VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Aplikace pro Android pro automatizaci činností na počítači

Android App for PC Operation Automation

Petr Krautwurst

Bakalářská práce

Vedoucí práce: Ing. Pavel Dohnálek, Ph.D.

Ostrava, 2024

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/23/113057
vsboee23037A0C

Zadání bakalářské práce

Student: **Petr Krautwurst**

Studijní program: B0613A140014 Informatika

Téma: Aplikace pro Android pro automatizaci činností na počítači
Android App for PC Operation Automation

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je vypracovat aplikaci pro operační systém Android, která bude na displeji zařízení zobrazovat tlačítka představující makroklávesy. Každé tlačítko bude možné naprogramovat tak, aby se po jeho stisku vykonala konkrétní posloupnost akcí – pohybu myší, stisku tlačítek myší, stisku klávesy na klávesnici, napsání konkrétního znaku/symbolu apod. Tyto posloupnosti akcí (makra) budou prováděny na počítači, k němuž bude zařízení s Androidem připojeno prostřednictvím USB nebo Wi-Fi sítě. Řešení tedy bude sestávat ze dvou částí – samotné aplikace pro Android, ze které budou posílány pokyny počítači, a serveru, který bude tyto pokyny přijímat a vykonávat na počítači.
Serverová část na straně počítače bude obsahovat i administrativní rozhraní, v němž bude možné definovat makra a v grafickém prostředí organizovat makrotlačítka zobrazovaná v aplikaci pro Android. Minimálními funkcemi aplikace pro Android bude zobrazování a uspořádávání makrotlačítek, a to responzivně s ohledem na rozlišení displeje zařízení, a odesílání pokynů počítači. Uživatelské rozhraní bude uzpůsobeno e-inkovým displejům, a to v barevné i černobílé variantě. Změnu uspořádání tlačítek v aplikaci musí být možné přenést do administrativního rozhraní na počítači a naopak. Počítač bude používat operační systém Windows.

Práce bude obsahovat:
1. Specifikaci problému a rešerši obdobných řešení dostupných na trhu.
2. Návrh vlastního řešení problémů a zdůvodnění zvolených postupů.
3. Implementaci všech povinných částí řešení.
4. Zhodnocení reálné použitelnosti a případných omezení výsledného řešení.

Seznam doporučené odborné literatury:

[1] Serhan Yamacli. 2017. Beginner's Guide to Android App Development: A Practical Approach for Beginners (1st. ed.). CreateSpace Independent Publishing Platform, North Charleston, SC, USA.
[2] Christian Nagel. 2018. Professional C# 7 and .NET Core 2.0 (7th. ed.). Wrox Press Ltd., GBR.

Další publikace dle pokynů vedoucího bakalářské práce.

17. listopadu 2172/15
708 00 Ostrava-Poruba
Česká republika

spojovatelka: +420 597 321 111
epodatelna: epodatelna@vsb.cz
ID datové schránky: d3kj88v

IČ: 61989100
DIČ: CZ61989100

email: studijni.fei@vsb.cz
www.fei.vsb.cz

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Dohnálek, Ph.D.**

Datum zadání: 01.09.2023

Datum odevzdání: 30.04.2024

Garant studijního programu: doc. Mgr. Miloš Kudělka, Ph.D.

V IS EDISON zadáno: 09.11.2023 10:38:27

Abstrakt

Tato bakalářská práce je cílená na vytvoření aplikace pro operační systém Android, která umožní spouštět makra na počítači. V první části se zaměřuje na to, co jsou to makra a jak se využívají k automatizaci činností na počítači. Dále se věnuje analýze existujících řešení a jejich možnostem. Práce následně popisuje, jak vytvořit nástroj, který by umožňoval jednoduchý a intuitivní způsob, jak vytvářet počítačová makra a spouštět je z mobilního zařízení. Výslednou aplikací je mobilní aplikace, serverová část a administrativní rozhraní.

Klíčová slova

Automatizace; Makro; Vzdálené ovládání; Android

Abstract

This bachelor's thesis is focused on creating an application for the Android operating system that will allow user to run macros on a computer. The first part focuses on what macros are and how they are used to automate activities on a computer. It then analyzes existing solutions and their capabilities. The thesis then describes how to create a tool that would allow a simple and intuitive way to create computer macros and run them from a mobile device. The resulting application is a mobile application, server part and administrative interface.

Keywords

Automation; Macro; Remote control; Android

Poděkování

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce, Ing. Pavlu Dohnálkovi, Ph.D., za věnovaný čas, cenné rady a ochotu ke konzultacím. Dále bych chtěl poděkovat své rodině a přátelům za podporu a trpělivost.

Obsah

Seznam použitých zkratek a symbolů

COM	–	Component Object Model
IP	–	Internet Protocol
JSON	–	JavaScript Object Notation
RPC	–	Remote Procedure Call
SDK	–	Software Development Kit
TCP	–	Transmission Control Protocol
UTF-8	–	Unicode Transformation Format 8-bit

Seznam výpisů zdrojového kódu

Seznam obrázků

Kapitola 1

Úvod

V dnešní době, kdy se digitalizace stále prohlubuje a naše interakce s technologiemi se stávají čím dál komplexnějšími, roste i potřeba efektivního řešení pro automatizaci opakujících se úkolů. Lidé se čím dál častěji setkávají s nutností provádět rutinní činnosti na svých počítačích, ať už se jedná o opakované úpravy dokumentů, řízení aplikací či jiné činnosti, které vyžadují opakované zásahy uživatele. Tyto úkoly nejenže zabírají čas, ale jsou také náchylné k lidským chybám, což může mít za následek nejen ztrátu času, ale i chybné výstupy nebo nedokončené procesy.

Tato práce se zaměří na vytvoření aplikace, která umožní uživatelům snadno a efektivně automatizovat činnosti na svých počítačích. Bude navržena s ohledem na uživatelskou přívětivost a dostupnost, přičemž bude klást důraz na jednoduchost vytváření a úpravy automatizačního procesu. Cílem této práce je tedy vytvořit aplikaci pro operační systém Android, jež bude obsahovat především makrotlačítka, po jejichž stisknutí se na počítači spustí předem vytvořené makro sestávající ze simulovaných uživatelských akcí. Uživatelé budou moci v grafickém rozhraní na počítači vytvářet a upravovat tato makra, která budou obsahovat posloupnosti příkazů, jenž se mají na počítači provádět.

Práce je rozdělena do čtyř hlavních kapitol. V kapitole 2 jsou popsána existující řešení a jejich možnosti a nedostatky. Na základě této analýzy jsou v kapitole 3 navrženy požadavky na vlastní řešení. Dále je v této kapitole popsáno, jak by mělo vlastní řešení fungovat a jaké by mělo mít vlastnosti. Kapitola 4 popisuje implementaci vlastního řešení, které se skládá z mobilní aplikace pro operační systém Android, serverové části a administrativního rozhraní. V kapitole 5 je provedeno zhodnocení výsledného řešení a navržena možná vylepšení a rozšíření.

Kapitola 2

Specifikace problému

V této kapitole se podíváme na problém automatizace činností na počítači pomocí maker. Nejprve se zaměříme na obecný popis automatizace, maker a jejich využití, poté prozkoumáme existující řešení a jejich výhody a nevýhody.

2.1 Automatizace činností a její význam

Automatizace činností se stává stále důležitější součástí pracovního prostředí. Jedná se o proces implementace technologií a systémů, které umožňují provádění úkolů s minimálním nebo žádným lidským zásahem. Automatizace přináší mnoho výhod. Umožňuje efektivní využití času a zdrojů tím, že uvolňuje lidi od rutinních úkolů a umožňuje jim zaměřit se na složitější práci.[1]

2.1.1 Makra a skripty

Makro je určitá posloupnost příkazů sloužící k automatizaci opakujících se úkolů na počítači. Zajišťují provedení těchto příkazů jako reakci na jednu akci, například stisknutí klávesy či tlačítka. Makra klávesnice a myši obvykle zahrnují příkazy jako stisk klávesy, pohyb či kliknutí myši nebo zadání určitého symbolu. Mohou být většinou vytvářena v grafických editorech, kde je možné jednoduše přidávat, odebírat či upravovat jednotlivé příkazy. Často se setkáme s nástroji, které pro jednoduchost vytváření nabízí možnost nahrávání akcí. Makra bývají zaměřena spíše na koncové uživatele, kterým vyhovuje intuitivní ovládání a jednoduchost použití.[2]

Skripty jsou, podobně jako makra, posloupnosti příkazů, ale většinou se jedná o zdrojové soubory s příkazy, které mohou být spuštěny v rámci určitého skriptovacího jazyka. Většinou umožňují na rozdíl od maker také využívání proměnných a řízení toku, jako například podmíněné větvení či opakování částí programu. Skripty jsou často vytvářeny programátory nebo pokročilými uživateli, kteří potřebují větší flexibilitu a možnost programování složitějších úkolů. Obvykle jsou psány v textových editorech a vyžadují znalost daného skriptovacího jazyka.[3]

2.2 Využití maker v automatizaci

Makra jsou užitečným nástrojem pro automatizaci různých činností na počítači. Běžně se využívají pro vyplňování formulářů, práci s textovými editory, administraci systému a v mnoha dalších oblastech. Existuje více typů maker, které mohou být využívány pro různé účely. Mezi ně patří například makra klávesnice a myši, na které se tato práce zaměřuje, makra pro specifický software – například makra součástí nástrojů Microsoft Office, makra nástroje Adobe Photoshop (kde jsou nazvána akcemi) nebo makra pro webové prohlížeče, která mohou být využita pro automatizaci činností na internetu.[2]

Mohou být také využívána pro automatizaci různých úkonů ve hrách, jako je například sbírání různých surovin, předmětů, plnění úkolů či boj s nepřáteli. V některých případech mohou být dokonce využita k vytvoření takzvaných botů, jejichž účelem je hrát hru za hráče. Zde je ale třeba brát v potaz etiku a pravidla dané hry, neboť tímto způsobem může docházet ke zhoršení zážitku ostatních hráčů nebo k potenciálnímu trestání ze strany provozovatele hry.

2.2.1 Výhody a nevýhody použití maker

Využití maker má řadu výhod, ale také několik nevýhod. Zde se podíváme na některé z nich:

- **Výhody:**
  - **Flexibilita a přizpůsobivost:** Makra umožňují uživatelům vytvářet a upravovat automatizační postupy podle svých individuálních potřeb a preferencí. Díky tomu mohou být snadno přizpůsobena různorodým úkolům a situacím bez nutnosti programování konkrétních řešení pro každý úkol.
  - **Snadná použitelnost:** Makra bývá snadné vytvářet a používat, bývají často dostupná v intuitivních grafických editorech, které umožňují jednoduché přidávání a úpravu příkazů.
  - **Možnost využití v celém systému:** Makra klávesnice a myši lze využívat v celém systému, je jich tedy možné využít v mnoha různých oblastech a aplikacích, od kancelářských programů, přes hry, až po specifické systémové úkoly.
- **Nevýhody:**
  - **Omezené využití:** Makra mohou být omezena v možnostech provádění složitějších úkolů, obzvláště pokud nelze deterministicky předpokládat možné scénáře a reakce na ně.
  - **Náročnost nastavení:** Nastavení a ladění může vyžadovat technické znalosti a časovou investici, zejména při složitějších úkolech.

2.3 Existující řešení

Existuje mnoho nástrojů, které umožňují automatizaci pomocí maker či skriptů. Mezi nejznámější patří aplikace jako AutoHotkey, která umožňuje vytvářet velmi flexibilní skripty, nebo zařízení Stream Deck, které poskytuje fyzické tlačítkové rozhraní pro spouštění funkcí a maker ve spojitosti s různými aplikacemi, zejména v oblasti streamování.

V této sekci zmapujeme některé z těchto nástrojů, prozkoumáme jejich funkce, možnosti, výhody a nevýhody, abychom lépe porozuměli současnému stavu technologií v této oblasti.

2.3.1 AutoHotkey

Jedním z nejznámějších takových nástrojů je AutoHotkey. Využívá vlastní open-source skriptovací jazyk pro Windows a poskytuje uživatelům možnost vytvářet jak jednoduché, tak i komplexní skripty pro širokou škálu úkolů. Mezi nejvýznamnější funkce patří klávesové a textové zkratky, které zajišťují spuštění určité funkce po zadání klávesové zkratky či po zadání určitého textu. Dále také simulace kláves a myši, spouštění programů nebo automatické vyplňování formulářů.

Mezi jeho hlavní výhody patří jednoduchost použití. I komplexnější skripty lze vytvořit pomocí jednoduché syntaxe. Zároveň má velkou komunitu uživatelů, kteří sdílejí své skripty a doporučení, což usnadňuje vyhledávání řešení pro vlastní skripty. Také nabízí nástroj pro kompilaci skriptů do samostatně spustitelných souborů.[4]

Nástroj má však i pár nevýhod. Tou největší je absence grafického uživatelského rozhraní pro editaci skriptů. I přes jednoduchost syntaxe, kvalitní dokumentaci a velkou komunitu může být tvorba složitějších skriptů náročná pro uživatele bez programátorských znalostí. Další nevýhodou může být omezená podpora pro jiné operační systémy než Windows.

AutoHotkey je tedy velice užitečný a všestranný nástroj pro automatizaci práce na počítači, nicméně specializuje se zejména na spouštění skriptů pomocí kláves či klávesových zkratek a nenabízí řešení pro uživatele, kteří chtějí tyto skripty spouštět z mobilního zařízení.

2.3.2 Stream Deck

Dalším nástrojem je zařízení Stream Deck od společnosti Elgato. Jedná se o jednoduché fyzické tlačítkové rozhraní, jaké můžeme vidět na obrázku 2.1, které umožňuje uživatelům stiskem tlačítka spouštět různé funkce či makra definované v aplikaci na počítači. Aplikace nabízí jednoduché a přívětivé uživatelské rozhraní a také širokou škálu funkcí, jako je ovládání hlasitosti, streamovacích aplikací, osvětlení, spouštění programů a mnoho dalších. Lze také tlačítku přiřadit skupinu akcí, které se po spuštění všechny provedou – ať už najednou, nebo s definovanými časovými intervaly mezi jednotlivými akcemi. Aplikace není omezena základním seznamem funkcí, ale umožňuje přidávání rozšíření, která mohou být stažena z oficiálního obchodu. Mezi nejpopulárnější rozšíření patří například integrace s OBS Studio, Spotify, Twitter nebo YouTube.[5]

Obrázek 2.1: Stream Deck [6]

Stream Deck je oblíbený nejen mezi streamery a youtubery, ale také mezi dalšími uživateli, kteří potřebují snadný a efektivní způsob ovládání různých funkcí bez nutnosti používání klávesnice nebo myši. Díky tomu, že se jedná o fyzické zařízení, je možné jednoduše spuštět funkce na počítači i bez přepínání mezi aplikacemi nebo okny. To je ale zároveň jeho nevýhodou, neboť je nutné fyzické zařízení zakoupit. Je možné stáhnout si aplikaci pro Android, která umožňuje vzdálené ovládání Stream Decku, ale tato aplikace je placená a v současnosti nenabízí bezplatnou verzi ani zkušební dobu.

2.3.3 AutoIt

AutoIt je freeware nástroj určený pro automatizaci úkolů ve Windows. Obsahuje vlastní skriptovací jazyk se syntaxí podobnou jazyku BASIC a nabízí běhové prostředí pro spouštění skriptů. Součástí je také textový editor, ve kterém lze skripty vytvářet a upravovat. Další užitečnou funkcí je možnost kompilace skriptů do samostatně spustitelných souborů pomocí přiloženého kompilátoru.

Ačkoli byl AutoIt skriptovací jazyk navržen speciálně pro automatiazci úkolů, poskytuje mnoho dalších funkcí. Mezi hlavní funkce patří:

- **Simulace interakce:** Skripty mohou simulovat interakci s uživatelem pomocí simulace stisku kláves a klikání myší.
- **Manipulace s okny:** Obecně funkce, jako je otevírání, zavírání, minimalizace, maximalizace a přepínání mezi okny.
- **Spouštění programů:** AutoIt umožňuje spouštění externích programů, volání funkcí z dynamicky linkovaných knihoven či volání funkcí z Windows API.
- **Práce s soubory a adresáři:** Standardní operace jako čtení, zápis, kopírování, mazání souborů a adresářů.

AutoIt také nabízí funkce pro tvorbu uživatelských grafických rozhraní. Uživatelé mohou vytvářet vlastní grafická rozhraní s různými prvky a propojovat je se svými skripty. Tímto způsobem lze

poměrně snadno vytvořit dialogová okna pro nastavení parametrů skriptu, zobrazování informací nebo jiné uživatelské interakce. Je však také možné takto vytvářet i jednoduché aplikace, například formulářové aplikace pro zadávání dat, jednoduché hry nebo třeba vlastní instalátory.

Další zajímavou funkcí je podpora pro COM (Component Object Model), což je technologie pro komunikaci mezi aplikacemi v operačním systému Windows. AutoIt umožňuje vytvářet COM objekty, volat metody a číst a zapisovat vlastnosti těchto objektů. To může být užitečné pro integraci s jinými aplikacemi, které podporují COM, nebo pro automatizaci úkolů v těchto aplikacích.[7]

Mezi nevýhody patří poměrně složitá syntaxe, potřeba porozumění alespoň základním konceptům programování, menší komunita uživatelů a menší flexibilita oproti jiným nástrojům, jako je AutoHotkey.

2.3.4 Shrnutí

Ačkoli autor této práce hledal, nenašel nástroj, který by umožňoval jednoduché vytvoření maker či skriptů pro simulaci vstupů od uživatele a jejich následné spouštění z mobilního zařízení. Většina nalezených nástrojů se zaměřuje na automatizaci přímo na počítači, například AutoHotkey či AutoIt, nebo se naopak jedná spíše o nástroje pro vzdálenou správu, nikoli pro automatizaci.

Nejblíže k požadavkům této práce se zdá být zařízení Stream Deck, které sice umožňuje jednoduché vytváření maker a jejich spouštění pomocí fyzického tlačítkového rozhraní, ale neumožňuje spouštění maker z mobilního zařízení bez nutnosti placení za aplikaci. Toto řešení není přímárně zaměřeno na simulaci vstupů od uživatele, ale spíše na spouštění funkcí v rámci různých aplikací. I přes to, že existují rozšíření pro simulaci vstupů, tato rozšíření nejsou součástí základního balíčku a poskytují tak spíše omezené možnosti pro toto využití, především při editaci sekvencí simulovaných vstupů.

Kapitola 3

Návrh vlastního řešení

V této kapitole se zaměříme na návrh vlastního řešení. Nejdříve specifikujeme požadavky na aplikaci, poté se podíváme na návrh funcionalit, návrh makra a příkazů. Nakonec se zaměříme na návrh jednotlivých částí aplikace a komunikaci mezi nimi.

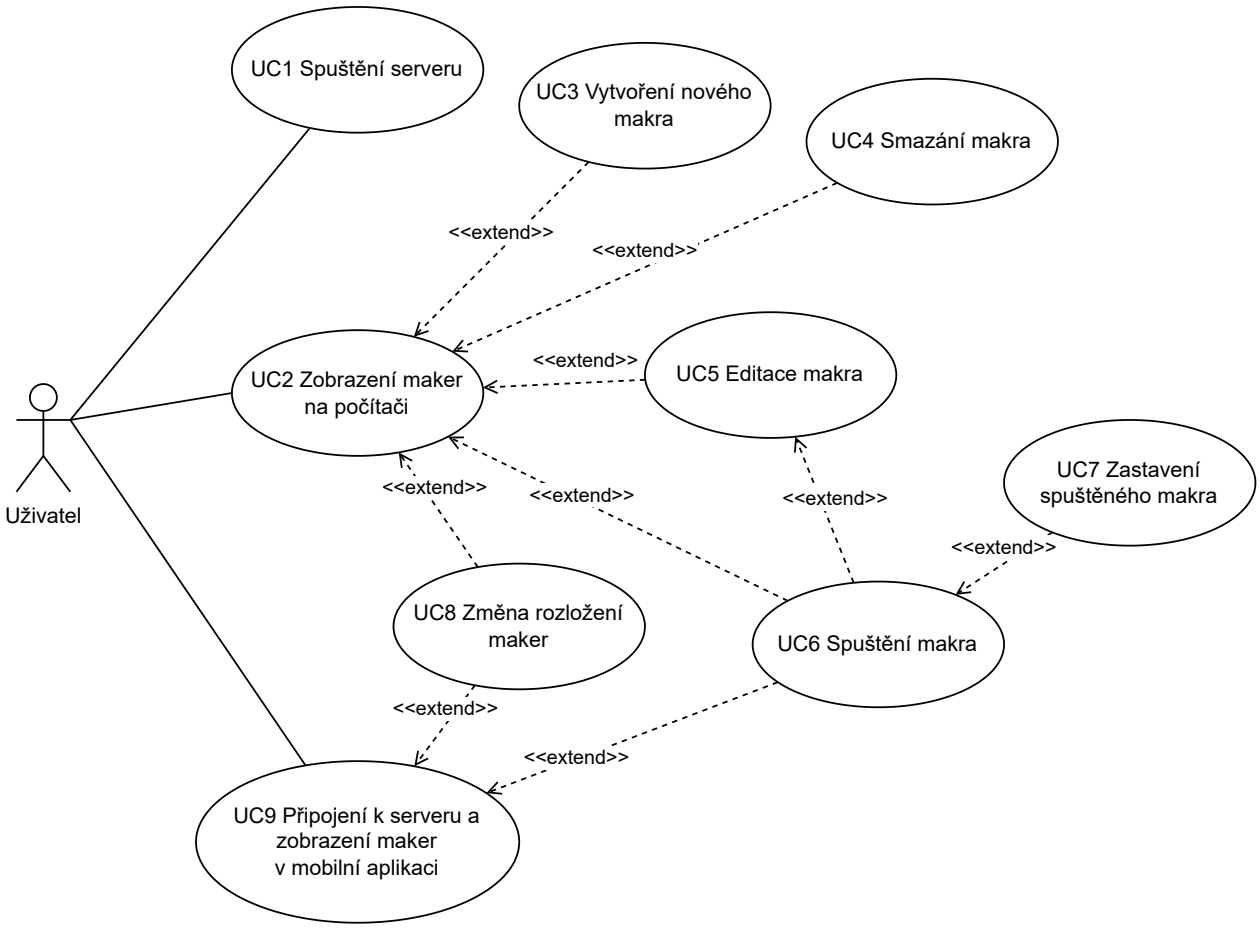
3.1 Požadavky na aplikaci

Výsledná aplikace bude splňovat následující požadavky:

- **Zobrazení makrotlačítek:** Aplikace bude uživateli responzivně zobrazovat tlačítka reprezentující makra.
- **Podpora e-inkových displejů:** Uživatelské rozhraní aplikace musí být uzpůsobeno pro e-inkové displeje v barevné i černobílé variantě.
- **Serverová část:** Na straně počítače musí existovat serverová část, která bude přijímat pokyny z aplikace pro Android a bude vykonávat korespondující akce. Mobilní část k ní bude připojena pomocí Wi-Fi sítě.
- **Administrativní rozhraní:** Serverová část musí obsahovat administrativní rozhraní, ve kterém bude možné makra definovat.
- **Makra:** Uživatel musí mít možnost programovat jednotlivá makra tak, aby po jejich spuštění byla vykonána určitá posloupnost akcí: pohyb myši, stisk tlačítka myši, stisk klávesy na klávesnici, napsání konkrétního textu či znaku.
- **Změna uspořádání tlačítek:** Změny v uspořádání tlačítek v aplikaci pro Android musí být možné převést do počítače a naopak.
- **Podpora operačního systému Windows:** Část na straně počítače musí být kompatibilní s operačním systémem Windows.

3.2 Funkcionality

Obrázek 3.1 zobrazuje diagram případů užití aplikace. Uživatel bude mít možnost spustit server, zobrazit makra v administrativním rozhraní, tam dále vytvořit nové či upravit existující makro, spouštět a případně zastavovat makra a měnit uspořádání tlačítek. v mobilní aplikaci se uživatel bude moci připojit k serveru a zároveň si tak zobrazit seznam maker, která budou na serveru dostupná, ta pak bude moci spouštět, případně zastavovat.



Obrázek 3.1: Diagram případů užití

3.2.1 Spuštění serveru

Uživatel bude mít možnost spustit serverovou část aplikace, která bude naslouchat na určité adrese a portu a bude přijímat pokyny od mobilní aplikace. Server bude mít možnost spouštět a zastavovat makra. Uživatel bude mít možnost nastavit následující parametry serveru:

- **IP adresa:** IP adresa, na které bude server naslouchat.

- **Port:** Port, na kterém bude server naslouchat.

- **Počet portů:** Pokud bude vybraný port obsazen, server se pokusí využít následující port. Uživatel bude moci nastavit maximální počet následujících portů, které má server vyzkoušet.

- **Firewall:** Server bude mít možnost nastavit za uživatele firewall. Vytvoří či upraví pravidlo tak, aby byl povolen přístup na daný port.

- **Autorizace:** Tato funkcionalita bude zajišťovat, že klienty, kteří se pokusí připojit na server, bude nutné autorizovat. Uživatel nastaví, zda bude autorizace povinná, nebo bude možné se připojit bez autorizace.

3.2.2 Administrativní rozhraní

Při spuštění administrativního rozhraní se načtou všechna makra, která se uživateli responzivně zobrazí. Uživatel bude mít možnost vytvořit nové makro, smazat existující makro, otevřít editor pro úpravu daného makra, spouštět a zastavovat makra a měnit uspořádání tlačítek.

Editor bude obsahovat sekvenci příkazů, které se mají po spuštění makra vykonat. Dále bude zobrazovat seznam všech dostupných příkazů, které lze přidat do sekvence. Po výběru příkazu se zobrazí jeho nastavení, které bude možné upravit. Příkazy bude možné přidávat, mazat a upravovat. Dále bude možné upravit nastavení makra jako jeho název, popis a jiné. Návrhu makra a příkazů se budeme věnovat v následující sekci.

Součástí editoru je také možnost nahrávání příkazů. Uživatel vybere, zda si přeje nahrávat vstupy z klávesnice nebo myši; v případě myši si dále může vybrat, které akce myši nahrávat (stisk tlačítka, pohyb myši, rolování myši). Pro nahrávání pohybu myši půjde nastavit, jak často se má pohyb myši zaznamenávat.

3.2.3 Mobilní aplikace

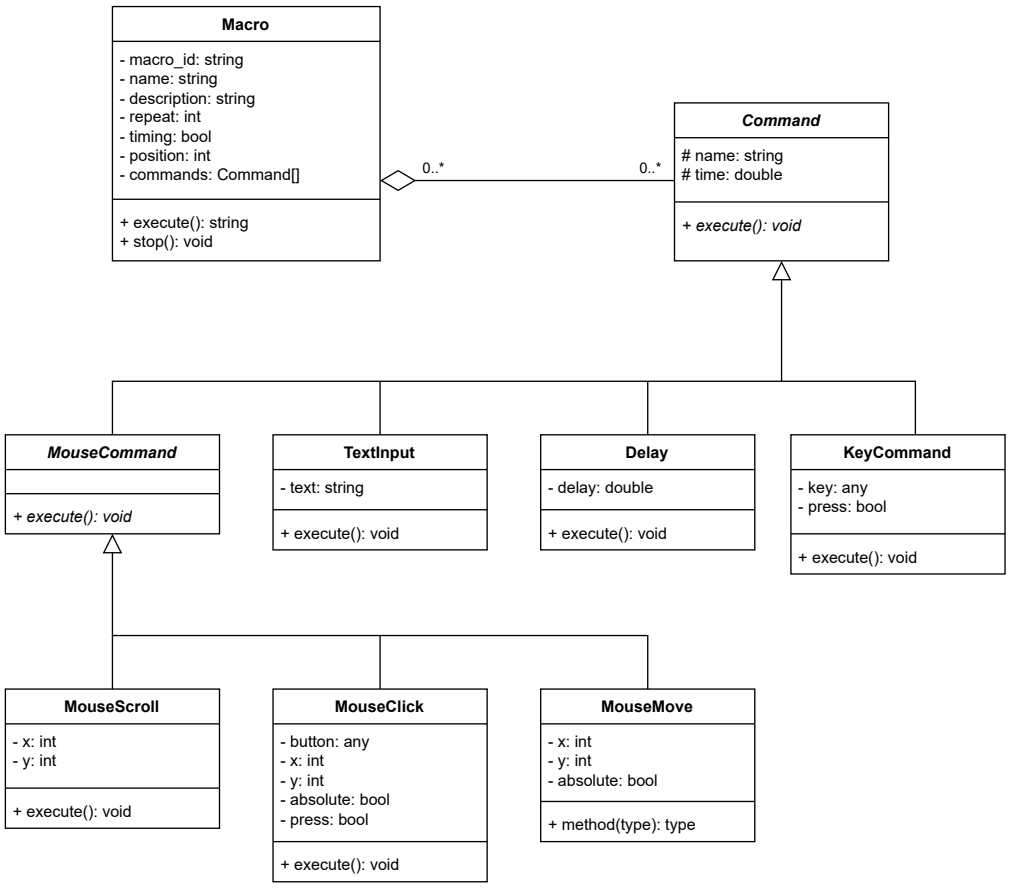
První funkcionalitou mobilní aplikace bude možnost připojení k serveru. Uživatel zadá IP adresu a port serveru, ke kterému se chce připojit. Pokud se připojení nezdaří, uživatel bude informován o chybě. Po úspěšném připojení se zobrazí seznam všech dostupných maker. U každého makra bude také tlačítko, pomocí kterého bude možné makro spustit či zastavit. Taktéž bude možné změnit uspořádání tlačítek, které se uloží na serveru.

Pokud bude k serveru připojeno více klientských zařízení, všechny akce provedené na jednom zařízení se projeví i na ostatních zařízeních. To znamená, že pokud uživatel spustí makro na jednom zařízení, makro se bude zobrazovat jako spuštěné i na ostatních zařízeních a z kteréhokoliv zařízení bude možné toto makro zastavit.

3.3 Makro a příkazy

Zde se zaměříme na návrh makra a jednotlivých příkazů. Na obrázku 3.2 je zobrazen třídní diagram, který znázorňuje vztah mezi makrem a příkazy. Makro bude obsahovat

- název,
- popis, který se spolu s názvem zobrazí v mobilní aplikaci,
- sekvenci příkazů, které se po spuštění makra vykonají,
- počet opakování, jenž udává, kolikrát se celá sekvence bude opakovat,
- informaci o tom, zda se má využívat systém časování příkazů,
- informaci o pozici daného makra v uspořádání tlačítek,
- identifikátor makra.



Obrázek 3.2: Třídní diagram makra a příkazů

Vykonání makra proběhne tak, že se postupně vykonají jednotlivé příkazy v sekvenci. Není však vždy žádoucí, aby se všechny příkazy vykonaly ihned po sobě v krátkém časovém úseku, proto je nutné nějakým způsobem umožnit časování příkazů. Toho lze jednoduše dosáhnout tak, že jeden z příkazů bude představovat časovou prodlevu – čekání po určitý čas. To však také nemusí být vždy ideální řešení. Vykonání každého příkazu zabere nějaký čas, což při velkém počtu příkazů může způsobit zbytečné zpoždění, které by mohlo být pro uživatele nežádoucí.

Proto bude možné využít systém časování příkazů. Tento systém bude založen na časových značkách, které budou přiřazeny k jednotlivým příkazům. Při vykonávání makra se vždy zkontroluje časová značka následujícího příkazu a porovná se s aktuálním časem. Pokud je časová značka menší než aktuální čas, příkaz se rovnou vykoná. v opačném případě se před vykonáním příkazu počká po dobu, jež je rozdílem mezi časovou značkou a aktuálním časem.

Příkazy můžeme rozdělit do tří kategorií – obecné příkazy (časová prodleva, zadání textu), příkazy pro práci s myší (pohyb myši, stisk tlačítka a rolování myši) a příkazy pro práci s klávesnicí (stisk klávesy). Každý příkaz bude mít několik atributů, které bude možné upravit. Všechny příkazy budou mít dva společné atributy. Jedním je označení či název příkazu, druhým je časová značka. Taktéž všechny příkazy budou mít metodu pro vykonání.

Příkaz pro časovou prodlevu bude mít navíc atribut pro dobu čekání. Příkaz pro zadání textu bude mít atribut pro text, jenž se má po spuštění napsat.

Příkaz pro stisk klávesy bude obsahovat dva další atributy – klávesu, které se tento příkaz týká, a informaci o tom, zda se má klávesa stisknout, nebo uvolnit.

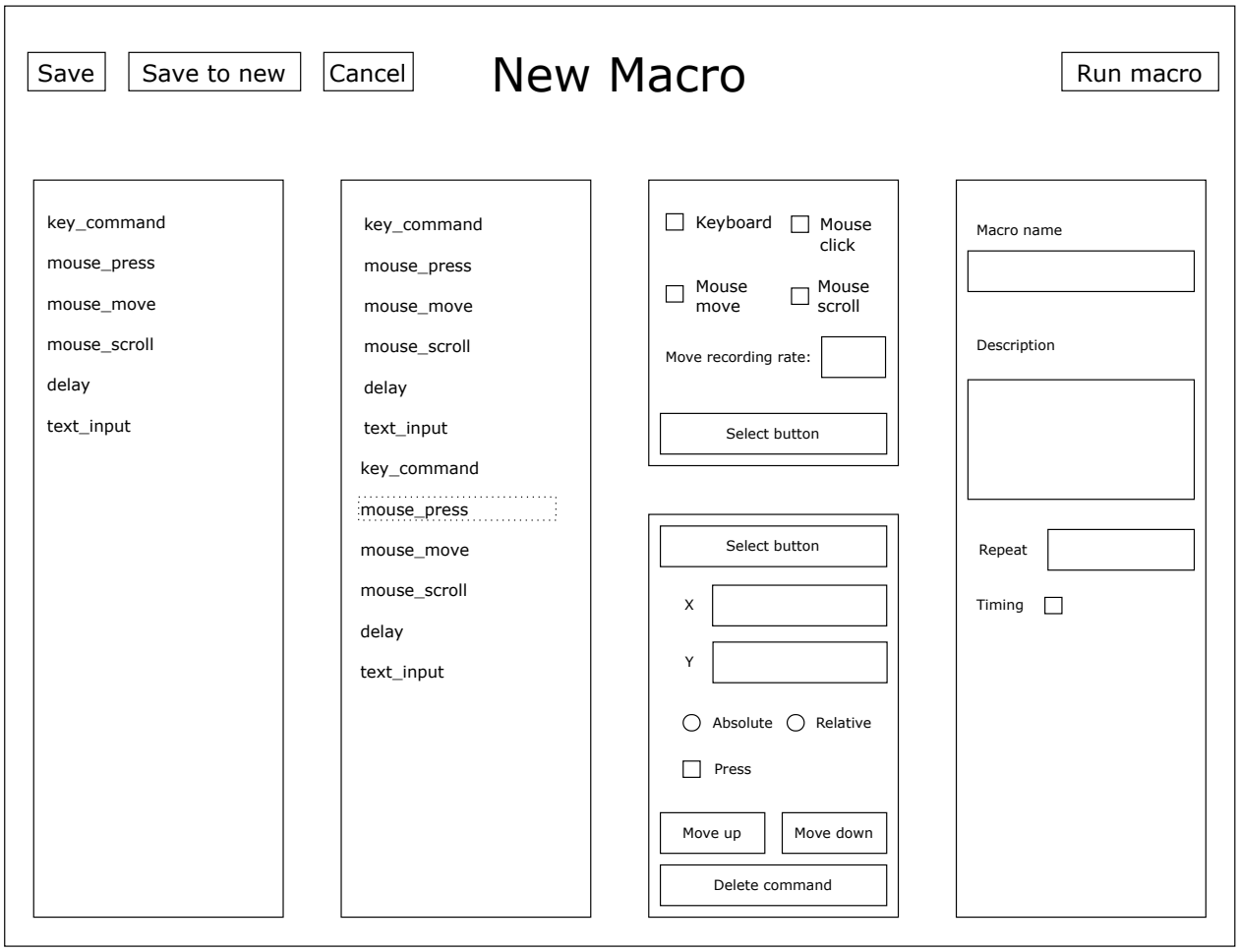
Součástí příkazu pro rolování myši budou atributy určující počet kroků pro horizontální a vertikální rolování. Příkaz pro pohyb myši bude obsahovat atributy pro novou pozici myši v osách x a y. Také bude obsahovat informaci o tom, zda se má myš pohybovat relativně k aktuální pozici (například pohyb myši o 10 pixelů doprava), nebo absolutně – zda se má kurzor na zadanou pozici přesunout. Příkaz pro stisk tlačítka myši bude obsahovat stejné atributy, jako příkaz pro pohyb myši, navíc ještě atribut pro určení tlačítka myši, které se má stisknout, a informaci o tom, zda se má tlačítko stisknout, nebo uvolnit.

3.4 Administrativní rozhraní

Administrativní rozhraní bude obsahovat seznam všech dostupných maker. Pro každé makro bude zobrazen název, popis a tlačítka pro editaci a spuštění (případně zastavení) makra. v horní části rozhraní bude prostor pro ostatní tlačítka. Jedním z nich bude tlačítko pro změnu zapnutí editačního režimu, ve kterém bude možno tlačítka přesouvat; v tomto režimu půjde také makra mazat. Pro návrat z editačního režimu budou sloužit tlačítka pro uložení a zrušení provedených změn. Dále se zde bude nacházet tlačítko pro přidání nového makra, po jehož stisknutí se zobrazí editor.

Návrh grafického uživatelského rozhraní editoru je zobrazen na obrázku 3.3. Editor bude rozdělen do horní části, určené pro nadpis a tlačítka, a do hlavní části se čtyřmi sloupci. v horní části budou

tlačítka pro spuštění (případně zastavení) makra, zrušení a uložení změn. Tlačítka pro uložení změn budou dvě: jedno pro uložení změn pro dané makro, druhé pro uložení změn do nového makra, které se tímto vytvoří. Při ukládání makra bude také zajištěna unikátnost názvu.



Obrázek 3.3: Návrh uživatelského rozhraní grafického editoru maker

V prvním sloupci bude zobrazen seznam všech dostupných příkazů. Po kliknutí na příkaz se přidá do sekvence příkazů, jež se zobrazuje ve druhém sloupci. Při výběru příkazu z této sekvence se zobrazí nastavení pro jeho atributy ve spodní části třetího sloupce. Pro každý příkaz zde budou navíc tlačítka pro smazání příkazu či jeho posunutí v sekvenci nahoru nebo dolů. Výběr klávesy a tlačítka myši bude možné provést tak, že uživatel klikne na tlačítko pro výběr klávesy / tlačítka a následně stiskne klávesu / tlačítko, které si přeje vybrat.

V horní části třetího sloupce bude nastavení pro nahrávání příkazů. To bude obsahovat přepínač pro výběr nahrávaných akcí – stisk kláves, pohyb myši, rolování myši, kliknutí myši. Dále bude možné nastavit, s jakou prodlevou se budou zaznamenávat akce pohybu myši. Nesmí chybět ani tlačítko pro zahájení a zastavení nahrávání. v posledním sloupci bude zobrazeno nastavení makra:

název, popis, počet opakování a přepínač pro časování příkazů. Tento přepínač také ovlivní, zda se v nastavení jednotlivých příkazů zobrazí pole s jejich časovou značkou.

3.5 Mobilní aplikace

Úvodní obrazovka mobilní aplikace bude obsahovat pole pro zadání IP adresy a portu serveru. Port nebude povinný – pokud nebude vyplněn, použije se výchozí hodnota. Po stisknutí tlačítka pro připojení se aplikace pokusí připojit k serveru. Pokud nebude zadán port, aplikace se pokusí připojit na výchozí port. Pokud se připojení nezdaří, pokusí se o připojení k 10 následujícím portům. Obdobné chování bude implementováno také na serveru – pokud vybraný port nebude k dispozici, pokusí se naslouchat na následujícím – takto nebude nutné, aby uživatel znal port, pokud bude využívat výchozí hodnotu, a to ani v případě, že výchozí port již bude obsazen. Po úspěšném připojení k serveru se uživateli zobrazí stav a aplikace bude čekat na uvítací zprávu. Jedná se buď o potvrzení úspěšného připojení, nebo o zprávu o zamítnutí připojení, například v případě zamítnutí při autorizaci.

V případě potvrzení úspěšného připojení aplikace požádá o seznam maker. Po jeho obdržení se získaná data zobrazí uživateli. Pro každé makro bude zobrazen název, popis a tlačítko pro spuštění či zastavení makra. v horní části aplikace bude tlačítko pro otevření menu, ve kterém bude možné zapnout editační režim, kde bude možné měnit uspořádání tlačítek pomocí funkce táhni a pusť. Pro návrat z tohoto režimu budou ve stejném menu sloužit tlačítka pro uložení či zahození změn. Dále bude v tomto menu tlačítko pro obnovení seznamu maker.

Při stisknutí tlačítka pro spuštění makra se odešle požadavek na server. Jako odpověď přijde informace o spuštění makra; až na základě této informaci se makro označí jako spuštěné. Obdobně se bude postupovat pro zastavení makra. Takto se zajišťuje, že se uživateli zobrazuje aktuální stav makra. Tento přístup také usnadňuje práci s více klienty. Pokud dojde k odpojení od serveru, zobrazí se znovu úvodní obrazovka a uživateli se zobrazí hláška o odpojení.

Po přijetí uvítací zprávy bude aplikace serveru periodicky posílat zprávy každých 5 sekund, aby se ujistila, že je stále připojena. Pokud server neodpoví, aplikace uživatele informuje o odpojení a zobrazí znovu úvodní obrazovku.

3.6 Serverová část

Po spuštění serveru se vyhodnotí parametry, s kterými byl uživatelem spuštěn. Pokud nebyla zadána adresa, naslouchá na všech dostupných rozhraních. Pokud nebyl zadán port, použije se výchozí hodnota. Pokud je port obsazen, server se pokusí využít daný počet následujících portů. Pokud se nepodaří najít volný port, na kterém by mohl naslouchat, s chybou se ukončí. Pokud je autorizace zadána jako povinná, bude při každém pokusu o připojení klienta vyžadováno, aby uživatel povolil či

zamítl připojení. Také bude mít možnost zablokovat dotyčného klienta. Ten se pak ze své IP adresy nebude moci připojit, dokud server nebude restartován.

Funkcionalitou serveru bude také nastavení firewallu. Pokud uživatel zadá parametr pro nastavení firewallu, server zkontroluje, zda je spuštěn s dostatečnými právy. Pokud ne, informuje uživatele a pokusí se restartovat s vyššími právy. Pokud dostatečná práva dostane, vytvoří nové, nebo upraví existující pravidlo tak, aby byl povolen přístup na daný port a na zadaný počet následujících portů. Uživatel dostane informaci o stavu nastavení firewallu.

Po připojení a případné autorizaci klienta server pošle uvítací zprávu. Poté si jej zařadí do seznamu klientů, jejichž požadavky bude vykonávat. Pokud klient požádá o seznam maker, server načte informace o všech makrech a odešle je klientovi. Pokud klient požádá o spuštění makra, server zjistí, zda je již nějaké makro spuštěné. Pokud ano, informuje o tom klienta, v opčaném případě makro spustí a oznámí všem klientům, že dané makro bylo spuštěno. Obdobně se bude postupovat při požadavku pro zastavení makra – pokud je makro spuštěné, bude zastaveno a všichni klienti o tom budou informováni.

Při požadavku na změnu uspořádání tlačítek se zkontrolují přijatá data. Pokud jsou validní, změny se provedou a zbývajícím klientům se odešle informace o změně s novým uspořádáním tlačítek. Pokud data nejsou validní, server odesílá informaci o chybě.

Server také bude zachytávat periodické zprávy od klientů. Při přijetí zprávy pošle zpět potvrzení a zároveň aktualizuje čas posledního přijetí zprávy od daného klienta. Pokud nedorazí zpráva od klienta po dobu 15 sekund, server klienta odpojí a odstraní jej ze seznamu klientů.

3.7 Komunikace serveru s klientem

Zde si popíšeme, jak bude vypadat komunikace mezi serverem a klientem. Komunikace bude realizovaná pomocí TCP socketů. Zprávy budou ve formátu JSON, budou kódovány ve formátu UTF-8 a budou ukončeny znakem pro konec řádku. Zprávy se budou skládat ze dvou částí. První část bude typ zprávy, druhá část bude obsahovat data. Server s klientem si budou vyměňovat následující zprávy: uvítací zpráva, žádost o seznam maker, seznam maker, žádost o spuštění makra, zpráva o spuštění makra, zpráva o tom, že jiné makro je již spuštěno, žádost o zastavení makra, zpráva o zastavení makra, žádost o změnu uspořádání tlačítek, zpráva o změně uspořádání tlačítek, zpráva o chybě a periodický signál.

Kapitola 4

Implementace řešení

V této kapitole se zaměříme na implementaci navrženého řešení. Nejdříve se podíváme na použité technologie a poté se zaměříme na implementaci jednotlivých částí řešení.

4.1 Použité technologie

Administrativní rozhraní a serverová část jsou napsány v jazyce `Python`. Pro implementaci administrativního rozhraní byla zvolena knihovna `PyQt6`, která poskytuje bohaté možnosti pro vytváření grafických uživatelských rozhraní. Serverová část komunikuje s klienty za využití knihovny `socket`, která poskytuje nízkoúrovňové rozhraní pro síťovou komunikaci. Pro serializaci a deserializaci dat byla zvolena knihovna `json`, která poskytuje jednoduché rozhraní pro práci s JSON formátem. Práci s vlákny zajišťuje knihovna `threading`. Pro snadnější práci s parametry byla zvolena knihovna `Click`. Nahrávání vstupů uživatele v grafickém editoru a jejich spouštění při přehrávání makra zajišťuje knihovna `pynput`.

Mobilní aplikace byla vytvořena v jazyce Java za využití `Android SDK`. Využity byly standardní knihovny – pro komunikaci se serverem `java.net`, pro práci s JSON formátem `org.json` a pro práci s vlákny `java.lang.Thread`. Pro zobrazení grafického uživatelského rozhraní byly využity standardní komponenty Androidu a materiálového designu.

4.1.1 Python

`Python` je vysokoúrovňový, interpretovaný, objektově orientovaný, dynamicky typovaný programovací jazyk. Je jednoduchý na použití a jeho jednoduchá syntaxe usnadňuje čtení i psaní kódu, což jej činí vhodným pro začátečníky, ale zároveň poskytuje dostatečné možnosti pro pokročilé uživatele. `Python` je multiplatformní a podporuje většinu operačních systémů. [8]

4.1.2 PyQt6

`PyQt6` je kompletní sada nástrojů pro vytváření grafických uživatelských rozhraní v jazyce `Python`. Poskytuje širokou škálu předdefinovaných widgetů a funkcí pro snadnou tvorbu a správu rozhraní. `PyQt6` je postaveno na `C++` knihovně `Qt6`, která je multiplatformní a podporuje tvorbu aplikací pro všechny hlavní operační systémy. [9]

Tato knihovna byla zvolena proto, že poskytuje bohaté možnosti, je velmi dobře zdokumentována a je široce používána. Díky tomu je možné najít mnoho návodů a příkladů, které usnadňují práci s touto knihovnou. Oproti jiným knihovnám, jako je například `Tkinter`, poskytuje `PyQt6` více možností. Nevýhodou však je velikost této knihovny a její náročnost na systémové prostředky.

4.1.3 Pynput

`Pynput` je knihovna, která se specializuje na práci s klávesnicí a myší. Poskytuje možnost zachytávání vstupů uživatele i jejich simulaci. Tato knihovna byla zvolena pro snadnou práci se vstupy uživatele v grafickém editoru a pro simulaci vstupů při přehrávání makra. Jedná se o jednoduchou a přehlednou knihovnu, která poskytuje všechny potřebné funkce pro práci s vstupy uživatele.[10]

Existují také alternativní knihovny, jako je například `PyAutoGui`, které poskytují podobné funkce. `PyAutoGui` se však spíše zaměřuje na simulaci vstupů a práci s okny a obrazovkou. Nenabízí jednoduché rozhraní pro zachytávání vstupů uživatele, na rozdíl od knihovny `pynput`, která umožňuje naslouchat vstupům uživatele a v reakci na ně spouštět různé funkce.[11]

4.2 Administrativní rozhraní

Administrativní rozhraní se skládá z hlavního okna, ve kterém jsou zobrazeny všechna dostupná makra, a z okna pro grafický editor, ve kterém je možné jednotlivá makra editovat.

4.2.1 Hlavní okno

Hlavní okno administrativního rozhraní obsahuje horní lištu ve formě horizontálního boxu (`QHbox`), ve kterém jsou umístěny tlačítka pro přidání nového makra, pro zapnutí editačního režimu a pro obnovení maker ze souborů. Tlačítka jsou realizovaná pomocí `PyQt6` třídy `QPushButton`, která poskytuje možnost vytvoření tlačítka s textem a funkcí spouštěnou po kliknutí na tlačítko. v horní liště je také umístěn název aplikace.

Pod horní lištou je umístěn seznam všech dostupných maker. Tento seznam je realizován pomocí kombinace rozložení `QGridLayout` a `QScrollArea`. `QGridLayout` poskytuje možnost vytvoření mřížky, ve které jsou umístěny jednotlivé prvky. `QScrollArea` je pak zodpovědná za zobrazení obsahu v případě, že je obsah větší než okno. Makra jsou zobrazena jako čtverce v této mřížce, kde každý čtverec obsahuje název název, popisek a tlačítka.

Pro zobrazení jednotlivých maker byl vytvořen vlastní widget, který obsahuje název makra, popisek a tlačítka pro editaci, smazání a spuštění makra. Tento widget je vytvořen pomocí třídy `QWidget`. Jednotlivé prvky jsou umístěny v mřížce – název makra je umístěn uprostřed nahoře, pod ním se nachází popisek a v dolních rozích jsou tlačítka. Ta jsou vytvořena pomocí třídy `QToolButton`, která poskytuje možnost vytvoření tlačítka s vlastní ikonou.

V levém dolním rohu je tlačítko pro editaci makra, to je připojeno na funkci pro zobrazení grafického editoru s daným makrem. v pravém dolním rohu je druhé tlačítko. Jeho funkce a vzhled se mění podle stavu aplikace a makra. Pokud je zapnut editační režim, pomocí tohoto tlačítka je možné makro smazat. Pokud je editační režim vypnutý, pomocí tohoto tlačítka je možné makro spustit. Po spuštění se tlačítko změní na tlačítko pro zastavení makra, do předchozího stavu se vrátí po stisknutí tlačítka či po dokončení běhu makra.

Při zapnutí aplikace jsou makra načtena ze souborů a seřazena podle jejich pozice, pro každé se vytvoří instance vlastního widgetu, spočítá se, kolik maker se vejde na řádek, a podle toho se nastaví mřížka i pozice maker v této mřížce. Po stisku tlačítka pro vytvoření nového makra se vytvoří nové makro s výchozími hodnotami, vytvoří se pro něj nový widget a otevře se grafický editor.

Po stisku tlačítka pro editační režim se aplikace přepne do režimu editace, kde je možné měnit pozice maker a mazat je. Makra lze mazat pomocí tlačítka pro smazání, které jsme si výše popsali. Měnit pozice maker lze pomocí funkce táhni a pusť, kdy je možné kliknout na makro a přesunout ho na jinou pozici, tímto se makra přeuspořádají. v editačním režimu se také mění tlačítka v horní části okna – místo tlačítka pro zapnutí editačního režimu jsou nyní zobrazena dvě tlačítka, jedno pro uložení změn, druhé pro jejich zrušení. Při stisku tlačítka pro uložení se všechny změny zapíší do souborů – pro změnu pozice makra se aktualizuje atribut pozice v souboru, pro smazání makra se odstraní soubory daného makra. Při stisku tlačítka pro zrušení změn se všechny změny zahodí a aplikace se vrátí do původního stavu.

4.2.2 Grafický editor

Grafický editor slouží k editaci jednotlivých maker. Je rozdělen na dvě části – horní část s tlačítky pro uložení makra, uložení do nového makra, zrušení změn a pro spuštění makra, a dolní část s vlastním editorem.

Horní část je, obdobně jako v případě hlavního okna, vytvořena pomocí horizontálního rozvržení. Spodní část je realizovaná pomocí čtyř sloupců (pomocí tříd `QHbox` a `QVBox`). v prvním sloupci se nachází seznam všech dostupných příkazů. Tento seznam využívá třídy `QListWidget`, která poskytuje možnost vytvoření seznamu s možností výběru. v druhém sloupci je pak pomocí dalšího seznamu zobrazena sekvence všech příkazů daného makra. Třetí sloupec je rozdělen na dvě části pomocí rozvržení `QVBox` – v horní části jsou zobrazeny možnosti týkající se nahrávání příkazů, v dolní části jsou pak možnosti týkající se vybraného příkazu ze sekvence. v posledním sloupci se nachází obecné nastavení makra, jako je název, popisek, počet opakování a přepínání časování.

Při otevření editoru se načtou hodnoty z daného makra a zobrazí se v jednotlivých polích. Původní třída makra je zachována, pro editované makro je vytvořena nová instance. Po stisku tlačítka pro uložení se zkontroluje, zda, pokud byl makru změněn název, již neexistuje makro se stejným názvem. Pokud ano, bude zajištěna unikátnost pomocí přidání čísla za název (to se opakuje, dokud není název unikátní). Poté se uloží změna do souboru, aktualizuje se mřížka maker v hlavním okně a editor se vypne. Pokud bylo stisknuto tlačítko pro uložení do nového makra, průběh je obdobný, jen s tím rozdílem, že původní makro zůstane nezměněno a bude vytvořeno nové makro. Pokud bylo stisknuto tlačítko pro zrušení změn, editor se vypne a změny se zahodí. Tlačítko pro spuštění makra funguje obdobně jako v hlavním okně.

Při výběru příkazu ze seznamu příkazů se vybraný příkaz přidá do sekvence příkazů za aktuálně vybraný příkaz. Po přidání bude zároveň vybrán a ve třetím sloupci se zobrazí možnosti týkající se tohoto příkazu. Použity jsou třídy `QLineEdit` pro textové pole, QSpinBox pro číselné pole, `QCheckBox` pro zaškrtávací pole a `QRadioButton` pro tlačítka pro výběr. Po změně hodnoty se změna uloží do příkazu a změny se projeví v názvu příkazu v sekvenci.

Nahrávání příkazů je realizováno pomocí knihovny `pynput`, konktrétně pomocí třídy `Listener`. Při stisku tlačítka pro nahrávání se spustí nové vlákno, které naslouchá vstupům uživatele. Zaznamenávají se stisknuté klávesy, pohyby myši, kliknutí myší a rolování myší nezávisle na sobě podle toho, které možnosti uživatel chce nahrávat (zvolí pomocí čtyřech zaškrtávacích polí). Po zaznamenání vstupu se vytvoří nový příkaz (za aktuálně vybraným) a zobrazí se v sekvenci. Při stisku tlačítka pro zastavení nahrávání nebo po stisku klávesy escape se nahrávání zastaví. Uživatel také nastaví, jak často se má zaznamenávat pohyb myši. To je zajištěno aktualizováním časové značky při každé získané události o pohybu myši a porovnáním s předchozí značkou.

4.3 Makro a příkazy

Makro je reprezentováno třídou Macro, která obsahuje název, popisek, sekvenci příkazů, počet opakování a přepínač časování. Třída Macro obsahuje metody potřebné pro spuštění či zastavení makra, pro získání a nastavení atributů a pro manipulaci se souborem makra – uložení, načtení, smazání.

Spuštění makra můžeme vidět ve výpisu zrdojového kódu 1 a probíhá následovně. Nejprve se zkontroluje, zda makro obsahuje nějaké příkazy. Pokud ne, pokusí se je načíst ze souboru – příkazy a informace o makru jsou uloženy ve dvou různých souborech, aby nebylo nutné načítat příkazy, dokud není makro spuštěno či editováno. Pokud se příkazy nepodaří načíst, funkce skončí a vrátí chybu. v opačném případě se vytvoří instance třídy `Controller` z knihovny `pynput` pro myš i klávesnici. Následuje cyklus přes počet opakování, na jehož začátku se nastaví počáteční čas na aktuální, a vnitřní cyklus přes všechny příkazy, ve kterém se nejdříve zkontroluje, zda je nastaveno časování příkazů. Pokud ano, zjistí rozdíl aktuálního času makra (rozdíl mezi aktuálním a počátečním) a času následujícího příkazu, program se pak na tuto dobu uspí (pokud je hodnota

času následujícího příkazu větší než aktuálního času). Čas se získává pomocí funkce `perf_count` knihovny `time`. Dále se volá metoda execute daného příkazu, které se podle jeho typu předá jako parametr příslušná instance kontroleru. Tato metoda pro spuštění makra vrací stav, zda bylo makro úspěšně dokončeno, zastaveno, nebo zda došlo k chybě.

Zastavení vykonávání makra je realizováno pomocí proměnné exit, která se zavoláním funkce stop nastaví na hodnotu `true`. Tato proměnná je kontrolována na začátku každého cyklu a po každé prodlevě mezi příkazy. Pokud je proměnná exit nastavena na `true`, makro se zastaví a funkce skončí.

Také bylo potřeba vyřešit, aby po sobě makro nezanechávalo stisknuté klávesy či tlačítka myši. To je zajištěno vytvořením množiny všech stisknutých kláves a tlačítek. Při stisknutí se do této množiny přidává daný stisknutý prvek, při uvolnění se odstraní. Při skončení (i zastavení) makra se pak tyto prvky uvolní.

Příkazy jsou, jak bylo navrženo v sekci 3.3 na straně 20, reprezentovány třídou `Command`, která obsahuje název, čas a virtuální metodu execute, jež potomky implementují.

```
    def execute(self):
        self.exit = False
        if not self.commands:
            if not self.load_commands():
                return "error"

        keyboard_controller = pynput.keyboard.Controller()
        mouse_controller = pynput.mouse.Controller()
        for i in range(self.repeat):
            start_time = time.perf_counter()
            for command in self.commands:
                if self.exit:
                    break
                if self.timing:  # Čekání po rozdíl časů, pokud je časování zapnuto
                    total_time = time.perf_counter() - start_time
                    if command.time > total_time:
                        time.sleep(command.time - total_time)
                    if self.exit:
                        break
                # Funkce pro zpracování příkazu podle jeho typu
                self.dispatch_command(command, keyboard_controller, mouse_controller)

        self.release_all_keys(keyboard_controller)  # Uvolnění všech stisknutých kláves
        self.release_all_btns(mouse_controller)
        return "success" if not self.exit else "stopped"
```

Výpis 1: Funkce pro spuštění makra

4.4 Serverová část

Při spuštění serveru se nejdříve zpracují argumenty předané z příkazové řádky, to zajišťuje knihovna `Click`. Pokud je zadán argument pro konfiguraci firewallu, provede se jeho nastavení. v opačném případě se vytvoří instance třídy `Server`, která obsahuje metody pro příjem a odesílání zpráv, pro zpracování příchozích spojení, provádění příkazů a pro monitorování stavu klientů, a zavolá se její metoda run s ostatními argumenty.

Metoda `run` vytvoří a spustí vlákno pro monitoring klientů využívající metody `heartbeat_monitor`, jež ve smyčce kontroluje, zda některý z klientů překročil stanovený časový limit bez poslání `heartbeat` zprávy. Pokud ano, klient je označen jako neaktivní a je odpojen. Dále se vytvoří instance třídy `socket`, která zajišťuje připojení klientů, a nastaví se na naslouchání na dané adrese a portu. Pokud je vybraný port již obsazen, program se pokusí nastavit následující port. v případě, že není možné naslouchat na žádném z následujících portů (počet definován parametrem `max_attempts`), program skončí s chybou.

Server využívá kombinaci knihoven `socket` a `selectors` pro neblokující naslouchání na socketu a zpracování příchozích spojení a zpráv. Po úspěšném nastavení socketu na naslouchání se vytvoří instance třídy `DefaultSelector`, která zajišťuje výběr socketu, který je připraven k čtení, a zpracování příchozích spojení. Socket se přepne do neblokujícího režimu a zaregistruje se ve vytvořeném selektoru. Následně se v nekonečné smyčce čeká na příchozí zprávy.

Pro zpracování příchozích spojení slouží metoda `accept`, která přijme příchozí spojení, zkontroluje, zda se IP adresa klienta nenachází mezi blokovanými adresami (pokud ano, odešle se zpráva o odmítnutí spojení a socket se uzavře), pokud je zapnut režim autorizace, nabídne se uživateli, zda klienta přijmout, odmítnout či zablokovat. Když je klient přijat, vytvoří se instance třídy `SocketClient` s informacemi o klientovi a jeho socketu a přidá se do seznamu aktivních klientů. Jeho socket je rovněž nastaven do neblokujícího režimu a zaregistrován v selektoru.

Zpracování příchozích zpráv od klientů zajišťuje funkce `handle_socket_message`. Po přijetí dat ze socketu se data přidají do bufferu klienta. Buffer představuje pole bytů. Využívá se proto, že data mohou přijít po částech a je nutné je ukládat do bufferu, dokud není zpráva kompletní. Až bude zpráva kompletní – to se pozná podle ukončovacího znaku (znaku konce řádku) – bude zpracována a v bufferu zůstane případný zbytek zprávy. Zpráva je deserializována pomocí knihovny `json` a její funkce `loads`. Následně se podle typu zprávy provede příslušná akce. Část kódu pro zpracování zprávy od klienta můžeme vidět v ukázce zdrojového kódu 2.

Odesílání zpráv klientům zajišťují metody `send_message` a `send_message_to_all`. První zmíněná přijme socket, typ zprávy a data, která se mají odeslat. Zprávu nejprve serializuje tak, že ji uloží do slovníku, tento slovník serializuje a enkóduje do formátu UTF-8 pomocí knihovny `json`. K tomuto zakódovanému řetězci se přidá ukončovací znak a odešle se klientovi. Kód této funkce je uveden ve výpisu zrdojového kódu 3. Druhá metoda přijme typ zprávy, data a nepovinně také

```
while b'\n' in client.buffer:
    message, client.buffer = client.buffer.split(b'\n', 1)
    msg_dict = json.loads(message)
    self.handle_message(msg_dict, client)
```

Výpis 2: Část funkce pro zpracování zprávy od klienta

klienta, kterému zprávu neposlat. Postupně se projde seznamem všech připojených klientů, získá se jejich socket a zavolá se metoda send_message.

```
def send_message(self, sock, msg_type, msg_data):
    message = {'type': msg_type, 'data': msg_data}
    enc_message = json.dumps(message).encode() + b'\n'
    sock.sendall(enc_message)
```

Výpis 3: Funkce pro odeslání zprávy klientovi

4.5 Mobilní aplikace

Mobilní aplikace se skládá ze dvou aktivit – hlavní aktivity, která se stará o zadání IP adresy a portu server a připojení k němu, a aktivity pro zobrazení a ovládání jednotlivých maker – a ze služby zajišťující komunikaci se serverem. Obě aktivity jsou vytvořeny pomocí třídy `AppCompatActivity`, která poskytuje základní funkce pro vytváření a správu aktivit.

Hlavní aktivita obsahuje textové pole pro zadání IP adresy a portu serveru, tlačítko pro připojení k serveru a tlačítko pro zobrazení nastavení. Po stisknutí tlačítka pro nastavení se zobrazí menu s možností přepínání mezi normálním a černobílým režimem. Tato možnost se při změně projeví a uloží do sdílených preferencí, které jsou dostupné v celé aplikaci. Hned při vytvoření aktivity se zároveň vytvoří instance služby `SocketService` a zavolá se funkce `bindService`, která zajistí připojení k této službě.

Při stisknutí tlačítka pro připojení se zkontroluje, zda byla zadána IP adresa a port, nebo pouze IP adresa. Pokud nebyl zadán port, využije se výchozí hodnota. Následně se zavolá metoda `connect` třídy `SocketService`, která vytvoří nové vlákno pro připojení k serveru. v tomto vlákně se vytvoří instance třídy `Socket`, která zajišťuje komunikaci se serverem. Pokud se podaří připojit, vytvoří se vlákno pro periodické posílání heartbeat zprávy a další vlákno pro naslouchání na příchozí zprávy – toto vlákno se rovnou spouští.

Přijímání zpráv probíhá obdobně jako v případě serverové části. Po úspěšném přijetí a deserializaci zprávy je nutno předat zprávu hlavní aktivitě, kde se zpracuje. To je zajištěno pomocí zpětného volání, kdy se v hlavní aktivitě vytvoří instance rozhraní `Consumer`, která implementuje

metodu `accept`. Tato instance se předá do služby pro komunikaci, kde je po přijetí zprávy zavolána metoda `accept` s přijatou zprávou jako argumentem. Jelikož je tato metoda volána v jiném vlákně, je nutné zajistit, aby byla zpráva zpracována v hlavním vlákně – to je zajištěno pomocí handleru. Část procesu načtení (po přečtení znaku konce řádku) a předání zprávy můžeme vidět v ukázce kódu 4.

```
String msgType;
String msgData;
JSONObject msg;
try {
    String message = stringBuilder.toString(); // zpráva ze socketu bez '\n'
    msg = new JSONObject(message);
    msgType = msg.getString("type");
    msgData = msg.getString("data");
} catch (Exception e) {
    Log.e("Socket service error", e.getMessage());
    continue;
}
// ...
if (this.handler == null) { // vytvoření handleru, pokud neexistuje
    this.handler = new Handler(Looper.getMainLooper());
}
handler.post(() -> {
    for (Consumer<JSONObject> callback : this.callbacks) {
        callback.accept(msg); // volání funkce accept
    }
});
```

Výpis 4: Předávání zpráv hlavnímu vláknu pomocí handleru

Po výměně uvítací zprávy, žádosti o makra a přijetí seznamu maker se vytvoří nová aktivita, jíž se předají získaná data. Při jejím vytvoření se seznam maker deserializuje a pro každé makro se vytvoří instance třídy `Macro`, která obsahuje informace o makru a zpětná volání pro spuštění a zastavení makra. Všechna makra se přidají do seznamu, který se následně setřídí podle jejich pozice. Následně se nastaví adaptér pro zobrazení maker v `RecyclerView`, jenž zajišťuje zobrazení jednotlivých prvků seznamu. o vzhled a chování jednotlivých prvků se stará třída `CustomMacoViewHolder`, která obsahuje prvky grafického rozhraní a instanci třídy Macro.

Při nečekaném odpojení od serveru služba `SocketService` pomocí zpětného volání informuje aktivity o odpojení. Dojde k ukončení aktivity pro zobrazení maker a zobrazí se opět hlavní aktivita, kde se uživatel může pokusit znovu připojit. Při ukončení aplikace se zavolá metoda `disconnect` třídy `SocketService`, která ukončí všechna vlákna a uzavře socket.

Kapitola 5

Zhodnocení

Požadavky práce byly splněny, implementace byla úspěšně dokončena a je funkční, nicméně existují omezení a možná vylepšení, která by bylo vhodné do budoucna implementovat. Podíváme se na některá z nich.

5.1 Omezení

Ačkoliv je možné přenést změnu organizace tlačítek z mobilní aplikace do administrativního rozhraní a naopak, změny se projeví ihned pouze ostatním klientům při změně organizace klientem. V případě provedení změny klientem se tyto změny v administrativním rozhraní projeví až při opětovném načtení ze souboru. Stejně tak změny v administrativním rozhraní se projeví až po opětovném načtení seznamu maker v aplikaci.

Toto omezení je způsobeno tím, že administrativní rozhraní nekomunikuje se serverem. Toto by bylo možné elegantně vyřešit pomocí vzdáleného volání procedur (RPC) nebo jiného způsobu komunikace mezi aplikacemi. Bylo by potřeba zajistit, aby se aplikace se serverem spojila nezávisle na tom, který program byl spuštěn jako první. To by se dalo vyřešit například implementací klientského i serverového chování v obou programech.

Řešení pomocí RPC bylo v této práci vyzkoušeno pomocí knihovny `rpyc`, nicméně se objevily problémy s kompatibilitou s knihovnou `PyQt6`. Povedlo se spustit funkci programu, ale už se nepovedlo předat hlavnímu vláknu instrukci k aktualizaci grafického rozhraní pomocí signálu a slotu ani za pomoci `QTimer` ani fronty zpráv.

5.2 Možná vylepšení

Mezi možná vylepšení patří například širší škála příkazů. Takovým příkazem by mohlo být například vyhledání určitého obrázku na obrazovce a kliknutí na něj. Mezi další možné příkazy patří například

získání informací o okně pod kurzorem či o okně podle názvu, získání informací o procesech, příkazy pro práci s médii či se soubory – například pro automatizaci zálohování nebo synchronizaci souborů.

Dalším vylepšením by mohlo být řízení toku programu pomocí podmínek a cyklů. To by umožnilo vytvořit složitější scénáře, kde by makra byly schopny dynamicky reagovat na různé situace a v závislosti na nich provádět různé akce. K tomuto by bylo vhodné vytvořit adekvátní editor. Ten by mohl být založen na principu blokového programování, kde by uživatel mohl jednoduše skládat bloky s příkazy do sekvencí a cyklů pomocí funkce táhni a pusť.

Také by bylo možno vytvořit funkci pro přepočítání časových značek příkazů na příkazy prodlevy a naopak. Spouštění a zastavení nahrávání vstupů by mohlo mít přiřazenou klávesovou zkratku, aby uživatel nemusel vždy ovládat nahrávání myší. Taktéž by bylo vhodné přidat možnost krátké časové prodlevy mezi stiskem tlačítka nahrávání a zapnutím nahrávání, aby bylo možné stihnout dosáhnout požadovaného počátečního stavu, například otevření okna nebo pozice kurzoru. Dále by bylo možné implementovat možnost zastavení běžícího makra pomocí klávesové zkratky – to by bylo užitečné obzvláště v případě, kdy by uživatel vytvořil makro tak, že by jej nebyl kvůli nastaveným pohybům myši schopen vypnout a bylo by nutné počkat, až se makro dokončí.

Nastavení serverové části by mohlo být přesunuto do konfiguračního souboru, aby nebylo nutné vždy zadávat parametry při spuštění. Součástí konfigurace by mohl být i seznam povolených IP adres, ze kterých je možné se připojit přímo bez nutnosti manuálního povolení uživatelem i se zapnutým režimem pro ověřování klientů. Taktéž by bylo možné implementovat seznam zakázaných IP adres, které by byly automaticky blokovány.

Komunikace mezi aplikacemi by mohla být zabezpečena pomocí šifrování, aby se zabránilo odposlechu a zneužití dat. Užitečné by rovněž bylo implementovat možnost správy připojených klientů na serverové části nebo také v administrativním rozhraní, aby bylo možné sledovat, kteří klienti jsou připojeni, případně klienta odpojit či opět povolit jeho připojení po zablokování.

Kapitola 6

Závěr

Tato práce popsala problematiku automatizace činností na počítači pomocí maker. Práce čtenáře dále seznámila s existujícími řešeními, jejich možnostmi, výhodami a nevýhodami. Na základě této analýzy byly navrženy požadavky na vlastní řešení, které by mělo být schopno nahradit existující řešení a přinést nové možnosti.

V rámci práce byla vytvořena mobilní aplikace pro operační systém Android, která umožňuje spuštět, zastavovat a přeorganizovávat makra na počítači. Aplikace byla navržena tak, aby byla jednoduchá na použití a intuitivní. Dále byla vytvořena serverová část, která zajišťuje komunikaci mezi počítačem a mobilní aplikací. Serverová část byla navržena tak, aby byla jednoduchá na nasazení a použití.

Součástí práce bylo také vytvoření administrativního rozhraní, které umožňuje správu maker a jejich organizaci. Administrativní rozhraní bylo navrženo tak, aby bylo jeho použití bylo intuitivní a přehledné.

Výsledné řešení bylo úspěšně implementováno a otestováno malou skupinou testerů. Bylo ověřeno, že aplikace je schopna spouštět, zastavovat a přeorganizovávat makra na počítači. Dále bylo ověřeno, že administrativní rozhraní je schopno spravovat makra a jejich organizaci. K serverové části může být zároveň připojeno více klientů, kteří mohou společně pracovat s makry. Všechny činnosti klientů jsou synchronizovány a zobrazeny v reálném čase u všech připojených klientů.

Na závěr bylo provedeno zhodnocení výsledného řešení a navržena možná vylepšení a rozšíření. Mezi možná vylepšení patří například širší škála příkazů, řízení toku programu, lepší grafický editor, šifrování komunikace a možnost správy připojených klientů.

Lze tedy konstatovat, že cíle práce byly splněny a výsledné řešení je funkční.

Literatura

1. IBM. *What is automation?* [online]. [cit. 2024-04-27]. Dostupné z: `https://www.ibm.com/topics/automation`.
2. PICKLE, Brian. *Macro Definition* [online]. 2023-03-02. [cit. 2024-04-27]. Dostupné z: `https://whatis.techtarget.com/definition/macro`.
3. PICKLE, Brian. *Script Definition* [online]. 2023-02-28. [cit. 2024-04-27]. Dostupné z: `https://whatis.techtarget.com/definition/script`.
4. *AutoHotkey* [online]. [cit. 2024-04-27]. Dostupné z: `https://www.autohotkey.com/`.
5. ELGATO. *Welcome to Stream Deck* [online]. 2024. [cit. 2024-04-27]. Dostupné z: `https://www.elgato.com/uk/en/s/welcome-to-stream-deck`.
6. *Stream Deck MK.2* [online]. [cit. 2024-04-27]. Dostupné z: `https://www.elgato.com/us/en/p/stream-deck-mk2-black`.
7. *AutoIt* [online]. AutoIt Consulting Ltd, 2024. [cit. 2024-04-27]. Dostupné z: `https://www.autoitscript.com/site/`.
8. SUMMERFIELD, Mark. *Programming in Python 3*. 2. vyd. Addison-Wesley Professional, 2009. ISBN 978-0-321-68056-3.
9. *What is PyQt?* [online]. Riverbank Computing. [cit. 2024-04-27]. Dostupné z: `https://www.riverbankcomputing.com/software/pyqt/`.
10. MOSES-PALMER. *pynput GitHub Repository*. 2023-07-11. Dostupné také z: `https://github.com/moses-palmer/pynput`.
11. SWEIGART, Al. *pyautogui GitHub Repository*. 2023-07-07. Dostupné také z: `https://github.com/asweigart/pyautogui`.

Příloha A

Návod na instalaci

Serverová část a administrativní rozhraní se nachází v adresáři `MacroServer`, pro jejich spuštění je potřeba mít nainstalovaný Python 3.10 nebo novější. Dále je potřeba nainstalovat potřebné knihovny ze souboru `requirements.txt`. I když to není potřeba, je doporučeno vytvořit virtuální prostředí a nainstalovat knihovny do něj. Instalační postup pro operační systém Windows je následující:

1. `python -m venv venv`
2. `venv\Scripts\activate`
3. `python -m pip install -r requirements.txt`

Pro spuštění serverové části je potřeba spustit příkaz `python macro_server.py`. Pro zobrazení nápovědy k serverové části je možné použít přepínač `--help`

Pro spuštění administrativního rozhraní je potřeba spustit příkaz `python macro_admin.py`.

Pro kompilaci a spuštění mobilní aplikace je potřeba mít nainstalované vývojové prostředí Android Studio a nainstalovaný emulátor nebo připojeno fyzické zařízení. Nejdříve je potřeba otevřít přiložený projekt, který se nachází v adresáři `MacroClient`, v Android Studiu a následně spustit aplikaci na emulátoru nebo fyzickém zařízení.

Přiložen je také soubor `macro_app.apk`, který je možno nainstalovat na mobilní zařízení s operačním systémem Android o verzi 8.1 nebo vyšší. Pro instalaci je potřeba povolit instalaci aplikací z neznámých zdrojů a následně spustit soubor `macro_app.apk`.