozdílem, že se data neukládají ale pouze porovnávají s uloženými. Na základě toho je uživatel identifikován.

2.2.4 Přihlašování a odemykání

Přihlášování i odemykání je řešeno pomocí PAM modulu. Python nabízí knihovnu Pam-python, díky které je možné psát PAM moduly přímo v jazyce Python.

PAM modulem je soubor *pam_face.py*, který má definované tyto funkce:

- *pam_sm_authenticate()*
- *pam_sm_setcred()*
- *pam_sm_acct_mgmt()*
- *pam_sm_open_session*
- *pam_sm_close_session()*
- *pam_sm_chauthtok()*

Všechny funkce kromě *pam_sm_authenticate()* vrací stejnou hodnotu PAM_SUCCESS. Řeší se zde pouze získání autentizace, nikoli změny autentizačního tokenu, nebo povolení přístupu a dalších autentizací, ke kterým slouží ostatní funkce.

pam_sm_authenticate()

Tato funkce zajišťuje získání práv při ověření uživatele. Je volána například při zavolání **sudo**, nebo při přihlášení na odemykací obrazovce. Autentizace je popsána v podsekci Autentizace.

pam_sm_setcred()

Touto funkcí se získávají credentials pro uživatele.

pam_sm_acct_mgmt()

Tato funkce má za úkol zjistit, zda je v tomto okamžiku uživateli povolen přístup.

pam_sm_open_session

Tato funkce je volána při zahájení nové relace.

pam_sm_close_session()

Funkce, která je volána při ukončení relace.

pam_sm_chauthtok()

Funkce se používá ke změně autentizačního tokenu pro daného uživatele.

2.2.5 Autentizace

Získání autentizace záleží na výsledku rozpoznání obličeje aktuálního uživatele a porovnání s uloženými modely.

Pokud je ověřován neznámý uživatel, který nemá přidaný žádný model, je vrácena chyba hodnotou PAM_USER_UNKNOWN, značící, že uživatel je neznámý. V opačném případě je spuštěň rozpoznávací algoritmus. Pokud byl uživatel rozpoznán, je navrácena hodnota PAM_SUCCESS. V opačném případě se vrací hodnota PAM_AUTH_ERR.

Ve všech ostatních případěch, které můžeme považovat za neznámé případy chyb, je vrácena hodnota PAM_AUTH_ERR.

2.3 Howdy

Howdy je projekt napsaný v programovacím jazyce Python. Pro identifikaci obličeje a jeho rozpoznání slouží knihovna dlib, která využívá algoritmů strojového učení. Dlib modely jsou k dispozici v github repozitáři, viz [5].

Howdy lze použít pouze z příkazové řádky a ke spuštění je vyžadován účet root. Obsahuje příkazy pro přidání, odebrání a vymazání obličeje pro konkrétně přihlášeného uživatele nebo nastavení aplikace. Dále si uživatel může zobrazit své přidané modely jako seznam obsahující id, datum přidání a název.

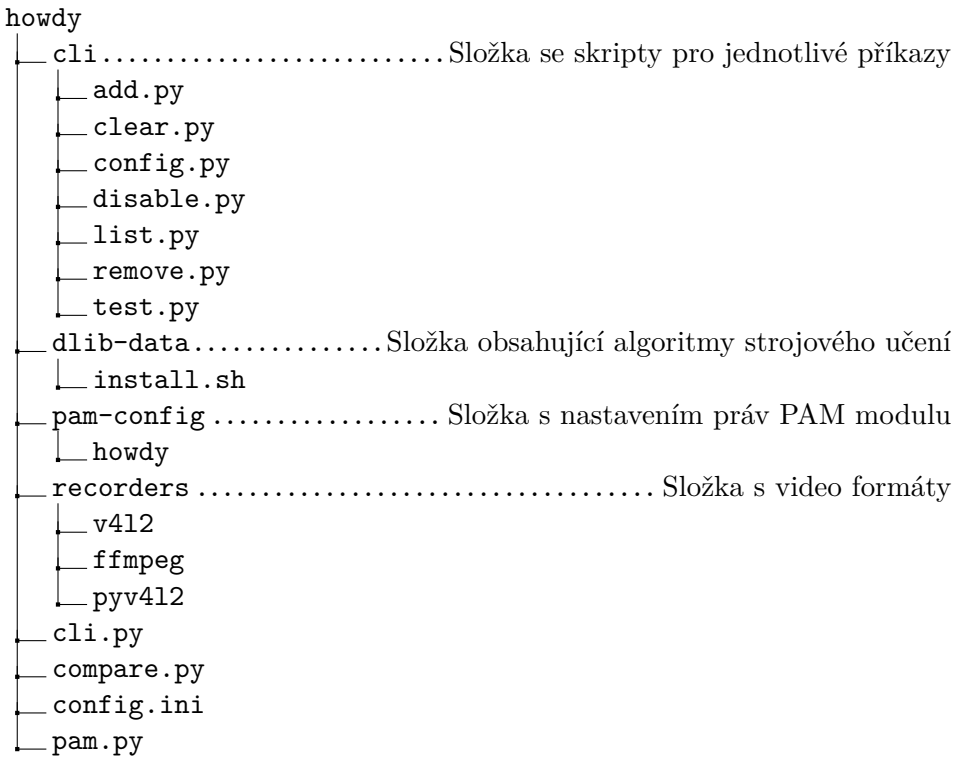
Projekt je k dispozici veřejně v github repozitáři, viz [6].

2.3.1 Struktura

Veškeré soubory, se kterými Howdy pracuje, jsou instalovány podle konvence PAM modulů do adresáře */lib/security/*. Strukturu složky *howdy* v tomto adresáři lze vidět na obrázku 2.2.

Skripty ve složce *cli* jsou příkazy, které se spouští pomocí Howdy. Složka *dlib-data* obsahuje Dlib modely, pomocí kterých je detekován a rozpoznáván obličej. Ve složce *pam–config* se nachází PAM konfigurační soubor, díky kterému je možné nainstalovat PAM modul. Poslední složkou je *recorders*. Zde se nachází video formáty, které se dále dají použít pro výstup z webkamery.

Kromě těchto složek jsou v kořenové složce *howdy* obsažené následující skripty:



Obrázek 2.2: Struktura složky howdy

cli.py Tento skript se stará o spouštění programu howdy. Ověřuje zda je spuštěný pod administrátorskými právy a analyzuje zadaný příkaz. Ten následně spustí.

compare.py Řeší porovnávání obličejů. Více v podsekci Porovnání obličejů.

config.ini Konfigurační soubor s flagy. Více v podsekci Konfigurace.

pam.py PAM modul napsaný v jazyce Python, spouští skript *compare.py*.

2.3.2 Rozpoznání a přidání modelu obličeje

Přidání obličeje zajišťuje skript *add.py*. Samotné rozpoznání obličeje umožnuje knihovna Dlib pomocí trénovaných rozpoznávacích modelů. Nejprve se nahrají hodnoty z konfiguračního souboru, jako například: cesta k webkameře, způsob konverze video formátů, výška a šířka snímku. Následně se získá snímek z webkamery, který je předán Dlib modelu *mmod_human_face_detector.dat*. Ten dokáže detekovat lidský obličej a vrátit jeho umístění. Pokud by na snímku bylo detekováno větší množství obličejů než jeden, program

skončí s chybou a upozorní uživatele. To samé v případě detekce nulového počtu obličejů. Pokud je tedy detekován pouze jeden obličej jsou díky snímku a získané pozice nalezeny hranice očí a spodní části nosu pomocí *shape_predictor_5_face_landmarks.dat*. Poslední fází je zakódování dat o obličeji Dlib modelem *dlib_face_recognition_resnet_model_v1.dat* a uložení do JSON souboru uživatele, který tento příkaz spustil.

2.3.3 Porovnávání obličejů

Porovnání obličejů probíhá podobným způsobem jako jejich přidávání. Slouží k tomu skript *compare.py*. Z výstupu webové kamery jsou rozpoznány všechny obličeje, nacházející se na snímku. Následně jsou zakódovány a porovnávány s modely obličejů uložených v souboru. Hledá se nejlépe vyhovující model obličeje s minimální přesností určené v konfiguračním souboru.

Úspěch, či neúspěch porovnání lze zjistit z návratové hodnoty skriptu:

- 10 – Nejsou přidané žádné modely.
- 11 – Byl překročen časový limit detekce obličeje.
- 12 – Detekce selhala.
- 0 – Úspěch, byl nalezen vyhovující model.

2.3.4 Konfigurace

Projekt Howdy umožňuje nastavit si jeho chování v konfiguračním souboru *config.ini*. Tento soubor se dělí na skupiny: core, video a debug.

Skupina core obsahuje flagy, kterými se nastavují výpisy do konzole, vypnutí a zapnutí Howdy, nebo technika při detekci obličeje. Všechny flagy v této skupině mohou obsahovat pouze hodnoty *true*, nebo *false*. Jsou to flagy:

detection_notice Pokud je nastavené *true*, je při ověřování obličeje do konzole vypsána hláška *Attempting face detection*.

no_confirmation Při nastavené hodnotě *false* je po úspěšném ověření obličeje vypsána hláška *Identified face as user*, kde *user* je jméno uživatele, který byl identifikován. V opačném případě se nic nevypisuje.

suppress_unknown Pokud uživatel nemá přidaný žádný model obličeje je při použití Howdy vypsána do konzole hláška *No face model known*. Při nastavení hodnoty *true* se výpis neprovádí.

ignore_ssh Umožňuje povolit, nebo zakázat ověřování Howdy při přihlašování přes protokol SSH. Hodnota *true* ověřování zakazuje a hodnota *false* povoluje.

ignore_closed_lid Umožňuje povolit, nebo zakázat ověřování skrz modul Howdy, pokud je víko notebooku zaklapnuté. Hodnota *true* ověřování zakazuje a hodnota *false* povoluje.

disabled Zakáže Howdy pro ověřování. Hodnota *true* zakazuje, hodnota *false* povoluje.

use_cnn Tímto flagem se nastavuje, zda se pro rozpoznávání obličeje použije technika CNN. Pokud je nastavené *false*, je použitá technika HOG.

Skupina video obsahuje flagy, kterými se nastavuje práce s videem při detekci obličeje a výstupem z webkamery. Jsou to:

certainty Jistota, že detekovaný obličej patří určitému uživateli. Může obsahovat hodnoty od jedné do deseti. Nejsou doporučovány hodnoty vyšší než pět.

timeout Časový limit, při kterém probíhá detekce obličeje. Pokud program poběží déle než definovaný limit, je ukončen.

device_path Cesta k webkameře, nebo k zařízení, ze kterého získáváme obraz.

max_height Omezuje maximální výšku videa na nastavenou hodnotu. Podle této hodnoty se škáluje získaný frame z webkamery.

frame_width Mění šířku videa. Největší šířka je nastavena při hodnotě mínus jedna.

frame_height Mění výšku videa. Největší výška je nastavena při hodnotě mínus jedna.

dark_threshold Pokud procento tmavých pixelů z celkového počtu pixelů je větší než tato hodnota, frame se přeskočí. Jelikož jde o procenta, může nabývat hodnot od nuly až po sto.

recording_plugin Určuje, který rekordér se má použít. Přípustné hodnoty jsou: opencv, ffmpeg a pyv4l2. V základu je nastavené opencv.

device_format Video formát, který se použije pro FFMPEG. Může obsahovat: vfwcap nebo v4l2.

force_mjpeg Může obsahovat hodnoty *true* nebo *false*. Hodnota *true* znamená, že se formát MJPEG použije při kódování videa.

exposure Nastavuje expozici videa. Hodnota mínus jedna se postará o automatické nastavení expozice.

Poslední skupinou je debug, která má pouze jeden nastavitelný flag a slouží pro odladění fungování programu Howdy.

end_report Přípustnými hodnotami jsou *true* a *false*. Nastavením hodnoty *true* získáme při provádění ověřování výpisy do konzole. Díky těmto výpisům vidíme například kolik tmavých framů bylo ignorováno, přesnost se kterou byl model odhadnut a čas potřebný pro ověření.

2.3.5 Přihlašování a odemykání

Přihlášování i odemykání je řešeno pomocí PAM modulu. Je využita knihovna Pam-python.

PAM modulem je soubor *pam.py*, který se stará o získání autentizace. Hlavními funkcemi tohoto modulu jsou:

- *pam_sm_authenticate()*
- *pam_sm_open_session()*
- *pam_sm_close_session()*
- *pam_sm_setcred()*

pam_sm_authenticate()

Funkce zajišťující získání práv při ověření uživatele. Autentizace je popsána v podsekci Autentizace.

pam_sm_open_session()

Funkce je volána, pokud začíná nová relace. K získání práv se postupuje stejně jako v předchozím případě, viz Autentizace.

pam_sm_close_session()

Tato funkce je volána při ukončení relace. Program Howdy nepotřebuje po sobě uklízet žádná data a je tedy pokaždé vrácena hodnota PAM_SUCCESS.

pam_sm_setcred()

Slouží pro získání credentials. Jelikož Howdy nepotřebuje žádné credentials získat, je vrácena hodnota PAM_SUCCESS.

2.3.6 Autentizace

Získání autentizace záleží na nastavených hodnotách v konfiguračním souboru a dále také na návratové hodnotě spouštěného skriptu *compare.py*.

Nejdříve je ověřen flag **disabled**. Pokud je nastavený na hodnotu *true*, je modul ukončen. Pokud není, pokračuje se ověřením flagu **ignore_ssh**. Ten, pokud obsahuje hodnotu *true* a zároveň ověřování probíhá přes protokol SSH, je opět ukončen. Jinak se pokračuje spuštěním skriptu *compare.py*, který vrací status jako návratovou hodnotu. Podle tohoto statusu se řídí získání autentizace. Pokud je hodnota rovná deseti, je vrácena hodnota PAM_USER_UKNOWN a autentizace není získána. Při hodnotách rovných číslům jedenáct, nebo dvanáct je vrácena hodnota PAM_AUTH_ERROR a autentizace opět neproběhne. Jediný status rovný nule značí úspěch a v tom případě je vrácena hodnota PAM_SUCCESS a autentizace je provedena.

Ve všech ostatních případech to znamená, že nastala neznámá chyba a tudíž je návratovou hodnotou PAM_SYSTEM_ERR a autentizace se opět nekoná.

2.4 Shrnutí

Na základě porovnání všech zmíněných projektů jsem se rozhodl použít v mé práci jako základ projekt Howdy. Nabízí oproti ostatním možnost nastavit si konfiguraci a práci s videem podle potřeb uživatele.

Také na něm stále pracuje několik lidí a jak jsem si všiml i v průběhu psaní mé práce, jsou vydávány nové verze, které opravují případné chyby, nebo vylepšují rychlost a funkčnost ověřování obličeje. V poslední době také vznikají balíčky pro distribuce jako například Fedora nebo Arch Linux.

Howdy je pod licencí MIT a je tedy možné ho v práci použít.

Kapitola 3

Analýza a návrh

V předchozí kapitole je popsána rešerše aktuálních projektů zabývajících se odemykáním pomocí obličeje v operačním systému GNU/Linux. Z důvodu nalezení řešení, které je dostatečně funkční a nabízí uživateli možnosti, jak si program přizpůsobit, nebo nastavit pomocí příkazů, bylo toto rešení zvoleno jako základ práce.

Neexistuje zatím projekt, který by se dal ovládat i pomocí grafického rozhraní. V prostředí KDE Plasma 5 se pro nastavení funkcionalit systému využívá systémových modulů (nebo také KCM), které jsou dostupné v systémovém nastavení. Cílem je tedy navrhnout systémový modul, jehož základ je nalezené řešení Howdy. Z tohoto důvodu celá analýza vychází z použitého projektu. Systémový modul bude muset obsahovat minimálně funkce, které nabízí projekt Howdy, zobrazí je a umožní uživateli s nimi lépe pracovat přes grafické prvky.

Analýza a návrh se zabývají čistě KCM modulem, který je důležitou součástí práce a umožní uživateli nastavení vybraného projektu. Důkladná analýza a návrh by měly být součástí vývoje každého softwaru a jsou velmi důležité pro pozdější implementaci. Analýza i návrh vycházejí z vlastních zkušeností a použitých principů v systémových modulech, které jsou již implementovány. V této kapitole je uveden přehled funkčních a nefunkčních požadavků, definice případů užití a návrh doménového modelu.

3.1 Definice požadavků

Jednotlivé požadavky výsledné aplikace jsou určeny na základě použitého projektu Howdy. Ten sám o sobě obsahuje několik přikazů: přidání a odebrání obličeje, vymazání všech modelů, nastavení, vypnutí a vylistování modelů. Podle nich se tedy řídí i funkční požadavky výsledného systémového modulu, který tyto příkazy bude volat a zobrazí je uživateli v přehlednější podobě, nebo mu umožní přenastavit si fungování programu. Každý uživatel má mož-

nost pracovat pouze se svými modely, z hlediska bezpečnosti není možné, aby upravoval modely jiných uživatelů.

3.1.1 Funkční požadavky

F1 – Přidání modelu obličeje

Uživatel si může přidat model obličeje, kterým se autorizuje při přihlašování do systému.

F2 – Odebrání modelu obličeje

Uživatel si může odebrat libovolný model, který je vylistovaný v tabulce aktuálních modelů.

F3 – Vymazání všech modelů

Odstraní celý seznam modelů pro aktuálně přihlášeného uživatele.

F4 – Zapnutí / vypnutí

Umožňuje povolit, nebo zakázat přihlášení do systému pomocí rozpoznání obličeje.

F5 – Nastavení

Uživatel má možnost nastavit si konfiguraci programu Howdy.

F6 – Vylistování modelů

Aplikace zobrazí seznam všech modelů pro aktuálně přihlášeného uživatele.

F7 – Uložení zadaných hodnot

Aplikace umožní uživateli uložení zadaných hodnot do konfiguračního souboru.

F8 – Resetování hodnot

Uživatel může vrátit zpět změny provedené v modulu před jejich uložením.

3.1.2 Nefunkční požadavky

N1 – Systémový modul

Aplikace bude implementovaná jako systémový modul pro desktopové prostředí KDE Plasma 5.

Tabulka 3.1: Tabulka pokrytí funkčních požadavků

	UC1	UC2	UC3	UC4	UC5	UC6	UC7
F1	X						
F2		X					
F3			X				
F4				X	X		
F5							X
F6						X	
F7				X	X		X
F8				X	X		X

N2 – Spolehlivost

Zobrazí vždy relevantní a aktuální data.

N3 – Napojení na projekt Howdy

Modul bude využívat data, která dostane z projektu Howdy, a umožní jeho ovládání.

N4 – Jazyk aplikace

Uživatelské rozhraní bude v anglickém jazyce.

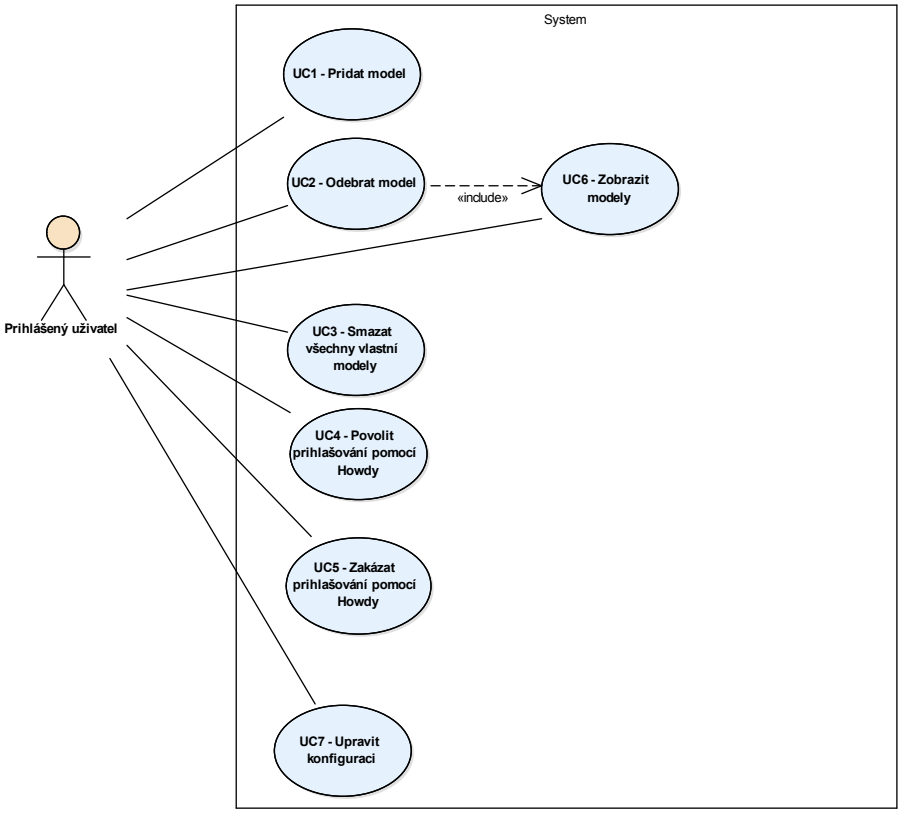
3.2 Případy užití

Případy užití slouží k detailní specifikaci funkčních požadavků. Jednotlivé požadavky se rozkládají na několik případů užití. Vycházel jsem tedy z funkčních požadavků. Všechny požadavky jsou pokryti, jak lze vidět v tabulce 3.1. Podrobnější popis případů užití je popsán níže a lze je vidět ve formě diagramu na obrázku 3.1.

UC1 - Přidání modelu

Umožňuje uživateli přidat nový model obličeje, díky kterému bude možné ho přihlásit do systému.

1. Případ užití začíná, jestliže se uživatel rozhodne přidat nový model obličeje.
2. Uživatel otevře systémový modul.
3. Uživatel zadá jméno modelu a potvrdí přidání.
4. Systém přidá a uloží nový model pod zadaným názvem.



Obrázek 3.1: Model případu užití

UC2 - Odebrání modelu

Umožňuje uživateli odebrat existující model obličeje.

1. Případ užití začíná, jestliže se uživatel rozhodne odebrat model ze seznamu.
2. Uživatel otevře systémový modul.
3. Uživatel si vybere jeden z modelů, který chce odstranit.
4. Systém odstraní vybraný model a aktualizuje data.

UC3 - Smazání všech vlastních modelů

Umožňuje uživateli vymazat veškeré modely, které si sám dříve přidal.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne vymazat všechny své modely.
2. Uživatel si otevře systémový modul.
3. Uživatel zvolí vymazání modelů.
4. Uživatel odsouhlasí vymazání modelů.
5. Systém odstraní všechny uživatelovy modely a aktualizuje data.

Alternativní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne vymazat všechny své modely.
2. Uživatel si otevře systémový modul.
3. Uživatel zvolí vymazání modelů.
4. Uživatel neodsouhlasí vymazání modelů.
5. Systém neprovádí žádné změny.

UC4 - Povolení přihlašování pomocí Howdy

Umožňuje uživateli zapnout funkci howdy pro autentizaci pomocí PAM.

Hlavní scénář

1. Případ užití začíná, pokud se uživatel rozhodne povolit Howdy, když je zrovna vypnuté.
2. Uživatel si otevře systémový modul.
3. Uživatel zvolí povolení Howdy.
4. Uživatel potvrdí provedení změn.
5. Modul povolí přihlašování pomocí Howdy v konfiguračním souboru.

Alternativní scénář

1. Případ užití začíná, pokud se uživatel rozhodne povolit Howdy, když je zrovna vypnuté.
2. Uživatel si otevře systémový modul.
3. Uživatel zvolí povolení Howdy.
4. Uživatel nepotvrdí provedení změn.
5. Modul neprovádí žádné změny v konfiguračním souboru Howdy.

UC5 - Zakázání přihlašování pomocí Howdy

Umožňuje uživateli vypnout funkci howdy pro autentizaci pomocí PAM.

Hlavní scénář

1. Případ užití začíná, pokud se uživatel rozhodne zakázat Howdy, v případě, že je zapnuté.
2. Uživatel si otevře systémový modul.
3. Uživatel zvolí zakázání Howdy.
4. Uživatel potvrdí provedení změn.
5. Modul zakáže přihlašování pomocí Howdy v konfiguračním souboru.

Alternativní scénář

1. Případ užití začíná, pokud se uživatel rozhodne zakázat Howdy, v případě, že je zapnuté.
2. Uživatel si otevře systémový modul.
3. Uživatel zvolí zakázání Howdy.
4. Uživatel nepotvrdí provedení změn.
5. Modul neprovádí žádné změny v konfiguračním souboru Howdy.

UC6 - Zobrazení modelů

Umožňuje uživateli zobrazit si modely, které si přidal pro autentizaci.

1. Případ užití začíná, pokud si uživatel otevře systémový modul.
2. Modul načtě data z aplikace Howdy a zobrazí je uživateli.

UC7 - Úprava konfigurace

Umožňuje uživateli nastavit si chování programu Howdy.

Hlavní scénář

1. Případ užití začíná, pokud se uživatel rozhodne změnit některé chování programu Howdy.
2. Uživatel si otevře systémový modul.
3. Modul zobrazí uživateli aktuální konfiguraci.

4. Uživatel přenastaví hodnotu v konfiguraci.
5. Uživatel potvrdí provedení změn.
6. Modul uloží vzniklé změny do konfiguračního souboru Howdy.

Alternativní scénář

1. Případ užití začíná, pokud se uživatel rozhodne změnit některé chování programu Howdy.
2. Uživatel si otevře systémový modul.
3. Modul zobrazí uživateli aktuální konfiguraci.
4. Uživatel přenastaví hodnotu v konfiguraci.
5. Uživatel nepotvrdí provedení změn.
6. Modul neprovede žádné změny v konfiguračním souboru Howdy.

3.3 Doménový model

Doménový model (viz obrázek 3.2) v této aplikaci je velmi jednoduchý. Při tvorbě jsem vycházel z definovaných požadavků a případů užití. Veškerá data bude aplikace získávat z použitého projektu Howdy. Jedná se o následující entity:

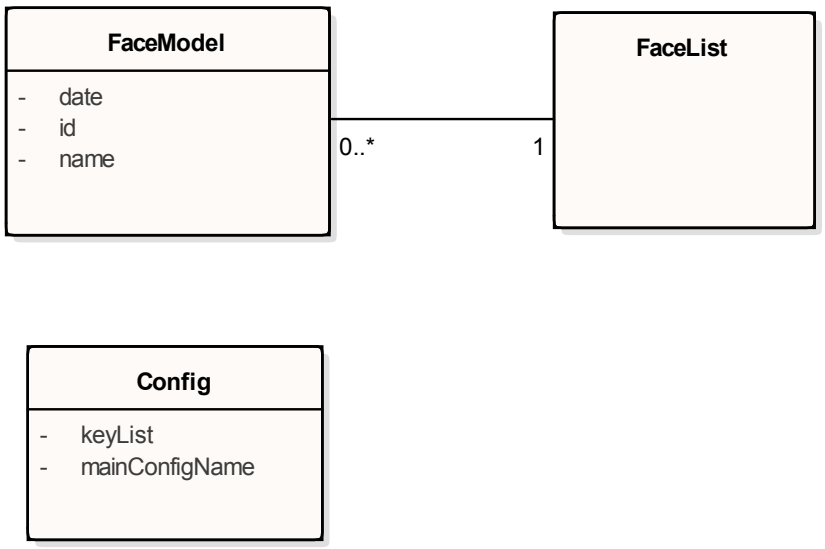
- FaceModel
- FaceList
- Config

Podrobněji jsou rozepsané níže.

3.3.1 FaceModel

Tato entita reprezentuje model obličeje, který si uživatel přidal. Aplikace bude zobrazovat tyto modely v tabulce, proto bude nutné si je udržovat. Při získávání výpisu modelů příkazem **sudo howdy list** dostáváme na jednotlivých řádcích hodnoty, podle kterých jsem určil budoucí atributy. Jsou to:

- id
- date
- name



Obrázek 3.2: Doménový model

3.3.2 FaceList

Entita představuje seznam modelů obličeje. Není nutné si udržovat další atributy.

3.3.3 Config

Entita config představuje konfigurační soubor pro projekt Howdy. Aplikace bude umožňovat nastavit si hodnoty v tomto souboru podle vlastních potřeb. Entita si tedy musí udržovat jméno souboru a seznam klíču (flagů). Představují je následující atributy:

- keyList
- mainConfigName

3.4 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní jsem provedl pomocí jednoduchých wireframů. Wireframy slouží programátorovi k určení budoucí funkcionality. Při návrhu jsem vycházel z předchozích stanovených funkčních požadavků. Výsledný modul by měl být přehledný a pro uživatele intuitivní. Rozdělil jsem si tedy wireframy na tři záložky. Záložky jsem zvolil z důvodu oddělení jednotlivých

Tabulka 3.2: Tabulka pokrytí případu užití

	WF Model	WF Add	WF Config
UC1		X	
UC2	X		
UC3	X		
UC4	X		
UC5	X		
UC6	X		
UC7			X

funkcí. Pro větší přehlednost jsem od sebe oddělil nastavení konfigurace, zobrazování modelů a přidávání modelů. Záložky jsou rozdělené na: Model, Add a Config. Každá má svůj vlastní wireframe. V tabulce 3.2 lze vidět pokrytí případů užití jednotlivými wireframy.

3.4.1 Model

Zobrazuje jednotlivé modely obličeje v tabulce, která umožňuje jejich odebrání. Pod tabulkou se nachází tlačítko Clear all a tlačítko pro povolení, nebo zakázání Howdy. Tlačítko Clear all odstraňuje všechny záznamy v tabulce. Návrh lze vidět na obrázku 3.3.

3.4.2 Add

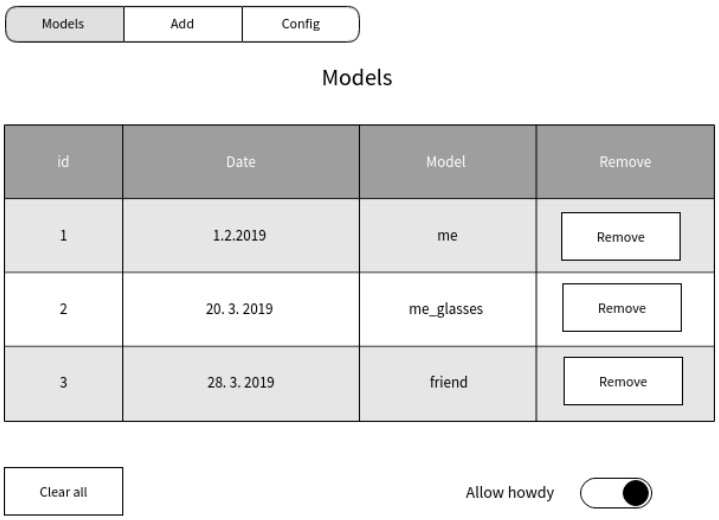
Jednoduchá záložka pro přidání nového modelu. V její horní části obsahuje textové pole, kam uživatel zadá jméno nově přidávaného modelu a stiskne tlačítko Add. Po stisknutí se spustí projekt Howdy a přidá nový model obličeje. Lze vidět na obrázku 3.4.

3.4.3 Config

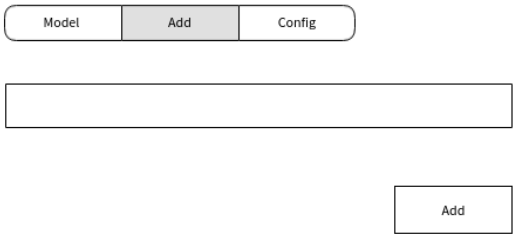
Umožňuje nastavit si funkčnost aplikace Howdy. Tento wireframe vychází ze souboru */lib/security/howdy/config.ini*. Obsahuje textově nadefinované proměnné s hodnotami, ze kterých Howdy vychází. Při návrhu jsem použil název, vedle kterého je prvek umožňující vybrat si z nabízených hodnot (ve většině případů true/false), viz 3.5.

3.5 Diagram tříd

Při vytváření diagramu tříd (viz obrázek 3.6) mi pomohly výchozí systémové moduly v prostředí KDE Plasma 5. Návrh také vychází z obecnějšího doménového modelu. Některé třídy souvisí s frameworkem KDE a Qt.



Obrázek 3.3: Wireframe Model



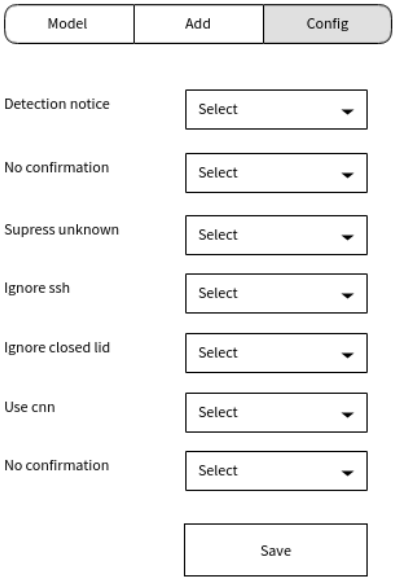
Obrázek 3.4: Wireframe Add

Třídy `FaceMode` a `FaceModelsList` jsou třídy doménového modelu, kde navíc třída `FaceModelsList` dědí od třídy `QAbstractTableModel` definované ve frameworku Qt. Třídy jako `ModelWidget`, `AddWidget` a `ConfigWidget` jsou opět třídy závislé na frameworku Qt, protože dědí od třídy `QWidget`, která je základem všech objektů pracujících s uživatelským rozhraním. Důležité je také zmínit třídu `KCModule`, která patří do frameworku KDE. Je to základní třída pro konfigurační moduly. Pro lepší přehlednost a pochopení základních funkcí tříd uvádím stručný popis:

KcmHowdy Třída, která je samotným systémovým modulem.

ModelWidget Třída, která zobrazuje tabulku s modely a obsahuje tlačítka pro vymazání a vypnutí funkce Howdy.

FaceModelsList Tato třída získává data ze souboru a zajišťuje zobrazení dat v tabulce třídy `ModelWidget`.

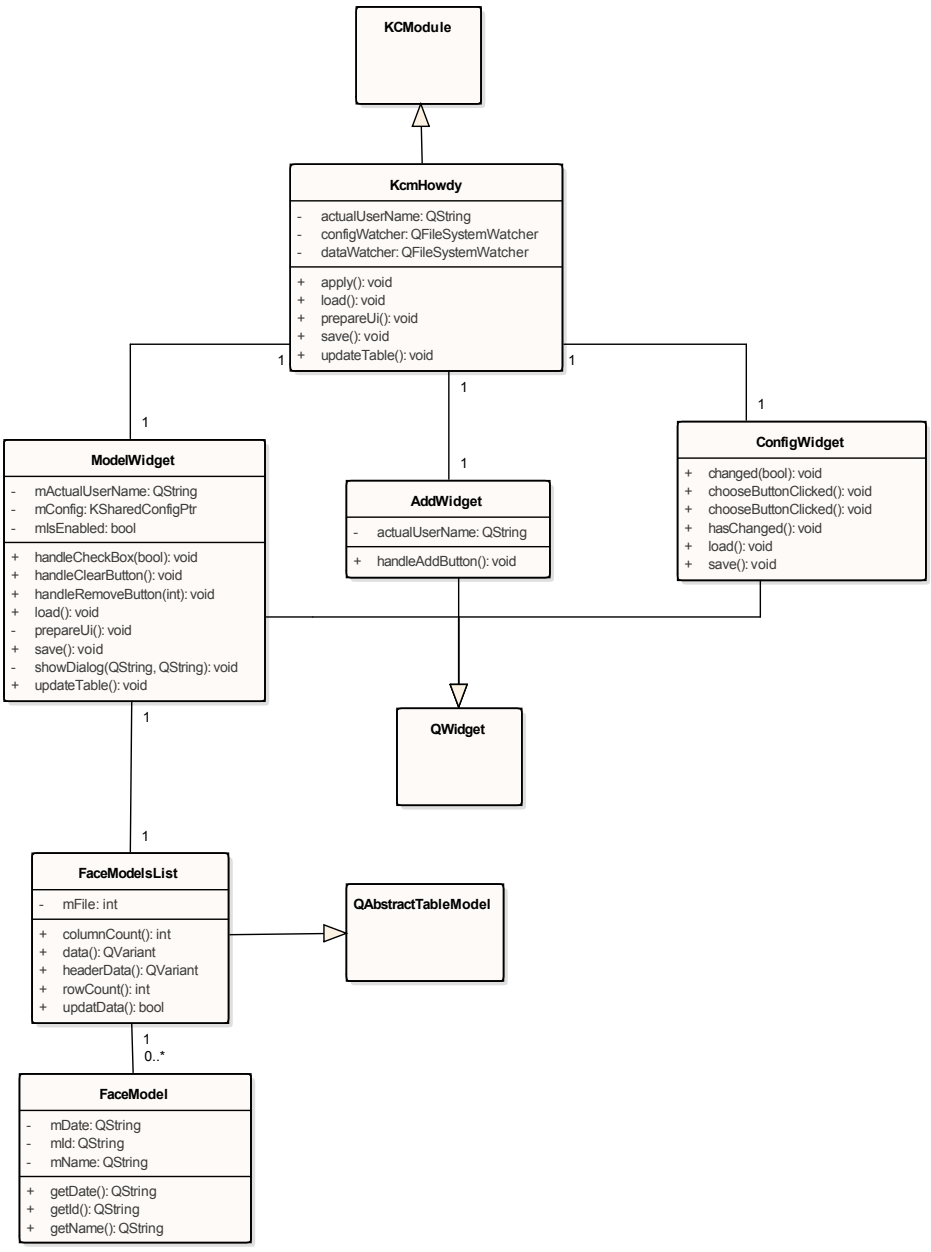


Obrázek 3.5: Wireframe Config

FaceModel Tato třída zastupuje model obličeje.

AddWidget Díky této třídě je možné nahrát nový model obličeje.

ConfigWidget Umožňuje změnit data v konfiguračním souboru.



Obrázek 3.6: Diagram tříd

Kapitola 4

Realizace

Vzhledem k tomu, že jsem objevil použitelný projekt Howdy, bylo potřeba vytvořit systémový modul pro prostředí KDE plasma 5. Pro implementaci jsem zvolil jazyk C++ a frameworky KDE, které staví na frameworku Qt.

Qt je framework napsaný v jazyce C++ a je rozšířený o funkce, jako jsou signály a sloty. Signály a sloty slouží pro komunikaci mezi objekty. Pokud změníme jednu komponentu, chceme, aby upozornila další, že nastala nějaká změna. Pokud nastane nějaká změna, objekt pošle signál dalšímu, který je na něm závislý, a ten vykoná funkci, která na tento signál reaguje. To lze vidět na obrázku 4.1, který jsem čerpal ze stránky dokumentace Qt frameworku [7].

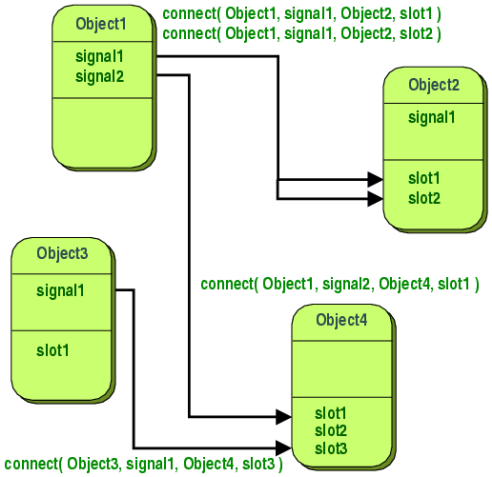
Co se týče grafického rozhraní, využil jsem prvků z frameworku Qt. Design aplikace pro jednotlivé třídy dědící od `QWidget` jsem vytvořil v programu *QtCreator* v souborech s příponou .ui. Tyto soubory používají formát XML pro reprezentaci elementů a jejich charakteristiku.

Jako vývojové prostředí jsem využíval *Kubuntu* ve verzi 18.10 s grafickým prostředím KDE Plasma s verzí 5.13.5. Implementaci jsem prováděl převážně v již zmiňovaném programu *Qt Creator*. Pro kompilaci kódu jsem zvolil program *CMake*.

4.1 Grafické rozhraní

Při tvorbě grafického rozhraní jsem vycházel převážně z navržených wireframů. Nejsem dlouhodobý uživatel prostředí KDE plasma 5 a mou inspirací byly systémové moduly, které jsou již v tomto prostředí implementovány. Zvolil jsem využití záložek pro oddělení jednotlivých funkcí projektu Howdy. Uživatel tak není rozptylován mnoha prvky najednou a může se přepnout do záložky funkce, kterou chce právě použít.

Záložky ve frameworku Qt jsou zajištěny třídou `QTabWidget` umožňující přepínaní mezi jednotlivými stránkami. Aplikace má celkem tři záložky, z nichž každá je samostatná třída `QWidget`. Detailněji jsou popsané níže.



Obrázek 4.1: Qt signály a sloty

4.1.1 Záložka Model

První z nich je nazvaná Model. Rozložení je navržené tak, aby se přizpůsobovalo aktuálně otevřenému oknu. Jsou zde prvky jako například `QCheckBox`, `QPushButton`, `SpacerItem` a `QTableView`.

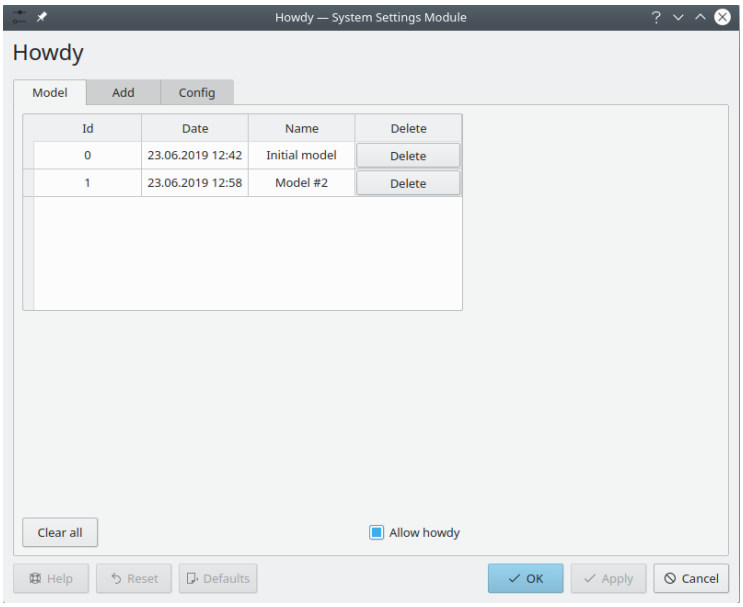
`QTableView` je hlavním prvkem. Zobrazuje tabulku uživatelových modelů. Tabulka je oddělená od ostatních prvků na této stránce pomocí tříd `QSpacerItem`, díky kterým lze dosáhnout prázdného místa při rozložení stránky. Projekt Howdy ve své specifikaci udává, že více než tři modely zpomalují algoritmus rozpoznávání. Z toho důvodu jsem nevolil tabulku moc velkou, i když je možné si přidat více modelů.

Tabulka je rozdělená na čtyři sloupce: id, datum, jméno a delete. Sloupce jsou vyplněné podle atributů každého modelu, kromě sloupce delete, ve kterém jsou tlačítka pro odebrání příslušných modelů.

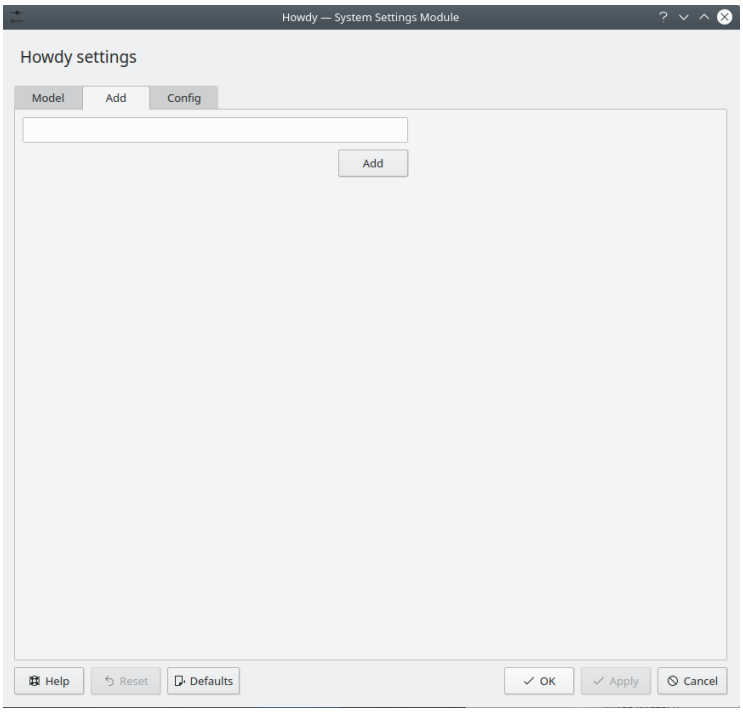
Pod tabulkou jsou dva prvky: `QPushButton` a `QCheckBox`. `QPushButton` s názvem **Clear all** slouží k vymazání celé tabulky a `QCheckBox` má funkci povolení, nebo zakázání spouštění Howdy při autentizaci uživatele. Výsledný vzhled lze vidět na obrázku 4.2.

4.1.2 Záložka Add

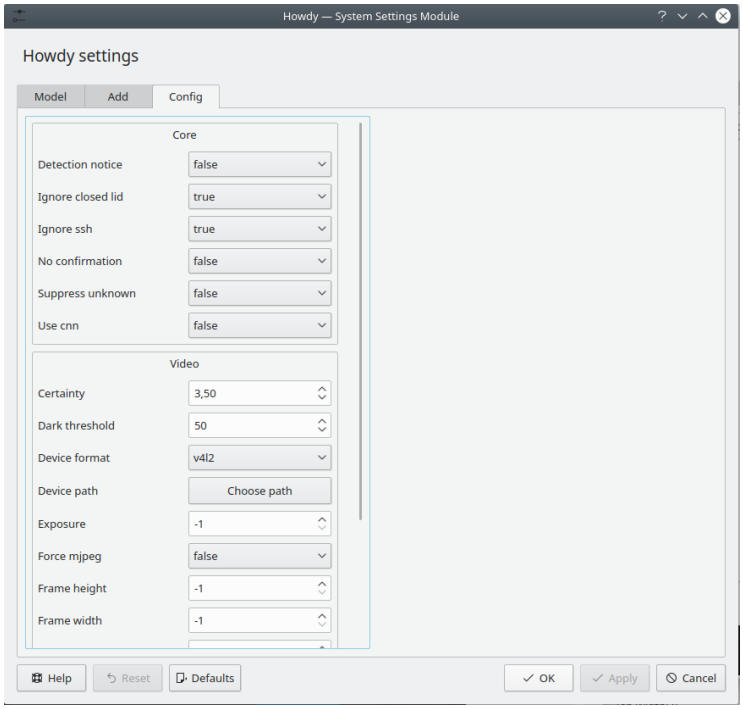
Jedná se o nejjednodušší stránku ze všech zmíněných. Obsahuje pouze textové pole `QTextEdit` a tlačítko `QPushButton`. Na této stránce se přidávají nové modely. Uživatel zadá jméno modelu do textového pole a po stisknutí tlačítka **Add** je model přidán, nebo se objeví chybová hláška. Výsledný vzhled lze vidět na obrázku 4.3.



Obrázek 4.2: Výsledná aplikace záložka Model



Obrázek 4.3: Výsledná aplikace záložka Add



Obrázek 4.4: Výsledná aplikace záložka Config

4.1.3 Záložka Config

Hlavním prvkem této záložky je `QScrollArea`, ve které se nachází jednotlivé skupiny vytvořené pomocí `QGroupBox`. Použít skupiny mě napadlo díky konfiguračnímu souboru, kde jsou jednotlivé části rozděleny na: core, video, debug. Zasadil jsem je tedy stejným způsobem do této aplikace. Každá skupina má pod sebou řádky, na kterých je vždy prvek `QLabel` a k němu příslušné pole, ve kterém uživatel vybere hodnotu.

Povolené hodnoty u jednotlivých prvků vychází ze sekce 2.3.4 Konfigurace a jsou nastavené tak, aby uživatel nemohl zadat chybnou hodnotu. Výsledný vzhled lze vidět na obrázku 4.4.

4.2 Napojení na projekt Howdy

Program Howdy je samostatně fungující řešení. Mým cílem bylo vytvořit pro něj systémový modul, ze kterého by bylo možné tento program ovládat, nebo si nastavit jeho fungování. Největší problém mi dělalo to, že Howdy nelze spouštět bez administrátorských práv.

Prvním řešením bylo volání příkazů pomocí třídy `QProcess`, umožňující spouštění externích programů a komunikaci s nimi. Této třídě jsem předával jako řetězec potřebný příkaz. Aby se spustil pod administrátorskými právy,

musel jsem před něj napsat **pkexec sudo**. Tím se po spuštění uživateli zobrazilo okno pro zadání administrátorského hesla a systém se postaral o provedení autentizace. Toto řešení není ovšem ideální. Na některých místech programu je třeba spouštět několik těchto příkazů za sebou a výsledkem pak dostáváme velké množství potvrzovacích oken, které chtějí zadat administrátorské heslo. Pro uživatele to může být časem zdržující.

Příkaz **pkexec**, je službou komponenty Polkit (nebo také PolicyKit). Komponenta Polkit poskytuje autorizační API a kontroluje ověření práv v Unixových operačních systémech. Ověřování přes Polkit se ukázalo jako dobrá cesta, ale pouze jiným způsobem. Ve frameworku KDE lze použít knihovnu `KAuth`. Tato knihovna se stará o autentizaci bez toho, aby vývojář musel řešit, jakou autentizaci cílový systém používá. `KAuth` je také možné použít pro spuštění části kódu přes takzvané helpery, se kterými se dále komunikuje pomocí rour.

Je to nejvhodnější cesta, jak umožnit uživateli spustit určité akce pod různými právy. To vše se provádí přes již zmíněný Polkit. Pro správnou funkčnost je třeba vytvořit třídu `HowdyAuthHelper`. Třídě se říká helper. Při vytváření musíme dodržovat určité konvence, musí dědit od třídy `QObject` a všechny definované metody musí vracet objekt typu `ActionReply`. Zároveň by každá metoda měla být akcí a její práva a popis by měly být definovány v souboru s příponou .actions, v tomto případě kcm_howdy.actions.

Pro lepší představu je akce v souboru .actions vidět na obrázku 4.5. Jednotlivá pole jsou definovaná následovně, viz [8]:

Název. Identifikátor akce

Jméno akce. Jméno akce

Popis. Popis akce, co akce vykonává

Práva. Výchozí práva potřebná pro vykonání akce

- yes – akce se spustí bez vyžadované autentizace
- no – akce je vždy odmítnuta
- auth_self – akce je autorizována, pokud uživatel autorizuje sám sebe
- auth_admin – akce je autorizována, pokud se uživatel přihlásí jako systémový administrátor

Trvalost. Volitelné pole definuje, jak dlouho vydrží autentizace

- session – přetrvá do chvíle, než je uživatel odhlášen
- always – přetrvává neurčitě dlouho

Tímto řešením jsem nakonec omezil počet autentizací v modulu pouze na dvě. Jednou z akcí je uložení dat do konfiguračního souboru, která má práva

```
[org.kde.kcontrol.příklad.akce]
Name=Příklad akce
Description=Toto je příklad akce
Policy=auth_admin
Persistence=session
```

Obrázek 4.5: Příklad akce

zápisu nastavená pouze pro administrátora. Další akcí je spuštění příkazů vymazání modelu, přidání modelu a vymazání všech modelů. Spouštění příkazu používá dohromady jen jednu společnou akci, které se posílá jako řetězec – příkaz pro vykonání.

4.2.1 Získání modelů

Získání modelů jsem zpočátku řešil stejnou metodou jako již zmíněné spouštění příkazů, avšak není to nutné. Soubory obsahující údaje o jednotlivých modelech jsou zapsané způsobem JSON, navíc mají práva ke čtení. Strukturu uložení dat lze vidět na obrázku 4.6.

Framework Qt obsahuje třídu `QJsonDocument`, která umožňuje číst a zapisovat JSON dokumenty. Postup načtení dat probíhá následovně:

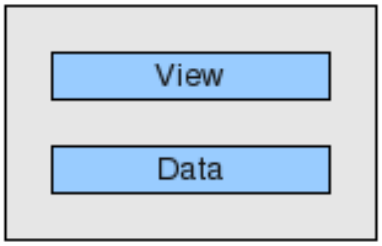
1. Otevření souboru s daty.
2. Přečtení souboru a uložení obsahu do proměnné.
3. Zavření souboru.
4. Získání JSON dokumentu pomocí `QJsonDocument` z proměnné.
5. Čtení dokumentu a získání potřebných dat.

O aktualizaci dat se stará `QFileSystemWatcher`. Tato třída dokáže zaznamenat změny v předaném souboru. Pokud třída detekuje, že nastala nějaká změna v souboru, vyšle signál, který odchytává implementovaný modul a odešle informaci, aby se data opět načetla.

4.3 Architektura

Ve frameworku Qt se většinou aplikace skládají ze zdrojových souborů a k nim příslušným UI souborů, pokud se jedná o UI soubory představují vzhled widgetu ve formátu XML. Program QtCreator nabízí nástroj [9] pro jednodušší tvorbu designu aplikace. Je na výběr velké množství prvků, které se dají pojmenovávat a vkládat.

Obrázek 4.6: Struktura JSON dat



Obrázek 4.7: Standardní komponenta

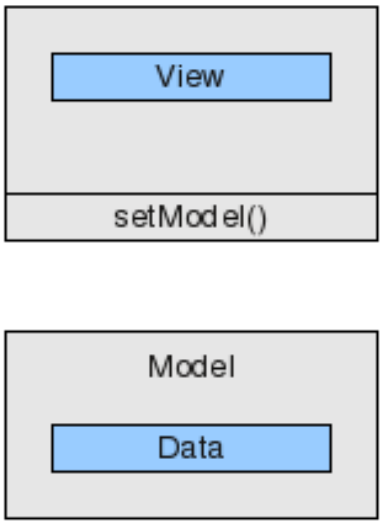
Qt nabízí dva typy grafických komponent, takzvaných Widgetů, viz [10]. Neliší se ve vzhledu, ale ve způsobu jak pracují s daty. Rozdělují se na standardní a na View třídy.

Standardní. Využívají data, která jsou jejich součástí, obrázek 4.7.

View. Operují nad externími daty, neboli modelem. 4.8

Ve své aplikaci používám oba přístupy. Model/View se hodí využít na tabulku, ve které uchovávám data modelů. Modelem je třída `FaceModelsList`, dědící od `QAbstractTableModel`. Model/View funguje obdobně jako MVC. Rozdíl je v tom, že zde není samostatný kontroler, ale pouze View, které oproti paternu MVC plní navíc funkci kontroleru. Třídy, které patří pod Model, jsou třídy zajišťující získání dat.

V ostatních případech je použitý přístup standardní, jelikož zde nemáme žádnou databázi. S daty se pracuje přes třídu `HowdyAuthHelper`, kterou jsem již zmiňoval.



Obrázek 4.8: Model/View

```
kcm_howdy.desktop ..... informace o aplikaci
kcm_howdy.actions ..... popis akcí a jejich práv
howdyauthhelper.cpp ..... Třída helperu
howdyauthhelper.h ..... Hlavičkový soubor helperu
src ..... zdrojové kódy
  ui ..... UI soubory
```

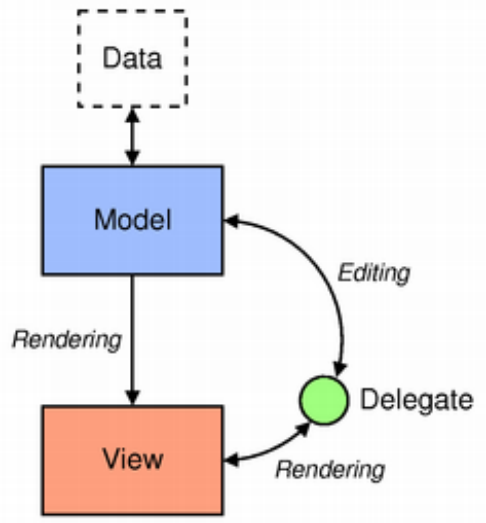
Obrázek 4.9: Struktura aplikace

4.4 Struktura aplikace

Jelikož jsem nikdy předtím žádný systémový modul pro prostředí KDE plasma 5 nevytvářel, inspiroval jsem se při tvorbě základní struktury moduly, které jsou již implementovány. Adresářová struktura aplikace obsahuje v první vrstvě soubory potřebné systémem KDE. Jsou to soubory kcm_howdy.desktop, kcm_howdy.actions a také hlavičkový soubory třídy `HowdyAuthHelper`.

Soubor kcm_howdy.destkop se vytváří proto, aby vzniklá aplikace byla viditelná pro uživatele v systémové nabídce. Tento soubor popisuje informace o aplikaci jako například: název, ikona, typ služby. Naopak soubor kcm_howdy.actions je soubor, popisující akce potřebné pro jednotlivé akce helperu. Je třeba si definovat název akcí, typ autentizace a popis.

Ostatní zdrojové soubory jsou umístěny ve složce src, ve které jsou oddělené UI soubory složkou ui. Lepší přehled je vidět na obrázku 4.9



Obrázek 4.10: Přístup Model/View použitý u QTableView

4.5 Zobrazení modelů

V této sekci čerpám ze zdroje [11] Aplikace Howdy používá pro zobrazení modelů příkaz **sudo howdy list**. Zobrazí které modely má uživatel přidané, datum přídání a jejich id. Tímto výstupem jsem se inspiroval při získávání modelů, jelikož z dat o obličeji se nedají vyčíst žádná užitečná data.

Rozhodl jsem se využít tabulku pro zobrazení dat a pro lepší přehlednost. Ve frameworku Qt jsou na výběr dva typy tabulek, lišící se způsobem získávání dat. Jsou to `QTableWidget` a `QTableView`. Použil jsem přístup druhý a to `QTableView`. Dávalo mi v tomto případě větší smysl použit přístup Model/-View a komunikovat s daty přes Model. Přístupy jsem popsal v části Architektura a lépe je lze vidět také na obrázku 4.10. Modelem v mé aplikaci je třída `FaceModelsList` dědící od třídy `QAbstractTableItem`, a má za úkol nahrát aktuální data ze souboru.

4.6 Přidání modelu

Přidání modelu probíhá ve třech krocích. Nejdříve si uživatel otevře druhou záložku a do textového pole zadá název nového modelu. Stiskne tlačítko add a tím se spustí akce. Při prvním provedení aplikace vyžaduje administrátorské heslo, protože se spouští příkaz **sudo howdy add**. Po správném zadání hesla jsou odeslána data jako jméno uživatele a název nového modelu třídě `HowdyAuthHelper`. Ta se postará o spuštění příkazu a nový model je přidán.

4.7 Odebrání modelu

Odebrání modelu probíhá obdobně jako jeho přidání. V tabulce je vybrán model k odebrání, u kterého je potřeba kliknout na tlačítko Delete. Každé tlačítko v tabulce se váže na určitý model podle jeho atributu id. Připojení signálu kliknutím se v tomto případě liší od ostatních. Signál clicked(), který je vyslaný po kliknutí tlačítka, nepřijímá žádný parametr. Parametr je potřeba poslat funkci, aby bylo jasné, který model se bude mazat. To se dá vyřešit pomocí třídy `QSignalMapper`, která ze signálů bez parametru vytvoří signál s parametrem a znovu ho vyšle. Tím se mi podařilo předat metodě handleRemoveButton parametr id. Samotné odebrání modelu probíhá opět přes třídu `HowdyAuthHelper`.

4.8 Nastavení konfigurace

Nastavení konfigurace vychází z konfiguračního souboru popsaném v sekci 2.3.4 Konfigurace. Úpravu hodnot v konfiguračním souboru řeší třída `ConfigWidget`, která reaguje na změny provedené uživatelem. Například změna hodnoty **certainty**. Pokud nastane změna v příslušném prvku, je odchytnuta třídou `ConfigWidget`. Ta vyšle signál třídě `KcmHowdy`, která ho odchytává a zjistí, že nastala změna. Pokud změna nastala je možné kliknout na tlačítko **Apply**, kterým se provedené změny uloží do konfiguračního souboru.

Při nahrávání a ukládání hodnot není použitý žádný algoritmus. Hlavním důvodem je, že by nebylo možné přidávat grafické prvky a zjistit jaké hodnoty mohou obsahovat. U každého prvku je tedy nahrávána hodnota z konfiguračního souboru pokaždé, když se soubor změní, nebo při spuštění aplikace.

4.9 Úpravy Howdy

Howdy poskytuje instalační balíčky pro distribuce ArchLinux, Debian a Fedora. V budoucnu se určitě rozšíří i na další distribuce. Implementaci a testování jsem prováděl převážně pro Debian distribuce. V prostředí Gnome nebylo třeba měnit PAM modul, ale v prostředí KDE jsem se setkal s několika problémy. Přihlašování přes sudo v příkazové řádce fungovalo bez problému, avšak na zamykací obrazovce a při prvotním přihlášení jsem se setkal s chybami, které bylo třeba opravit.

Na zamykací obrazovce je nutné pro spuštění stisknout klávesu enter s prázdným heslem, taktéž i na přihlašovací obrazovce. Spustí se Howdy, ale zobrazí nespecifikovanou chybu Unknown error: 1. Není nikde blíže popsané, co tato chyba znamená. Jako řešení jsem upravil řádek v PAM konfiguračním souboru common-auth. Místo [success=2 default=ignore] jsem použil sufficient. Sufficient znamená, že pokud modul dopadne úspěšně, už není vyžadovaná žádná další autentizace a moduly, které by měli nastavené required se

nespustí. Pokud by selhal, spustí se další moduly v pořadí. Důležité je, aby se nacházel před všemi required moduly. Pokud by byl required před sufficient a neuspěl by, autentizace pomocí Howdy by neproběhla a celý PAM modul by dopadl chybně. Správné nastavení PAM konfiguračního souboru s moduly vypadá následovně:

```
auth sufficient pam_python.so /lib/security/howdy/pam.py
auth required   pam_env.so
auth required   pam_unix.so   try_first_pass
```

Zprovoznily se tím přihlašovací i zamykací obrazovky, ve kterých stačí stisknout klávesu enter bez hesla. Spustí se Howdy a pokud uživatele ověří, pustí ho bez problému do systému. V prvním kroku je vždy ověřena autentizace pomocí Howdy. Pokud neproběhne je ověřené heslo.

Další chyba byla při přidávání modelů. Modelům bylo zadané id podle počtu přidaných modelů. Id začínalo nulovou hodnotou. Avšak pokud byl vymazán první model s nulovým id a přidán další, byly mezi datami dva modely se stejným id. Toto chování bylo potřeba opravit v souboru *add.py*. Šlo o jednoduchou změnu id tak, aby nový model měl vyšší id než model předchozí.

4.10 Bezpečnost

Z hlediska bezpečnosti aplikace jako taková neporušuje žádná pravidla. Akce, které vyžadují jiná než uživatelská práva, jsou spouštěna pomocí nástroje Polkit a není třeba se dále starat o fungování a autorizaci. Je ale třeba se podívat jak funguje projekt Howdy.

Hlavním záměrem je co nejvíce se přiblížit funkci Windows Hello, využívající infračervené kamery při získávání dat o obličeji. Testováním programu Howdy se nepodařilo infračervenou kameru zprovoznit a ve všech případech autentizace proběhla neúspěšně. Z tohoto důvodu jsem projekt Howdy testoval pouze na obyčejné webkameře. Použití obyčejné webkamery rozhodně není ideální volba. V tomto případě jde spíše o rychlejší autentizaci než v případě zadávání hesla. Ale klasické ověření hesla je mnohem bezpečnější a na lepší úrovni. Na základě experimentu při použití obyčejné webkamery jsem zjistil, že se lze autentizovat pouze přiložením fotografie. Postupoval jsem tak, že jsem si přidal model svého obličeje pomocí Howdy. Mobilním telefonem jsem pořídil fotografii svého obličeje a přiložil jí před webkameru ve fázi autentizace. Howdy mě rozpoznal bez problému a přihlásil do systému. Každý by si měl tedy rozmyslet, zda mu rychlá autentizace stojí za méně zabezpečený systém.

Jestliže by se podařilo využít infračervené kamery tak, jak jí používá funkce Windows Hello, dalo by se mluvit o lepším zabezpečení. Infračervená kamera by měla zabránit útokům pomocí přikládání fotografií, viz článek [12]. I přesto se nedoporučuje používat Howdy jako samostatnou autentizační metodu, jak

také uvádí autor na svém githubu projektu [6]. Mohlo by se také stát, že se do systému nepujde přihlásit. Nastávají situace, ve kterých algoritmus obličej nerozpozná. Při jeho testování jsem zjistil, že je lehce ovlivnitelný světelnými podmínkami a prostředím, ve kterém je model obličeje pořízen. Jednou za čas je tedy nutné nahrát model nový.

PAM moduly se používají v operačním systému GNU/Linux jako univerzální autentizační systém. Hlavní myšlenkou je zajistit, aby programy byly nezávislé na autentizačních postupech. PAM má čtyři různé oblasti, z nichž je v tomto případě důležitá oblast autentizace uživatele (auth). Je zvykem, že vše, co není povoleno, tak je jednoduše zakázané. Tudíž zapomenuté zabezpečení některé služby neposkytne nikomu cestu do systému. Více se lze dočíst na stránce [13].

Výchozí lokací pro PAM moduly je adresář */lib/security*, ve kterém se nachází i složka *howdy* se všemi potřebnými soubory. Práva jsou nastavená pouze pro čtení, s výjímkou spustitelných souborů. Proto nikdo kromě administrátora nemůže změnit obsah těchto souborů

Kapitola 5

Instalační balíčky a závislosti

Tato kapitola se zabývá popisem instalačních balíčků pro jednotlivé distribuce, jejich vytvářením a popisem jejich závislostí. Instalační balíčky jsou vytvářené pro distribuce Linux Mint, Gentoo a OpenSUSE. Operační systém GNU/Linux využívá těchto balíčků pro jednodušší instalaci a odinstalaci softwaru. Každá distribuce však používá rozdílný přístup k jejich vytváření a jiné příkazy k jejich instalaci.

Balíčky mohou být buď binární, nebo zdrojové. Zdrojové balíčky poskytují všechny potřebné soubory pro kompilaci nebo jiné sestavení požadovaného softwaru [14]. Binární balíčky obsahují předem vytvořené instalační soubory a jsou pohodlnější a rychlejší pro instalaci. Linux si pamatuje nainstalované balíčky a umí je jednotlivě odstraňovat.

Jako součástí práce jsem vytvořil instalační balíčky pro implementovaný KCM modul a také pro použitý projekt Howdy, který je potřebný ke správnému chodu výsledného KCM modulu. Pro distribuce založené na Debianu je již balíček Howdy vytvořen samotným autorem. Nebylo tedy potřeba ho vytvářet opakovaně.

5.1 Instalace Howdy

Pro manuální instalaci je důležité provést několik kroků. Prvním krokem je stažení projektu a vytvoření adresáře */lib/security/howdy*. Do tohoto adresáře se překopíruje obsah složky *src*. Dalším krokem je zkopírování Bash-completion souboru. Jedná se o inteligentní doplňování příkazů, které je součástí Bash shellu. Bash-completion soubor pro Howdy se nachází ve složce *autocomplete*. Soubor *autocomplete/howdy* musí být zkopírován do adresáře s Bash-completions, jedná se většinou o adresář */usr/share/bash-completion/completions*. Dále je nutné nastavit symbolický odkaz na skript */lib/security/howdy/cli.py* do adresáře, kde se nachází spustitelné soubory, například */usr/bin*. Symbolický odkaz bude mít název howdy. Souboru */lib/-*

security/howdy/cli.py musí být nastavena práva pro spuštění. Po těchto krocích lze spouštět Howdy z příkazové řádky a přidávat, nebo odebírat modely. Na závěr je třeba vykonat několik kroků, aby bylo možné Howdy používat při přihlášení. Jedná se o nastavení PAM modulu a přidání řádku s textem do správného souboru, většinou */etc/pam.d/common-auth*. Přidávaný řádek vypadá následovně:

```
auth sufficient pam_python.so /lib/security/howdy/pam.py
```

Je také nutné upravit v konfiguračním souboru */lib/security/howdy/config.ini* cestu k webkameře, vyřešit závislosti a nainstalovat potřebné knihovny.

5.2 Závislosti

Tato sekce popisuje jednotlivé závislosti implementovaného KCM modulu a použitého řešení Howdy, které jsou potřebné k jejich spuštění, nebo sestavení.

5.2.1 KCM modul

Implementovaný KCM modul závisí na několika knihovnách z KDE frameworku a na knihovně Qt. Pro instalaci je vyžadován framework KDE ve verzi minimálně 5.50.0 s balíčky `KAuth` a `KConfigWidgets`. Dále knihovna Qt s minimální verzí 5.11.0 s balíčkem `Widgets` a v neposlední řadě balíček ECM ve verzi minimálně 5.50.0 pro software CMake. CMake musí mít minimálně verzi 3.12. Poslední závislostí nutnou ke spuštění modulu je projekt Howdy ve verzi 2.5.1.

Všechny verze jsem určil na základě distribuce Kubuntu 18.10, ve které jsem KCM modul implementoval a podle verzí, se kterými jsem pracoval.

5.2.2 Howdy

Program Howdy závisí také na několika knihovnách. Závislosti jsem určil na základě autorem vytvořeného instalačního balíčku pro distribuci Debian. Spuštění programu Howdy závisí na knihovnách python3, dlib, pam-python, python3-numpy a python3-opencv. Na jiné závislosti jsem při spouštění programu nenarazil.

5.3 Linux Mint

Verze distribuce Linux Mint jsou založené na distribuci Ubuntu, případně Debianu. Jelikož vychází z Debianu používá se formát DEB. DEB má lépe vyřešený systém závislostí než formát RPM, který se používá například u distribucí OpenSUSE, Fedora a Red Hat Enterprise Linux. Program **apt-get** dokáže závislosti sám vyřešit [15].

```
Source: debian
Section: unknown
Priority: optional
Maintainer: First Name <email@debian.org>
Build-Depends: debhelper (>=10)
Standards-Version: 4.0.0
Homepage: <insert the upstream URL, if relevant>

Package: debian
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: <insert up to 60 chars description>
<insert long description, indented with spaces>
```

Obrázek 5.1: Příklad souboru control

5.3.1 Struktura Debian balíčků

Aby bylo možné pochopit vytváření instalačních balíčků pro distribuce Debian, je nutné zmínit soubory, se kterými se pracuje. Jedná se o následující soubory:

debian/control

Textový soubor *debian/control* obsahuje základní informace o balíčku. Informace jsou rozdělené na několik hodnot, které jsou potřebné pro nástroje spravující balíčky. Těmito nástroji mohou být například: **dpkg**, **apt-get** a **dselect**. Příklad souboru lze vidět na obrázku 5.1.

První část popisuje zdrojový balíček a obsahuje položky:

Source Určuje jméno zdrojového balíčku.

Jedná se o povinnou položku.

Section Určuje, jaké oblasti se balíček týká.

Jedná se o doporučovanou položku.

Priority Určuje důležitost balíčku. Může obsahovat hodnoty:

- required – nezbytný pro řádné fungování systému.
- important – měl by být nalezen ve všech Unixových systémech.
- standard – standardní pro jakýkoliv Linuxový systém.
- optional – volitelný balíček.
- extra – je užitečný pouze tehdy, pokud uživatel ví, co balíček obsahuje.

Jedná se o doporučovanou položku.

Maintainer Uvádí, kdo balíček vytvořil. Jedná se o povinnou položku.

Build-Depends Určuje závislosti balíčku, tedy, které balíčky musí být přítomny při kompilaci.

Standards-Version Určuje verzi Debian Policy, viz [16].

Jedná se o doporučovanou položku.

Vcs-Git Určuje, kde správce balíčku udržuje svoji pracovní verzi.

Tato položka je nepovinná.

Homepage Určuje domovskou stránku projektu.

Položka je nepovinná.

Další části popisují jednotlivé binární balíčky. Pro každý binární balíček bude v souboru *control* právě jedna tato sekce. Obsahuje položky:

Package Jméno balíčku.

Jedná se o povinnou položku.

Architecture Architektura binárního balíčku.

Jedná se o povinnou položku.

Depends Touto položkou jsou definované závislosti. Lze použít nástroj **debhelper**, který je schopný některé závislosti najít automaticky.

Položka není povinná, ale bez závislostí se balíčky většinou neobejdou.

Description Popis balíčku. První řádek určuje krátký popisek, další odsazené řádky dlouhý popisek.

debian/compat

Povinný soubor, který určuje verzi nástroje **debhelper**, se kterým byl balíček otestován.

debian/copyright

Povinný soubor s informacemi o licenci balíčku.

debian/docs

Nepovinný soubor obsahující seznam dokumentace, kterou má nástroj *debhelper* nainstalovat.

debian/changelog

Povinný soubor, který bsahuje seznam změn mezi revizemi balíčku.

debian/rules

Soubor obsahuje pravidla pro make, díky kterým je možné balíček zkompilovat.

5.3.2 Balíček pro KCM modul

Vytvořený KCM modul se instaluje pomocí programu **make**. Potřebný Makefile je vytvořen softwarem CMake. Ke kompilaci v instalačním balíčku jsem zvolil nástroj **debhelper**. Ten je použitý v suboru *debian/rules*, který provádí kompilaci.

Debhelper

Tomuto nástroji je možné předat množství parametrů, podle kterých se řídí kompilace samotného programu, více viz [17]. Použil jsem tyto parametry:

with Přidá příkazy zadaným doplňkem.

buildsystem Vynutí použití zadaného sestavovacího systému.

without Deaktivuje daný doplněk.

builddirectory Určuje jméno složky, ve které se provede sestavení aplikace.

Výsledný příkaz vypadá následovně:

```
dh $@ --with kf5,pkgkde_symbolshelper --buildsystem kf5
      --without build-stamp --builddirectory=build
```

Znamená to, že je použité prostředí a systém KF5, sestavení aplikace se provádí do složky build a je deaktivován doplněk build-stamp.

Debian závislosti

Sestavení modulu KCM závisí na Debian balíčcích, které poskytují potřebné knihovny popsané v sekci 5.2.1 KCM modul. Jedná se o následující balíčky:

cmake Poskytuje software CMake.

debhelper Poskytuje nástroj Debhelper.

extra-cmake-modules Instaluje ECM do softwaru CMake.

libkf5configwidgets-dev Jedná se o knihovnu KConfigWidgets.

libkf5auth-dev Knihovna KAuth.

qtbase5-dev Framework Qt.

```
Source: kcmhowdy
Section: kde
Priority: optional
Maintainer: petr <dusekpe2@fit.cvut.cz>
Build-Depends: cmake (>= 3.12),
               debhelper (>= 11~),
               extra-cmake-modules (>= 5.50.0~),
               libkf5configwidgets-dev (>= 5.50.0~),
               libkf5auth-dev (>= 5.50.0~),
               qtbase5-dev (>= 5.11.0~),
               pkg-kde-tools (>= 0.15.28)
Standards-Version: 4.1.3
Vcs-Git: https://gitlab.fit.cvut.cz/dusekpe2/kcmhowdy

Package: kcmhowdy
Architecture: any
Depends: howdy (=2.5.1), ${shlibs:Depends}, ${misc:Depends}
Description: Howdy management tool for KDE Plasma 5.
 A System Settings module for managing project Howdy.
 Howdy is face authentication module.
```

Obrázek 5.2: Soubor debian/control

pkg-kde-tools obsahuje KF5 systémové cesty a adresáře pro nástroj Debhelper.

Výsledný soubor *debian/control* KCM modulu lze vidět na obrázku 5.2.

5.4 OpenSUSE

Nativní správce balíčků distribuce OpenSUSE je modul pro správu softwaru YaST a správce softwaru Zypper, který se dá použít pouze z příkazové řádky [18]. Tito správci balíčků využívají RPM balíčků. Pro vytvoření RPM balíčku je nutný Spec soubor, který je čistě textový a definují se v něm položky jako například název, verze a další.

5.4.1 Spec soubor

Spec soubory se skládají z položek a částí, ve kterých jsou prováděny funkce jako například sestavení aplikace. Jednotlivé položky Spec souboru jsou popsané níže:

Name Jméno balíčku.

Version Verze balíčku.

Release Číslo vydání balíčku.

Summary Krátký popis, čeho se balíček týká.

License Licence balíčku.

Url Adresa, na které se nachází projekt.

Group Skupina, do které balíček spadá.

Source Zdrojové soubory balíčku. Má za účel dokumentovat, kde zdrojové soubory najít a určuje jméno zdrojového souboru.

BuildRequires Závislosti potřebné k sestavení balíčku.

Requires Závislosti potřebné ke spuštění výsledné aplikace.

%description Část, která obsahuje podrobnější popis, může být na více řádcích.

%prep Přípravná část, slouží k přípravě balíčku pro sestavení.

%build V této části se provádí samotné sestavení balíčku.

%install Část s instalací balíčku.

%files V této části jsou definované soubory, které spadají do balíčku.

5.4.2 Balíček pro KCM modul

Instalační balíček pro KCM modul jsem vytvářel opět na základě softwaru CMake. V části build jsem použil makro *cmake_kf5*. Tímto makrem se spustí Cmake se všemi potřebnými parametry pro framework KDE tak, aby došlo k instalaci do správných adresářů. Následně je použito makro *make_jobs*, které spustí příkaz **make**. Nakonec v části install je použité makro *kf5_makeinstall*, které spustí příkaz **make install**. V části files jsou vypsány soubory, které se vytvořily.

Výsledný balíček je dostupný v OpenSUSE repozitáři pro verzi OpenSUSE Tumbleweed a lze jej nainstalovat nástrojem Zypper přidáním repozitáře. Na obrázku 5.3 je vidět výsledný Spec soubor.

5.4.3 Balíček pro Howdy

Instalace Howdy se skládá z několika kroků, viz sekce 5.1 Instalace Howdy. V souboru spec jsem definoval závislosti potřebné ke spuštění a to knihovny python3, python3-dlib, python3-opencv, pam-python, python3-numpy. Část **%install** vychází z kroků instalace Howdy. V části **%post** jsou vypsané dodatečné úkony pro uživatele jako nastavení cesty k webkameře a přidání PAM konfigurace. Jako zdroj balíčku jsem si vytvořil vlastní archiv, ve kterém jsou opraveny chyby v Howdy a nainstalované Dlib modely potřebné k rozpoznání obličeje. Tyto modely jsou u balíčku pro distribuce Debian stahovány zároveň při instalaci. Toto řešení mi nepřišlo příliš vhodné, například při offline

```
Name:           kcmhowdy
Version:         1.0.0
Release:        0
Summary:        KCM module for prooject Howdy
License:        GPLv2
Group:          System/Libraries
Source0: https://github.com/dusekpe2/%{name}/%{version}.tar.gz
BuildRequires: cmake >= 3.12
BuildRequires: gcc-c++
BuildRequires: kf5-filesystem
BuildRequires: extra-cmake-modules >= 5.50.0
BuildRequires: libQt5Widgets-devel >= 5.11.0
BuildRequires: kconfigwidgets-devel >= 5.50.0
BuildRequires: libKF5Auth5 >= 5.50.0
Requires:           howdy = 2.5.1

%description
A System Settings module for managing project Howdy - face
authentication module.

%prep
%setup -q -n %{name}-%{version}

%build
%cmake_kf5 -d build
%make_jobs


%install
%kf5_makeinstall -C build


%files
%{_kf5_plugindir}/
%{_kf5_dbuspolicydir}/org.kde.kcontrol.kcmhowdy.conf
%{_kf5_sharedir}/polkit-1/actions/org.kde.kcontrol.kcmhowdy.policy
%{_kf5_sharedir}/dbus-1/system-services/
org.kde.kcontrol.kcmhowdy.service
%{_kf5_libdir}/libexec/
%{_kf5_servicesdir}/

%changelog
```

Obrázek 5.3: Spec soubor pro KCM modul

instalaci. Vlastní vytvořený archiv už tyto modely obsahuje a nemusí se tedy dodatečně stahovat.

Balíček je opět dostupný v OpenSUSE repozitáři pro verzi OpenSUSE Tumbleweed a je možné ho nainstalovat nástrojem Zypper. Výsledný Spec soubor je vidět na obrázku 5.4.

5.5 Gentoo

Distribuce Gentoo používá systém Portage. Portage je systém správy instalačních balíčků, který v Gentoo plní mnoho klíčových funkcí, například získání nejnovějšího softwaru, instalaci balíčků, nebo aktualizaci systému [19]. V každém balíčku musí být Ebuild soubor. Zatímco RPM nebo Deb šíří zkompilované binární soubory. Ebuild je čistě textový soubor obsahující instrukce, jak získat, nakonfigurovat a instalovat daný software.

```
Name:           howdy
Version:        2.5.1
Release:        0
Summary:        Windows Hello style authentication for Linux
License:        MIT
Group:          System/Libraries
URL:            https://github.com/dusekpe2/%{name}
Source0:        https://github.com/dusekpe2/%{name}/releases/
download/v%{version}/%{name}-%{version}.tar.gz

Requires:        python3
Requires:        python3-dlib
Requires:        python3-opencv
Requires:        pam-python
Requires:        python3-numpy

%description
Package for project Howdy, which is PAM authentication module for
face recognition

%prep
%setup -q

%install
mkdir -p %{buildroot}/lib/security/%{name}
cp -pr src/* %{buildroot}/lib/security/%{name}

mkdir -p %{buildroot}%{_datadir}/bash-completion/completions
install -Dm 644 autocomplete/%{name} %{buildroot}%{_datadir}/bash-
completion/completions

mkdir -p %{buildroot}%{_bindir}
chmod 0755 %{buildroot}/lib/security/%{name}/cli.py
ln -s /lib/security/howdy/cli.py %{buildroot}%{_bindir}/%{name}

%post
echo "Review settings in /usr/lib/security${D}/config.ini. Option
device_path must be set to an existing device."
echo "Add PAM configuration to /etc/pam.d, e.g.:common-auth"
echo "auth sufficient pam_python.so /usr/lib/secutiry${P}/pam.py"
echo "This package requires environment variable ``SUDO_USER'' set
even if ``sudo'' is not used."


%files
%license LICENSE
%doc README.md
%{_bindir}/%{name}
/lib/security
%{_datadir}/bash-completion/completions/%{name}

%changelog
```

Obrázek 5.4: Spec soubor pro Howdy

5.5.1 Ebuild položky

Soubory Ebuild obsahují několik základních položek. Uvádím zde jejich popis a význam:

EAPI Verze PMS, které soubor dodržuje.

DESCRIPTION Krátký popis účelu balíčku.

HOMEPAGE Domovská stránka balíčku.

SRC_URI Seznam zdrojových adres balíčku.

LICENSE Licence balíčku.

SLOT Slot balíčku. Pokud není vyžadován, je definované ”0”.

KEYWORDS Klíčová slova

IUSE Seznam použitých příznaků. Není vyžadován.

DEPEND Seznam závislostí potřebných pro sestavení balíčku.

RDEPEND Seznam závislostí potřebných pro spuštění aplikace.

src_configure() Funkce, která nakonfiguruje balíček.

src_install() Funkce, která instaluje balíček.

pkg_postints() Funkce, která se volá po instalaci balíčku.

5.5.2 Balíček pro KCM modul

Při vytváření balíčků je nutné udržovat adresářovou strukturu, podle které se balíček zařazuje do určité kategorie. První vytvořený soubor musí obsahovat název podle kategorie, do které patří. Přehled kategorií je dostupný na stránkách Gentoo, viz [20]. Balíček pro KCM modul jsem zařadil do kategorie **kde-plasma**, kde se nachází i ostatní systémové moduly. Další soubor musí mít název podle názvu balíčku, tedy **kcmhowdy**. V tomto souboru se nachází soubor ebuild.

Ebuild a Gentoo závislosti

V souboru ebuild bylo zapotřebí pracovat s frameworkem KDE. Toho lze dosáhnout použitím příkazu **inherit kde5**, čímž zdědíme KDE5 eclass. Eclass obsahuje kolekci funkcí, které můžou být použity v souboru ebuild. Závislosti na jednotlivých knihovnách jsou řešené pomocí funkcí *add_frameworks_dep* a *add_qt_dep*. Funkce *add_frameworks_dep* zajistí potřebné závislosti z frameworku KDE a funkce *add_qt_dep* zajistí závislosti na frameworku Qt. Dále je také potřeba CMake s minimální verzí 3.12. Výsledný ebuild pro KCM modul lze vidět na obrázku 5.5.

5.5.3 Balíček pro Howdy

Instalační balíček Gentoo pro Howdy opět vychází ze sekce 5.1 Instalace Howdy. V Ebuild souboru jsou použité funkce jako *insinto*, která změní lokaci pro instalaci. Lokací pro instalaci se myslí adresář */lib/security/howdy*. Instalace se následně provádí funkcí *doins*, která zkopíruje soubory ze složky *src* do adresáře definovaného u funkce *insinto*. Dále se vytvoří symbolický link funkcí *dosym* a přenastaví práva skriptu */lib/security/howdy/cli.py* funkcí *fperms*. A na závěr funkcí *dodoc* se zkopíruje soubor s licencí a README. Tento postup je proveden ve funkci *src_install()*. Po instalaci je zavolána funkce *pkg_postinst()*, která vypisuje uživateli další kroky pro správné nastavení.

```
# Copyright 2019 Gentoo Authors
# Distributed under the terms of the GNU General Public License v2

EAPI=7

inherit kde5

DESCRIPTION="KDE Plasma 5 control module for for Howdy"

HOMEPAGE="https://github.com/dusekpe2/${PN}.git"
SRC_URI="https://github.com/dusekpe2/${PN}/archive/v${PV}.tar.gz"

LICENSE="GPL-2+"
SLOT="0"
KEYWORDS="~amd64 ~x86"
IUSE=""

DEPEND="
        $(add_frameworks_dep kauth)
        $(add_frameworks_dep kconfigwidgets)
        $(add_qt_dep qtwidgets)
        >=dev-util/cmake-3.12
"

RDEPEND=" ${DEPEND}
        =sys-auth/howdy-2.5.1
"

src_configure() {
        kde5_src_configure
}

pkg_postinst() {
        kde5_pkg_postinst
        elog "Complete"
}
```

Obrázek 5.5: Ebuild soubor pro KCM modul

Kapitola 6

Testování

Poslední kapitolou je testování aplikace. Testování bylo prováděno na vytvořeném KCM modulu. Rozdělil jsem testování do dvou částí. První část je zaměřená na funkční testování, které bylo třeba provádět i v průběhu vývoje aplikace. Část druhá se zabývá testováním použitelnosti se skupinou uživatelů.

6.1 Funkční testování

Funkční testování se zaměřuje na to, jestli aplikace funguje správně a plní všechny úkoly, pro které je určena. Tímto testováním by se měly projít všechny požadavky stanovené v analýze a zjistit, zda jsou splněné a správně implementované. Během testování jsem narazil na několik chyb, které bylo potřeba opravit. Jednotlivě jsou popsané níže.

Chyba při přidání nového modelu

Přidávání modelů sice proběhlo úspěšně, ale při implementaci jsem netestoval hraniční hodnoty a těmi jsou přidání modelu s prázdným a nebo hodně dlouhým jménem. Při prázdném jménu se název neuložil a třídě `HowdyAuthHelper` se odeslala prázdná hodnota. Z toho důvodu se žádný model nepřidal. Vyřešil jsem to podmínkou ve třídě `HowdyAuthHelper`, která ověří jestli je jméno prázdné. Pokud je prázdné, je příkaz spuštěn s parametrem `-y` a program Howdy doplní název modelu automaticky. Ošetření shora při velmi dlouhém řetězci jsem opravil omezením maximální velikosti editačního textu.

Chybná synchronizace dat

Chybné chování nastávalo v případě vymazání všech modelů. Implementoval jsem sice synchronizaci se souborem, ale v tomto případě se smazal soubor celý. Problém byl vyřešen přidáním adresáře se souborem také třídě `QFileSystemWatcher` a odchytáváním signálu změny adresáře. Tedy pokud

se objeví nějaký soubor (s požadovaným jménem a daty) v tomto adresáři, bude se opět kontrolovat jeho obsah a aktualizovat tabulka.

6.2 Testování použitelnosti

Testováním použitelnosti se zjišťuje, jestli je aplikace správně navržena a s jakými problémy by se mohli při používání setkat budoucí uživatelé. Také se dá zjistit, které prvky nejsou příliš srozumitelné a zda se dokáží uživatelé v aplikaci orientovat. Nejlepší je samozřejmě testovat na skupině uživatelů, kteří budou aplikaci sami využívat.

Uživatelům z testovací skupiny jsem vytvořil testovací scénář, kterým si každý zkusil projít a odpovědět na otázky ve vytvořeném dotazníku. Testovací skupina byla sestavena z uživatelů různých věkových skupin. Avšak žádný z nich neměl zkušenosti s prostředím KDE Plasma 5.

6.2.1 Testovací scénář

Testovací scénář byl vytvořen tak, aby pokrýval všechny stanovené funkční požadavky. Každý uživatel si vyzkoušel následující úkoly, při kterých jsem pozoroval jejich chování.

1. Otevřete si modul v systémovém nastavení.
2. Pokuste se přidat nový model obličeje.
3. Zobrazte si tabulku s modely obličeje.
4. Odeberte libovolný model obličeje.
5. Odeberte všechny obličeje.
6. Vypněte ověřování pomocí Howdy.
7. Přenastavte hodnotu v konfiguračním souboru a uložte změny.
8. Vraťte zpět provedené změny.
9. Povolte ověřování pomocí Howdy.

6.2.2 Dotazník

Po vyzkoušení testovacího scénáře uživatelé vyplnili anonymní dotazník, který jim byl zadán. Dotazník byl složený z následujících otázek:

1. Co se Vám v aplikaci líbilo?
2. Připadala Vám aplikace složitá?

3. Co byste zlepšili?
4. Co se Vám na aplikaci líbilo?
5. Ohodnoťte aplikaci od jedné do deseti.

6.2.3 Shrnutí výsledků

Skupina testovacích uživatelů byla složená převážně z osob v mém okolí. Cílová skupina nebyla příliš rozmanitá, jelikož se skládala převážně z osob, kteří jsou na průměrné úrovni a žádný z nich prostředí KDE Plasma 5 nikdy předtím nepoužíval.

První uživatel prošel testovacím scénářem bez problému. Další uživatel přemýšlel nad tím, co znamenají jednotlivé položky v konfiguraci, ale scénářem také prošel. Poslední uživatel také zvládl projít scénářem bez obtíží.

V dotazníku byla aplikace hodnocená převážně kladně. Jako návrh vylepšení byla například možnost filtrování modelů v tabulce. Mezi kladnými odpovědmi převažovala jednoduchost aplikace. Výsledné hodnocení aplikace bylo 22 bodů ze 30.

Z pozorování práce uživatelů a dotazníku jsem usoudil, že by bylo dobré přidat některá vylepšení, jako filtrování modelů, vysvětlivky u položek konfigurace nebo zobrazování aktuálního výstupu z webkamery při pořizování modelů obličeje. Aplikace by mohla být časem vylepšena a doplněna o tyto funkcionality.

Závěr

Cílem práce bylo vytvoření aplikace, jež umožní uživateli přihlášení do systému Linux pomocí obličeje s využitím existujících projektů.

Nejdříve jsem provedl rešerši, při které jsem narazil na několik fungujících a několik nefungujících projektů, a zvolil jsem projekt, který je na dobré úrovni a funguje podle očekávání. Následně jsem provedl analýzu KCM modulu, který pracuje s vybraným projektem. Při analýze jsem si rozdělil budoucí výslednou aplikaci na třídy, sepsal funkční a nefunkční požadavky a vytvořil doménový model.

Poté jsem přistoupil k implementaci, jež vycházela z předchozí analýzy. Implementaci jsem věnoval poměrně hodně času, zejména kvůli dokumentaci knihovny Qt, ve které jsem se zpočátku nedokázal vůbec orientovat. Také pochopení principů autentizace jednotlivých akcí pro mě bylo zpočátku velmi složité. Po úspěšné implementaci systémového modulu jsem řešení zhodnotil z hlediska bezpečnosti. Součástí práce bylo také vytvoření instalačních balíčků pro distribuce Gentoo,Linux Mint a OpenSUSE. Tvorba instalačních balíčků byla poměrně náročná, jelikož mám zkušenosti pouze s distribucemi založenými na Debianu a s ostatními použitými distribucemi jsem se setkal poprvé. Celé řešení bylo potřeba na závěr otestovat, což jsem provedl pomocí funkčních testů a testováním použitelnosti.

Výsledkem této bakalářské práce je funkční aplikace splňující zadání a předchozí stanovené požadavky. Všechny stanovené cíle byly úspěšně splněny i s vytvořením instalačních balíčků pro zadané distribuce.

Našly by se určitě věci k vylepšení, například rozložení a vzhled systémového modulu. V budoucnu by bylo dobré aplikaci lépe otestovat, zejména kvůli odhalení případných chyb a přidat lokalizaci do různých jazyků.

Literatura

1. MORGAN, Andrew G.; KUKUK, Thorsten. *The Linux-PAM System Administrators' Guide* [online]. 2010 [cit. 2019-03-27]. Dostupné z: http://www.linux-pam.org/Linux-PAM-html/Linux-PAM_SAG.html.
2. CONLEY, Devin A. *pam-facial-auth* [online]. 2017 [cit. 2019-04-10]. Dostupné z: https://github.com/devinaconley/pam-facial-auth.
3. INTEL CORPORATION; GARAGEA, Willow; ITSEEZ. *OpenCV* [online]. 2019 [cit. 2019-04-10]. Dostupné z: https://opencv.org.
4. MEISBERGER, Philipp. *PAM Face* [online]. 2019 [cit. 2019-04-10]. Dostupné z: https://github.com/philippmeisberger/pam-face.
5. KING, Davis E. *dlib-models* [online]. 2018 [cit. 2019-04-10]. Dostupné z: https://github.com/davisking/dlib-models.
6. SEVEREIN, Lem. *Howdy* [online]. 2019 [cit. 2019-04-10]. Dostupné z: https://github.com/boltgolt/howdy.
7. THE QT COMPANY LTD. *Qt Documentation* [online]. 2019 [cit. 2019-03-27]. Dostupné z: https://doc.qt.io/qt-5/signalsandslots.html.
8. *KDE TechBase* [online]. 2012 [cit. 2019-03-27]. Dostupné z: https://techbase.kde.org/Development/Tutorials/KAuth/KAuth_Basics#What_is_KAuth.
9. THE QT COMPANY LTD. *Qt Documentation* [online]. 2019 [cit. 2019-03-27]. Dostupné z: https://doc.qt.io/qt-5/designer-using-a-ui-file.html.
10. THE QT COMPANY LTD. *Qt Documentation* [online]. 2019 [cit. 2019-03-27]. Dostupné z: https://doc.qt.io/qt-5/modelview.html.
11. THE QT COMPANY LTD. *Qt Documentation* [online]. 2019 [cit. 2019-03-27]. Dostupné z: https://doc.qt.io/qt-5/model-view-programming.html.

12. GRAFF, Eliot; WOOD, Dawn; HUDEK, Ted; BAXTER, Joshua. *Microsoft Documentation* [online]. 2017. Dostupné také z: https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/windows-hello-face-authentication.

13. BOBČÍK, Boleslav. *PAM - správa autentizačních mechanismů* [online]. 2000 [cit. 2019-03-27]. Dostupné z: https://www.root.cz/clanky/pam-sprava-autentizacnich-mechanismu/.

14. PROJECT, The Debian. *Debian Wiki* [online]. 2019 [cit. 2019-03-27]. Dostupné z: https://wiki.debian.org/Packaging/SourcePackage.

15. PRŮDEK, Miloš; BURDA, Michal. *Binární balíčky* [online]. 2000 [cit. 2019-06-12]. Dostupné z: https://www.root.cz/specialy/linux-na-serveru/binarni-balicky/.

16. THE DEBIAN PROJECT. *Debian Policy Manual* [online]. 2019 [cit. 2019-06-10]. Dostupné z: https://www.debian.org/doc/debian-policy.

17. HESS, Joey. *Debian Manpages* [online]. 2019 [cit. 2019-05-13]. Dostupné z: https://manpages.debian.org/testing/debhelper/debhelper.7.en.html.

18. OPENSUSE CONTRIBUTORS. *OpenSUSE Wiki* [online]. 2016 [cit. 2019-05-13]. Dostupné z: https://en.opensuse.org/Package_management.

19. GENTOO FOUNDATION, INC. *Gentoo Linux* [online]. 2019 [cit. 2019-05-13]. Dostupné z: https://gentoo.org/get-started/about.

20. GENTOO FOUNDATION, INC. *Gentoo packages database* [online]. 2019 [cit. 2019-05-13]. Dostupné z: https://packages.gentoo.org/categories.

Příloha A

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

PAM Pluggable Authentication Modules

MVC Model View Controller

API Application Programming Interface

UI User Interface

JSON Javascript Object Notation

SSH Secure Shell

LBPH Local Binary Patterns Histogram

KCM KDE Config Module

KF5 KDE Frameworks 5

PMS Package Manager Specification

HOG Histogram of Oriented Gradients

CNN Convolutional Neural Network

GPL General Public License

Příloha B

Instalační příručka

Aplikace je určena pro operační systémy obsahující desktopové prostředí KDE Plasma 5. Samotná aplikace je možná nainstalovat pomocí připravených instalačních balíčků pro distribuce založené na Debianu, distribuci OpenSUSE a Gentoo. Další možností je instalace pomocí nástrojů CMake a make. Postup instalace je obsažený v souboru README.md.

B.1 Instalace z připravených balíčků

Balíčky pro distribuce založené na Debianu jsem úspěšně testoval na čisté instalaci systému Kubuntu 18.10 64-bit. Balíčky se instalují příkazem **dpkg -i DEB_PACKAGE**. Jelikož balíček kcmhowdy je závislý na programu Howdy ve verzi 2.5.1 musí se nejdříve nainstalovat Howdy s touto verzí. Příkazem **apt-get install -f** lze vyřešit potřebné závislosti automaticky.

Balíčky pro distribuce OpenSUSE jsem testovat na čisté instalaci systému OpenSUSE Tumbleweed. Balíčky se dají nainstalovat dvěma způsoby. První z nich je přidání repozitáře[1] příkazem **zypper addrepo** . Dále se musí aktualizovat přikazem **zypper refresh** a příkazem **zypper install kcmhowdy** se nainstaluje systémový modul zároveň s projektem Howdy. Také jsou automaticky vyřešeny závislosti. Další možností je instalace pomocí rpm přiložených balíčků. Rozdíl je v tom, že si uživatel musí vyřešit závislosti. Příkazem **rpm -i RPM_PACKAGE** se nejdříve nainstaluje přiložený balíček python3-dlib, poté balíček howdy a nakonec balíček kcmhowdy.

Balíčky pro distribuce Gentoo jsem testovat na čisté instalaci systému Gentoo Base System release 2.6. Instalace se provádí buďto příkazem **ebuild GENTOO_PACKAGE manifest clean merge**, který vytvoří manifest a provede merge. U tohoto postupu je nutné zjistit si závislosti a nainstalovat všechny potřebné balíčky. Další možností je přidání soborů systému Portage

[1] `https://download.opensuse.org/repositories/home:dusekpe2/openSUSE_Tumbleweed/home:dusekpe2.repo`

a spuštěním příkazu **emerge CATEGORY/PACKAGE_NAME**. Například **emerge sys-auth/howdy**.

Příloha C

Obsah přiloženého USB

- `readme.txt` ..... stručný popis obsahu USB
- `packages` ..... binární instalační balíčky
  - `debian` ..... deb instalační balíčky
  - `opensuse` ..... rpm instalační balíčky
- `src`
  - `kcmhowdy` ..... zdrojové kódy implementace
    - `debian` ..... zdrojové kódy balíček debian
    - `gentoo` ..... zdrojové kódy balíček gentoo
    - `opensuse` ..... zdrojové kódy balíček opensuse
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF