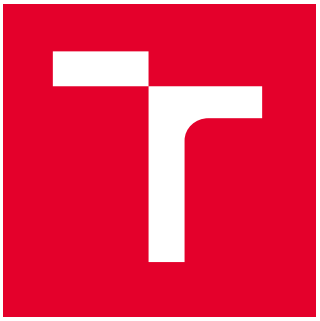


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

EMAILOVÝ KLIENT PRO SENIORY

EMAIL CLIENT FOR SENIORS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Robin Vala**

AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. Dan Komosný, Ph.D.**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Robin Vala ***ID:*** 241124

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Emailový klient pro seniory

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte základního emailového klienta, který bude přizpůsobený pro seniory ve věkové skupině 90 let a více. Emailový klient bude jednoduše ovladatelný a bude umožňovat spolupráci s opatrovníkem seniora. Implementujte ochranu proti odkazům v emailech na podvodné (phishing) stránky. Klienta zhotovte v programovacím jazyce Python. Výsledky práce publikujte na repozitáři GitHub pod licencí MIT.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: prof. Ing. Dan Komosný, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se věnuje návrhu a vývoji e-mailového klienta, který je přizpůsoben potřebám seniorů ve věku 90 let a starším. Hlavním cílem bylo vytvořit uživatelsky přívětivé rozhraní, které minimalizuje komplexitu a zahrnuje pouze nezbytné funkce relevantní pro tuto cílovou skupinu lidí. Navíc byla do aplikace integrována zvuková asistence, která usnadňuje navigaci v prostředí programu a zlepšuje celkovou uživatelskou zkušenost. Jako součást zabezpečení aplikace byla implementována funkce, která varuje uživatele před potenciálním nebezpečím spojeným s odkazy vedoucími na podvodné webové stránky obsažené v přijatých e-mailových zprávách. Toto opatření přispívá k ochraně uživatelů před online hrozbami a podvody. Kompletní zdrojový kód této bakalářské práce je volně dostupný v repozitáři na platformě Github.

KLÍČOVÁ SLOVA

E-mailový klient, IMAP, Phishing, Senior, SMTP, Sociální inženýrství

ABSTRACT

The bachelor thesis focuses on the design and development of an e-mail client that is adapted to the needs of seniors aged 90 years and older. The main goal was to create a user-friendly interface that minimizes complexity and includes only the necessary features relevant to this target group of people. In addition, audio assistance was integrated into the application, which significantly facilitates navigation in the program environment and improves the overall user experience. As part of the application's security, a special feature has been implemented to warn users of the potential dangers associated with links leading to fraudulent websites contained in received email messages. This measure helps to protect users from online threats and scams. The complete source code of this thesis is freely available in the Github repository.

KEYWORDS

Email Client, IMAP, Phishing, Senior, SMTP, Social Engineering

Vysázeno pomocí balíčku `thesis` verze 4.09; `https://latex.fekt.vut.cz/`

VALA, Robin. *Emailový klient pro seniory*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: prof. Ing. Dan Komosný, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Robin Vala
VUT ID autora:	241124
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Emailový klient pro seniory

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu prof. Ing. Danovi Komosnému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Cílem této bakalářské práce je vytvořit uživatelsky přívětivý a snadno ovladatelný e-mailový klient, který bude speciálně navržen pro potřeby seniorů ve věku 90 let a více. Při pohledu na dnes již existující e-mailové klienty je evidentní, že jejich složité funkce a rozhraní mohou být pro seniory matoucí a obtížně ovladatelné. Proto v rámci projektu *Senior-os* vznikl návrh na vytvoření e-mailového klienta, který by byl přizpůsoben a zjednodušen pro potřeby právě této specifické uživatelské skupiny. Cílem je vytvořit jednoduché a intuitivní rozhraní, zminimalizovat složité funkce a zaměřit se na základní operace, které mohou senioři potřebovat v rámci komunikace přes e-mail.

Teoretická část této práce začíná kapitolou 1, která se věnuje popisu e-mailové komunikace. První podkapitola 1.1 se věnuje formátům a skladbě e-mailové zprávy, v podkapitole 1.2 je popsán přenos e-mailové zprávy od odesílatele k příjemci. Následuje podkapitola 1.3 věnující se stručnému popisu systému DNS a doménových jmen.

Dále se zde nachází kapitola 2, která se zabývá problematikou sociálního inženýrství. V podkapitolách 2.1 a 2.2 jsou zkoumány fáze a typy útoků sociálního inženýrství, následuje podkapitola 2.3 o podvodných e-mailových zprávách. Poslední podkapitola 2.4 se věnuje sociálnímu inženýrství v České republice.

Kapitola 3 bakalářské práce se věnuje základním protokolům, které se používají pro práci s e-maily. Konkrétně se jedná o protokol *SMTP*, který slouží uživateli k odeslání pošty, protokol *IMAP* umožňující přistupovat ke schránce uživatele z libovolného zařízení a číst e-maily přímo z online schránky, protokol *POP* sloužící ke stažení pošty na zařízení uživatele a protokol *MIME* sloužící pro odesílání jiných než textových dat prostřednictvím e-mailu.

V závěru teoretické části jsou v kapitole 4 popsány moduly, které bylo potřeba implementovat do aplikace, aby bylo možné dosáhnout požadovaných výsledků.

Praktická část bakalářské práce popisuje vytvoření e-mailového klienta a jeho testování. V kapitole 5 je popsán postupný vývoj aplikace pomocí programovacího jazyku *Python*. Jednotlivé podkapitoly se věnují konkrétním částem vývoje. Podkapitola 5.1 se zabývá návrhem a implementací grafického uživatelského rozhraní. V sekci 5.2 je popsáno ovládání e-mailového klienta pomocí tlačítek. Následuje podkapitola 5.3 věnující se konfiguraci parametrů. Dále jsou v podkapitolách 5.4 a 5.5 popsána prostředí pro psaní a čtení e-mailových zpráv. Podkapitola 5.6 se věnuje tvorbě a implementaci hlasové asistence. Poslední dvě podkapitoly 5.7 a 5.8 se věnují ochraně před podvodnými e-mailovými zprávami a zaznamenávání vzniklých událostí.

Předposlední kapitola 6 se zabývá testováním vytvořeného e-mailového klienta. V podkapitole 6.1 jsou popsány jednotkové testy klíčových funkcí aplikace. Následuje část 6.2 věnující se testování načítání aplikace. Podkapitola 6.3 přibližuje testování uživatelů, včetně výsledků z dotazníkového šetření. Poslední dvě sekce této kapitoly 6.4 a 6.5 se věnují odladění aplikace na základě výsledků testování a návrhům na zlepšení.

V závěru bakalářské práce (kapitola 7) je popsáno zveřejnění výsledného kódu a postup při instalaci a spuštění aplikace.

1 E-mailová komunikace

E-mailová komunikace je základní služba, kterou lze najít v prostředí internetu a hraje klíčovou roli při výměně informací mezi jednotlivci či mezi různými organizacemi. Jedná se o prostředek, který umožňuje rychlý a efektivní způsob přenosu zpráv prostřednictvím elektronické pošty, bez ohledu na geografickou vzdálenost.

Existuje celá řada e-mailových platforem, které tuto službu poskytují. Jedním z nejrozšířenějších poskytovatelů je *Gmail*, který uživatelům nabízí širokou škálu funkcí a výhod. Kromě toho existují i další poskytovatelé, jako *Outlook* nebo *Seznam* a mnoho dalších, každý s vlastními výhodami.

Elektronická média, jako jsou e-maily, mění způsob, jakým lidé navzájem komunikují a výrazně se odlišují od klasické osobní komunikace. Tato média spojují písemné sdělování s rychlostí a dostupností, kterou nabízí dnešní mobilní telefony. Díky e-mailům máme možnost komunikovat prakticky kdykoli a odkudkoli, což nám umožňuje rychle a efektivně sdílet informace [1].

S těmito výhodami přicházejí i určité výzvy. Osobní komunikace často zahrnuje více nuancí a emocionálního vyjadřování, komunikace pomocí elektronických médií může někdy tuto hloubku ztrácet. Důraz na písemnou formu (na psané slovo bez osobní interakce), může vést ke zjednodušení komunikace a ke ztrátě jejího kontextu. Také může docházet k rozvoji povrchního stylu komunikace kvůli velkému množství elektronických zpráv a potřebě rychlých odpovědí [1].

Lze konstatovat, že elektronická média vytváří nový model komunikace, ve kterém je důležité najít rovnováhu mezi rychlostí a hloubkou vyjádření, aby komunikace byla efektivní a smysluplná [1].

1.1 E-mailová zpráva

E-mailová zpráva se skládá ze dvou hlavních částí – záhlaví a těla. Záhlaví obsahuje různá pole, která reprezentují informace o odesílateli a příjemci. Typicky jsou do záhlaví e-mailové zprávy zakomponována tato pole [2]:

- **předmět zprávy** – pole poskytující shrnutí obsahu e-mailové zprávy,
- **odesílatel** – pole pro označení e-mailové adresy odesílatele,
- **datum a čas přijetí** – pole vyplňované automaticky a povinně, poskytuje časové informace,
- **odpovědět** – pole sloužící pro reakci na přijatou zprávu,
- **příjemce** – pole pro zobrazení příjemce pošty,
- **kopie (CC)** – pole sloužící pro odeslání e-mailové zprávy lidem, od kterých se neočekává odpověď,

- **skrytá kopie (BCC)** – pole umožňující utajení informací o dalších příjemcích,
- **přílohy** – pole obsahující soubory přiložené k e-mailové zprávě.

Tělo e-mailu obsahuje samotný obsah zprávy, může se jednat o text, videa nebo přiložené soubory a mnoho dalšího. Formát těla e-mailu může být buď čistý jednoduchý text, nebo HTML v závislosti na uživatelově uvážení. E-maily napsané pomocí HTML umožňují navíc vkládání speciálního formátování a multimediálního obsahu přímo do těla zprávy. Na konci obsahu e-mailové zprávy může být také přidán automaticky generovaný text nebo podpis [2].

Formáty e-mailových zpráv

Výběr formátu e-mailové zprávy hraje klíčovou roli v tom, jak bude příjemci prezentována. *Outlook*, jeden z nejvíce používaných e-mailových klientů, nabízí tři odlišné formáty zpráv: *HTML*, *čistý text* a *RTF (Rich Text Format)*. *Gmail* omezuje výběr na dvě varianty: *HTML* a *čistý text*. Volba formátu závisí na obsahu a vizuálních prvcích použitých v těle e-mailové zprávy.

Čistý textový formát

Tento formát je kompatibilní a čitelný ve všech e-mailových klientech. Je založený na čistém textu, tudíž chybí podpora barev, různých řezů písem a dalších formátovacích vlastností. Obrázky je možné vkládat jako přílohy, nikoliv jako součást těla e-mailové zprávy [3].

HTML formát

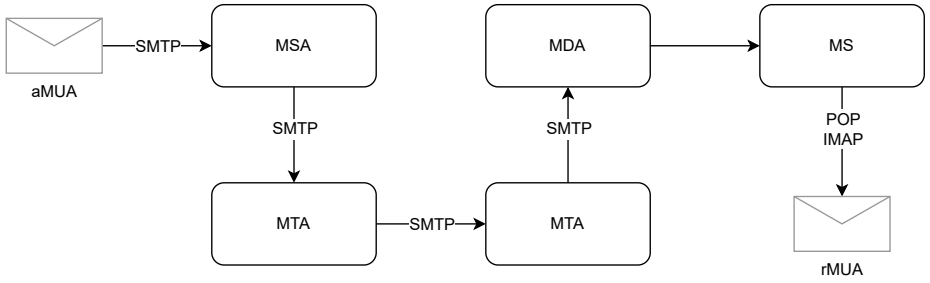
Jedná se o formát, který je ideální pro vytváření zpráv s různými typy písem, barvami a formátováním. Podporuje vkládání multimédií, jako jsou například obrázky, do těla e-mailové zprávy [3].

Rich Text Format (RTF)

Jedná se o exkluzivní formát podporovaný ve všech verzích *Outlook*. U zpráv poslaných na jiný e-mailový klient dochází ke konverzi na HTML formát [3].

1.2 Přenos e-mailové zprávy

Po odeslání e-mailové zprávy z e-mailového klienta odesílatele může tato zpráva projít několika mezilehlými servery, než bude úspěšně doručena do schránky příjemce. Tento proces řídí několik tzv. *agentů*. Jednotlivé agenty lze vidět na obr. 1.1.



Obr. 1.1: Přenos e-mailové zprávy.

Mail User Agent – MUA

Jedná se o e-mailový klient, který uživatel používá pro psaní nebo čtení e-mailových zpráv. Existují dva typy MUA – autorský MUA (*aMUA*) a MUA příjemce (*rMUA*).

Autorský MUA slouží k vytváření a odesílání e-mailových zpráv, jeho funkcí je rovněž uchovávání kopií zpráv na svém úložišti. MUA příjemce přijímá a zpracovává zprávy, které mu byly adresovány [4].

Message Store – MS

Slouží pro dlouhodobé ukládání e-mailových zpráv. Získává zprávy prostřednictvím MDA a umožňuje MUA k těmto zprávám přistupovat pomocí protokolů IMAP nebo POP [4].

Mail Submission Agent – MSA

Jedná se o prostředníka mezi e-mailovým klientem MUA a agentem pro přenos e-mailové pošty MTA. Přijímá zprávy od e-mailového klienta, před samotným přenosem provádí kontrolu zpráv a pokud se v nich nevyskytuje chyba, předá je k odeslání MTA [4].

Mail Transfer Agent – MTA

Zajišťuje přenos e-mailových zpráv, přičemž prostřednictvím směrování určuje optimální cestu k MDA příjemce. E-mailová zpráva může při této cestě projít několika mezilehlými MTA servery než dorazí do cíle. V případě selhání umožňuje MTA opakovaný přenos zprávy. Komunikace mezi jednotlivými MTA servery je často uskutečňována pomocí protokolu SMTP.

Pro směrování se používá DNS MX záznam, který specifikuje, jaký MTA by měl být použit pro dosažení cílové domény [4].

Mail Delivery Agent – MDA

Přebírá e-mailové zprávy od MTA a zajišťuje jejich doručení do úložiště zpráv, případně přímo do schránky příjemce. Stará se také o přesměrování e-mailových zpráv na základě preferencí příjemce [4].

1.3 Doménová jména a systém DNS

Domain Name System je zodpovědný za překlad doménových jmen na číselné IP adresy, které identifikují konkrétní počítače či servery v síti Internet. DNS si lze představit jako distribuovanou databázi doménových jmen, která má hierarchickou strukturu.

DNS funguje na bázi klient-server. Serverovou část tvoří tzv. *name servery*, což jsou servery obsahující informace o určitých segmentech databáze. Tyto informace poskytují uživatelům, tzv. *resolverům*. *Resolver* vytváří dotazy a odesílá je na *name server* [5].

Doménová jména slouží pro jednodušší přístup k informacím v DNS. Každá doména je spojena s unikátní IP adresou, kterou lze identifikovat pomocí DNS databáze [5].

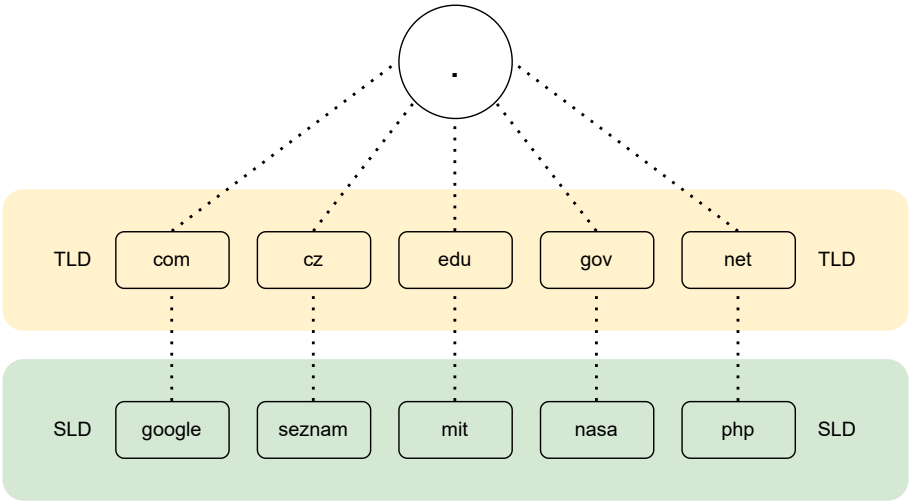
Hierarchická struktura DNS

Celá struktura DNS je modelována jako obrácený strom, kde každý uzel reprezentuje konkrétní doménu. Tato hierarchie umožňuje organizaci domén do podstromů a subdomén, což usnadňuje správu a decentralizaci informací. Každá doména má unikátní název, který identifikuje její pozici v hierarchii a umožňuje jednoznačný přístup k informacím [5].

Doménová jména jsou dělena do několika úrovní, oddělených tečkami. Existují *top-level domény (TLD)*, *domény druhé úrovně (SLD)* a *domény dalších úrovní*, které vytvářejí hierarchickou strukturu, viz obr. 1.2. Domény jsou rozděleny tečkami, plný

zápis tvoří úplné doménové jméno. Doménová jména musí být unikátní v rámci dané úrovně [6]. Tímto způsobem je zajištěno, že každý uzel v rámci DNS má jedinečnou identitu.

Každá *TLD* eviduje informace o všech doménách druhého řádu. Obdobně každá doména druhého řádu uchovává informace o doménách třetího řádu [6].



Obr. 1.2: Hierarchie doménových jmen.

Decentralizace je umožněna díky delegaci odpovědnosti za subdomény různým organizacím [5]. Například organizace spravující doménu „cz“ může delegovat odpovědnost za subdoménu „seznam.cz“ na jinou organizaci, což vytváří novou autonomně spravovanou zónu.

Díky doménovým jménům a systému DNS je zaručeno, že si uživatel nemusí pamatovat IP adresu jednotlivých uzlů v síti, ale může použít lépe zapamatovatelné doménové jméno.

Princip fungování DNS

1. Uživatel otevře ve svém webovém prohlížeči libovolnou webovou stránku, např. „*www.seznam.cz*“.
2. Počítač nejprve kontaktuje svůj lokální DNS server. Pokud lokální DNS server nezná požadovanou IP adresu, zeptá se na jméno „*www.seznam.cz*“ některého z kořenových serverů.
3. Pokud kořenový server odpověď nezná, předá lokálnímu DNS serveru seznam *name serverů* pro doménu „*.cz*“.
4. Lokální DNS server kontaktuje jeden z těchto serverů a dotáže se na jméno „*www.seznam.cz*“.

5. Pokud server odpověď opět nezná, pošle informaci o serveru, který obsahuje doménu „*seznam.cz*“.
6. Lokální DNS server pošle dotaz na server, který poskytuje doménu „*seznam.cz*“.
7. Dotázaný server pošle odpověď s IP adresou příslušící doménovému jménu „*www.seznam.cz*“.
8. Tuto adresu předá lokální DNS server zpět počítači, který poté může zobrazit webovou stránku [6].

Typy DNS záznamů

Existuje několik typů DNS záznamů, vybrané z nich popisuje tabulka 1.1.

Tab. 1.1: DNS záznamy [7].

Záznam	Význam
A	Slouží pro nasměrování domény na konkrétní IPv4 adresu.
AAAA	Slouží pro nasměrování domény na konkrétní IPv6 adresu.
CAA	Určuje certifikační autoritu, která může doméně vystavit certifikát.
CNAME	Slouží k přesměrování jedné domény na druhou.
MX	Určuje, na jaký poštovní server je směrovaná pošta zaslaná na doménu.
NS	Určuje autoritativní *name servery* pro doménu.

2 Sociální inženýrství

Sociální inženýrství můžeme chápat jako umění manipulace a klamání s cílem získat osobní a citlivé informace od jednotlivců prostřednictvím digitálního prostředí. Tato metoda kybernetických útoků spočívá v tom, že útočníci využívají různé psychologické triky a manipulační techniky s cílem přesvědčit oběti k prozrazení jejich citlivých údajů.

Hlavním cílem sociálního inženýrství je získání informací bez nutnosti použít technické dovednosti nebo napsat kód. Manipulační postupy zahrnují získání důvěry nebo vydávání se za důvěryhodnou osobu, čímž se útočník snaží získat přístup k citlivým údajům oběti. I přesto, že tato metoda nevyžaduje hluboké technické znalosti, je považována za účinnou a neměla by být podceňována, protože velká část kybernetických hrozeb vzniká právě prostřednictvím sociálního inženýrství [8].

2.1 Fáze útoků sociálního inženýrství

Celý proces útoku se skládá z několika fází. Začíná fází přípravy, kde útočník shromažďuje klíčové informace o své oběti. Následuje fáze infiltrace, během které útočník navazuje kontakt s obětí a snaží se získat její důvěru. Samotný útok probíhá ve fázi, kdy útočník využívá různé metody k získání požadovaných informací od oběti. Po dosažení cíle přichází fáze opuštění, kde útočník okamžitě opouští komunikaci, aby minimalizoval riziko odhalení. Pro útočníky je často snadné získat přístup k citlivým datům pouhým získáním jediného uživatelského jména a hesla [8].

Snižování dopadu útoků sociálního inženýrství

Nejrizikovější fází sociálního inženýrství je fáze navazování vztahu s potenciální obětí. Klíčovým opatřením k minimalizaci úspěchu útoků je systematická příprava oběti na možná rizika a úskalí v této fázi.

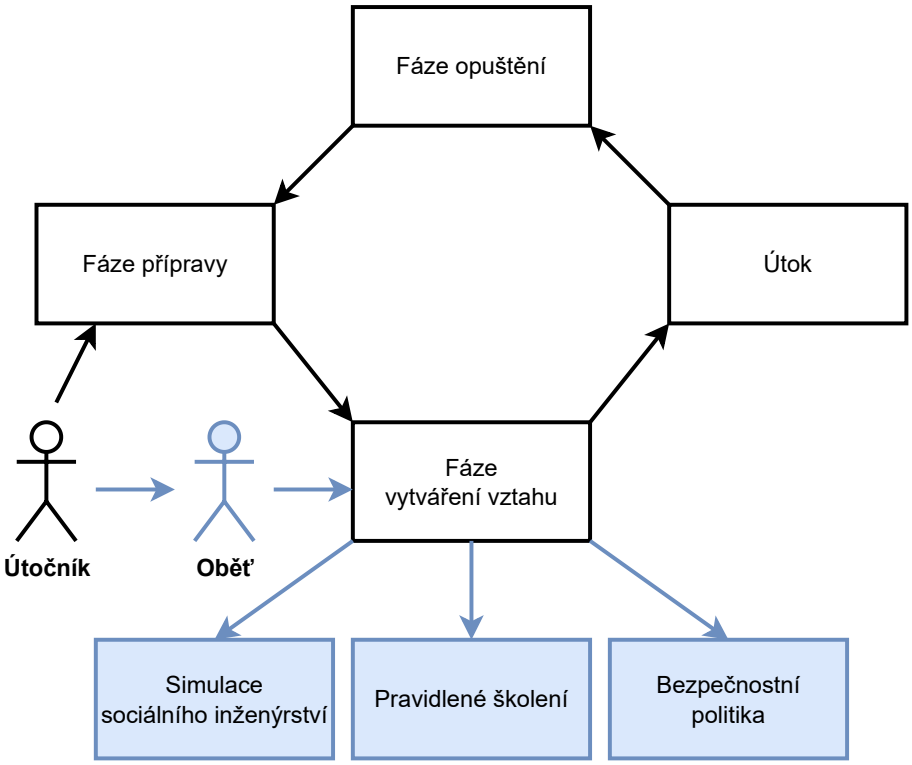
Příprava by měla zahrnovat pravidelné simulace útoků sociálního inženýrství, které umožní obětem lépe rozpoznat potenciální hrozby a adekvátně na ně reagovat. Kromě toho je nezbytné zajistit pravidelná školení, poskytující obětem aktuální informace o nových metodách sociálního inženýrství [9].

Implementace bezpečnostní politiky, kterou by všichni uživatelé měli striktně dodržovat, představuje další účinné opatření. Tato politika by měla zahrnovat [9]:

- nastavení filtrů pro rozpoznání podvodných e-mailů a ověřování totožnosti odesílatele e-mailu,
- používání ověřeného a bezpečného softwaru s pravidelnými aktualizacemi,
- pravidelnou účast na školeních v oblasti bezpečnosti,

- implementaci dvou-faktorové nebo více-faktorové autentizace pro zvýšení bezpečnosti,
- pravidelnou aktualizaci hesel a zabránění opakování stejných hesel,
- minimalizaci digitální stopy na sociálních médiích s cílem omezit dostupnost osobních informací,
- vyvarování se návštěv podezřelých internetových stránek a otevírání e-mailů nebo příloh od neznámých zdrojů.

Jednotlivá opatření a fáze útoků sociálního inženýrství zachycuje obrázek 2.1.



Obr. 2.1: Fáze a snižování dopadu útoků sociálního inženýrství [9].

2.2 Typy útoků sociálního inženýrství

Útočníci používají různé strategie a širokou škálu útoků k získání neoprávněného přístupu k citlivým informacím. V této části bakalářské práce jsou popsány některé z nejběžnějších typů útoků.

Phishingové útoky

Útočníci využívají falešné e-maily tvářící se jako důvěryhodný zdroj od známe společnosti nebo státní instituce. Cílem je přesvědčit oběť k zaslání osobních údajů jako jsou hesla nebo bankovní údaje [8].

Spam útoky

Tato metoda představuje taktiku, kdy útočník hromadně rozesílá e-mailové zprávy s cílem oslovit co největší množství lidí. Primárním účelem je získání citlivých informací od potenciálních obětí, které reagují na tyto podvodné e-maily [8].

Search Engine podvody

Jedná se o strategii, při níž útočníci manipulují s výsledky vyhledávání tak, že falešné webové stránky dosahují umístění na vrcholu seznamu výsledků. Tímto způsobem se webové stránky tváří důvěryhodně a bezpečně, což může přimět oběť k nevědomému navštívení těchto falešných stránek [8].

Baiting útoky

Tato taktika zahrnuje například strategické umístění USB na veřejných místech nebo odesílání e-mailových zpráv s přílohami, které lákají oběti na bezplatný software nebo jiné atraktivní nabídky. Útočníci využívají lidskou tendenci k zájmu o bezplatné či atraktivní nabídky, což může mít za následek, že oběti podlehnou těmto nabídkám a stáhnou si přílohu do svého zařízení. Přitom může dojít k nakažení systému nebo kompromitaci citlivých informací obětí [8].

Fyzické útoky na bezpečnost

Jedná se o situace, kdy útočník osobně vstupuje do budovy společnosti. Během tohoto typu útoku se útočník často vydává za zástupce důvěryhodné společnosti, případně může být i bývalým zaměstnancem, který díky znalosti interních systémů a politik společnosti umí lépe maskovat své skutečné záměry. Tento typ útoku nese obrovská rizika, zejména kvůli vysoké pravděpodobnosti odhalení [8].

Quid pro Quo útoky

Tato strategie zahrnuje lákání na speciální slevy nebo dárky zdarma výměnou za osobní informace oběti. Často se jedná o webové stránky nebo reklamy, které nabízejí produkty nebo dárky zdarma, pokud uživatelé poskytnou své jméno, příjmení a adresu. Tato technika je běžně používaná v rámci sociálního inženýrství, kde útočníci využívají lidskou touhu po získání něčeho zdarma k dosažení svých cílů [8].

DNS spoofing

Jedná se o techniku podvržení DNS, při které útočníci mění proces překladu doménových jmen na IP adresy. Pokud pak oběť zadá legitimní URL adresu, je přesměrována na podvodnou webovou stránku, která vypadá totožně jako legitimní, což značně ztěžuje tento podvod odhalit [8].

Scareware útoky

Tyto útoky si kladou za cíl vyděsit oběť a vynutit v ní pocit nutnosti okamžité reakce na událost. Může se jednat například o e-mailovou zprávu, která oběť informuje o podezřelém pokusu o přihlášení a vyzývá k okamžité změně hesla s hrozbou ztráty uživatelského účtu. Následkem podlehnutí nátlaku může dojít k odhalení citlivých informací nebo infikaci škodlivým softwarem [8].

Existuje spousta dalších strategií a útoků, které by se daly rozdělit do mnoha kategorií. V rámci této bakalářské práce bude blíže prozkoumána problematika podvodných e-mailových zpráv.

2.3 Podvodné e-mailové zprávy

Podvodné e-mailové zprávy představují jednu z nejběžnějších forem sociálního inženýrství. Útočníci využívají klamavé a manipulační techniky k získání citlivých informací od obětí. E-mailová zpráva se může tvářit jako legitimní komunikace od důvěryhodného zdroje, jako jsou banky, obchodní společnosti nebo sociální sítě. Cílem je nalákat oběti pomocí této zdánlivě legitimní komunikace a využít tak získané důvěry ke krádeži hesel, platebních údajů nebo jiných citlivých informací.

Charakteristiky podvodných e-mailů

E-mail od podvodníka lze rozpoznat pomocí určitých charakteristik, kterými jsou například [10, 11]:

- Přítomnost elektronických formulářů v těle e-mailové zprávy, případně odkaz na podvodnou webovou stránku s formulářem, která nabádá uživatele k vyplnění citlivých údajů.
- HTML kód v e-mailové zprávě, obsahující skryté odkazy nebo skripty, které mohou být použity k přesměrování na podvodné webové stránky.
- URL adresy odkazující na falešné webové stránky, které se podobají nebo jsou zkopírované z legitimní stránky.
- Podvodné e-maily mohou obsahovat infikované přílohy, které mohou uživatele připojit k webové stránce, kde bude vyzván k zadání citlivých údajů. Přílohou může být také infikovaný soubor, který je stažen na zařízení uživatele.
- Podvodné e-maily často obsahují pravopisné a gramatické chyby.
- Útočníci využívají jazyk, který záměrně vyvolává dojem naléhavosti a snaží se podnítit uživatele k okamžité reakci.

Typy podvodných e-mailů

Existuje nemalá škála způsobů, jak klasifikovat podvodné e-maily, mohou se lišit podle svého účelu, obsahu a způsobu, jakým se snaží uživatele oklamat. Tato kapitola se zaměřuje na popis některých hlavních a obecně známých typů těchto podvodných e-mailů a poskytuje vhled do jejich charakteristik a strategií, které útočníci využívají k oklamání obětí.

Spear phishing

Jedná se o útok, který je zaměřený na konkrétní osobu nebo skupinu osob. Útočník obvykle nejprve shromažďuje různé informace o oběti, které pak využívá pro vymyšlení personalizovaného útoku, který zvyšuje pravděpodobnost úspěchu.

Cílem útoku je oklamat oběť pomocí e-mailové zprávy, která obsahuje odkaz na falešně vytvořené stránky, pomocí kterých může útočník získat přihlašovací údaje. Případně se může jednat o podvod využívající infikovaný dokument přiložený k obsahu e-mailové zprávy jako příloha, který oběť stáhne do svého počítače. Dokument může obsahovat malware, který umožní útočníkovi krádež dat a získání kontroly nad zařízením [12].

E-mailová zpráva tohoto typu může na první pohled vypadat jednoduše. Útočník může využívat techniku, kdy oběti nejprve zasílá e-mailové zprávy, ve kterých se snaží navázat konverzaci na nějaké téma. To může způsobit větší otevřenost a důvěru

oběti vůči útočníkovi. Tato konverzace může být rozložena do několika e-mailů, ve kterých postupně útočník stupňuje své požadavky.

Útočník se také může vydávat za vysoce postaveného ředitele společnosti a prostřednictvím e-mailu žádat oběť o okamžitou pomoc. Může například tvrdit, že je mimo kancelář a nachází se v tíživé situaci. K zakrytí případných chyb nebo překlepů v textu využívá různé záminky, které mají ospravedlnit jeho nepřesnosti a zároveň posilovat dojem jeho autenticity. Tato taktika mu umožňuje získat důvěru oběti a snadněji dosáhnout svých podvodných cílů [13].

Clone phishing

Jedná se o typ podvodu, který se zaměřuje na velké množství osob. Útočník se pokouší napodobit legitimní e-mailovou zprávu, kterou již oběť obdržela. Může se jednat například o zprávu od banky nebo internetového obchodu, kterou útočník pozmění a odešle oběti pod záminkou opakované nebo aktualizované verze předchozí zprávy. Znovu doručený e-mail často u oběti nevzbudí podezření, protože vypadá důvěryhodně. Útočník však do zprávy může zahrnout podvodné odkazy nebo škodlivý kód [12].

Whaling

Jedná se o podvod zaměřený na vysoce postavené osoby jako jsou například pracovníci ve vrcholovém managementu. Na rozdíl od běžných podvodných technik, které často využívají falešné webové stránky nebo škodlivé odkazy, vystupuje tento typ podvodu skrytěji. V tomto případě se útočníci snaží přesvědčivě napodobit významné postavení vrcholového manažera.

Jednou ze stále častějších technik, které útočníci používají (zejména v zahraničí) je zneužití falešných daňových formulářů. Tyto formuláře obsahují klíčové informace, jako jsou jména, adresy nebo údaje o bankovních účtech. Útočník tak získává citlivé údaje prostřednictvím legitimně vypadajících dokumentů. Jedná se o daleko cílenější metodu než u běžných podvodných útoků [14].

Deceptive phishing

Jedná se o podvodné e-mailové zprávy, které se snaží napodobit komunikaci od důvěryhodné společnosti. E-mailová adresa může mít tvar „*support@apple.com*“, který se vizuálně podobá oficiálním adresám. E-mailová zpráva obdržená od této podvodné adresy může uživatele nabádat například k obnovení hesla k jeho uživatelskému účtu pod záminkou opatření bezpečnosti [15].

Ochrana před podvodnými e-maily

Poskytovatelé e-mailových služeb standardně poskytují bezpečnostní opatření proti podvodným e-mailům. Implementují například opatření využívající umělou inteligenci nebo pokročilé filtrační technologie [16, 17, 18].

Následující popis se zaměří na specifická bezpečnostní opatření, která používá společnost Google při ochraně uživatelů před podvodnými e-maily.

Filtrování spamu a podvodných e-mailů

Využití umělé inteligence a algoritmů, podporovaných uživatelskou zpětnou vazbou, slouží k identifikaci spamu a nebezpečných e-mailů. Interakce od uživatelů, zahrnující označování e-mailů jako nevyžádané pošty, pomáhají zlepšovat proces filtrování. Filtrování přijaté pošty probíhá také na základě IP adres, domén a autentizací [19].

Varování před nebezpečnými přílohami

Bezpečnostní systémy jsou vybaveny funkcemi pro rozpoznávání rizikových příloh a před jejich otevřením poskytují uživatelům upozornění [16]. Tato upozornění se aplikují zejména na přílohy obsahující skripty, které mohou spustit škodlivý software. Také se monitorují zašifrované soubory, u nichž nelze ověřit přítomnost virů nebo malware [20].

Varování před nebezpečnými odkazy

V případě, že e-mailová zpráva obsahuje potenciálně nebezpečný odkaz, jsou uživatelé informováni o možném riziku ještě před zobrazením odkazu [16].

Šifrovaná komunikace

E-mailová komunikace je zabezpečena šifrováním jak v průběhu jejího přenosu, tak během ukládání samotných zpráv, což přispívá k ochraně dat před neoprávněným přístupem. Toto šifrování se provádí pomocí standardu TLS (Transport Layer Security). Implementace tohoto standardu zajišťuje, že veškerá komunikace mezi odesílatelem a příjemcem je chráněna šifrovací vrstvou, což přispívá k minimalizaci rizika odposlechu nebo manipulace s daty [16].

Jedná se o základní bezpečnostní opatření, která společnost Google poskytuje. Tato opatření chrání uživatele před podvodnými e-maily a zajišťují bezpečnou e-mailovou komunikaci.

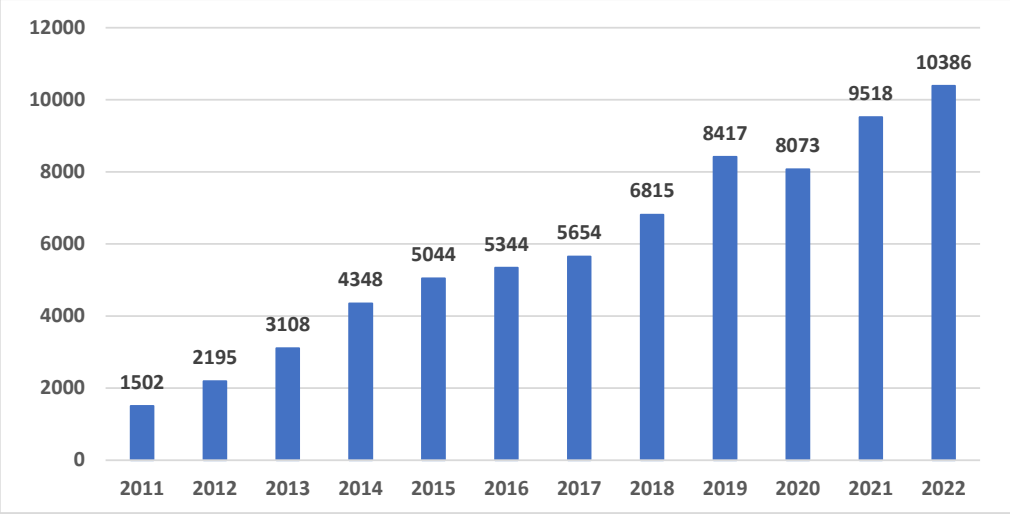
2.4 Sociální inženýrství v České republice

V posledních pár letech došlo v České republice, dle České bankovní asociace, k několikanásobnému nárůstů útoků na klienty bank. Podle dostupných dat se v roce 2022 průměrná škoda na jednom poškozeném klientovi vyšplhala na 161 500 korun, u případů podvodného navolávání dosahovaly částky až čtvrtmilionových škod [21].

Útočníci se zaměřují na lidský faktor a vytvářejí různé legendy využívající nátlak, strach a časovou tíseň ke zmanipulování obětí a dosažení finančního zisku. Mezi časté podvodné praktiky patří vydávání se za bankéře, policisty nebo pracovníky technické podpory, s cílem získat peníze nebo vzdálený přístup k zařízení oběti. Dalšími metodami jsou lákavé nabídky investic, reverzní inzertní podvody a podvody typu Nigerijské dopisy [21].

Podvodníci využívající metodu známou jako „Nigerijské dopisy“ komunikují s obětí výhradně na dálku, například prostřednictvím e-mailové zprávy. Záměrem je vylákat peníze od oběti prostřednictvím vymyšlených příběhů. U těchto typů podvodů často útočník využívá emocionální manipulace, oslovuje citovou stránku oběti a vytváří zdání náklonnosti či soucitu. Další častou taktikou je lákat oběť na možnost získání odměny za dočasnou pomoc s převozem, převodem, doručením nebo jinou transakcí týkající se majetku a peněz [22].

V poslední době se objevují i podvody na sociálních sítích, kdy útočníci krátkodobě kradou identity reálných osob a oslovují jejich přátele s úmyslem vylákat peníze. Nově se také šíří tzv. reverzní inzertní podvody, kdy jsou prodávající klamáni fiktivními platebními branami, což může vést ke ztrátě veškerých úspor [21].



Obr. 2.2: Počty registrovaných trestných činů páchaných v kyberprostoru v České republice [21].

3 Protokoly elektronické pošty

Pro efektivní používání služeb elektronické pošty vznikly standardizované protokoly, které umožňují výměnu elektronických zpráv mezi servery a poštovními klienty po celém světě. Tato kapitola se zaměří na základní seznámení s protokoly elektronické pošty.

3.1 Protokol SMTP

Simple Mail Transfer Protocol se používá pro přenos e-mailových zpráv na internetu. Jeho hlavním cílem je spolehlivý přenos zprávy od odesílatele k příjemci. Důležitou vlastností protokolu SMTP je schopnost přenášet zprávy přes více sítí, tedy ze serveru odesílatele přes mezilehlé servery až na server příjemce. Funguje na principu přijetí zprávy od e-mailového klienta na poštovní server a následném přenosu této zprávy mezi jednotlivými poštovními servery [23].

Když chce e-mailový klient odeslat e-mailovou zprávu, naváže spojení s SMTP serverem a určí adresu SMTP serveru příjemce zprávy. SMTP server může být buď konečný server pro doručení zprávy nebo *relay* server, který předá zprávu dalším serverům [23].

Komunikace mezi klientem a serverem a mezi jednotlivými servery probíhá pomocí sérií příkazů a odpovědí [23]. Klient obvykle odesílá příkazy pro určení odesílatele, příjemce a přenos obsahu zprávy, některé příkazy jsou uvedeny v tabulce 3.1. Server na ně reaguje odpověďmi, kterými oznamuje stav zpracování příkazů.

Tab. 3.1: Základní příkazy klienta SMTP.

Příkaz	Význam
HELO/EHLO	Zahájení spojení
MAIL FROM	Identifikace odesílatele
RCPT TO	Identifikace příjemce
DATA	Samotný obsah zprávy

Může se stát, že je jedna zpráva adresována více příjemcům ve stejné doméně, což je výhodné, protože přenos této zprávy snižuje redundanci. Jakmile server e-mailovou zprávu příjme, stává se zodpovědný za její správné doručení, případně za oznámení o selhání doručení. Po odeslání zprávy může klient požádat o uzavření spojení nebo provádět další e-mailové transakce [23].

V případě komunikace přes *relay* servery se používají DNS MX záznamy [23]. Standardními porty pro protokol SMTP jsou port 25 a port 587.

3.2 Protokol MIME

Multipurpose Internet Mail Extension (MIME) je doplňkový protokol, který umožňuje prostřednictvím protokolu SMTP odesílat data jiná než ASCII. V e-mailové zprávě umožňuje přenášet navíc od klasického textu například audio, video, obrázky nebo dokonce aplikační programy [24].

SMTP umožňuje pouze přenos krátkých řádků složených ze 7bitového ASCII, to znamená, že všechna netextová data musí být před přenosem převedena na 7bitový formát. Nastává zde problém u jazyků jako jsou například francouzština nebo ruština, které nepodporují tento 7bitový formát. Právě kvůli těmto nedostatkům došlo k rozšíření SMTP o MIME [24, 25].

Výhody protokolu MIME

- Umožňuje do jedné zprávy přidat několik příloh.
- Není omezena celková délka zprávy.
- Různý typ obsahu – audio, video, spustitelné soubory.
- E-mailové zprávy mohou být psány libovolným jazykem.
- Možnost formulovat zprávu pomocí HTML a kaskádových stylů [24].

Fungování protokolu MIME

E-mailové zprávy vytvořené pomocí MIME lze přenášet běžnými protokoly, jako je SMTP, POP nebo IMAP, přičemž záhlaví těchto zpráv může být tvořeno jinými znakovými sadami něž ASCII.

Servery přenášející zprávu vkládají hlavičku MIME na začátek přenosu, klienti volí vhodné aplikace pro zobrazení datového typu uvedeného v této hlavičce. Aplikace mohou být zabudované přímo ve webovém klientu nebo musí být data nejprve stažena a poté zobrazena v požadovaném přehrávači.

Odesílající strana transformuje data na 7bitová NVT ASCII a doručuje je klientovi SMTP. Na straně příjemce jsou pak data rekonstruována na původní [26].

Hlavička protokolu MIME

Hlavička protokolu MIME se skládá z několika dílčích částí, které specifikují vlastnosti e-mailové zprávy. Mezi tyto dílčí části patří [24, 25, 26]:

- **MIME-Version** – Tato část indikuje, že zpráva používá formát MIME, hodnota je obvykle nastavena na „1.0“.
- **Content-Type** – V této části je zahrnutý popis média, které e-mailová zpráva přenáší. Skládá se z dvojice typ–podtyp. Typ obecně označuje druh datového

přenosu, může se jednat například o video nebo text. Podtyp určuje přesný druh dat. Vybrané typy a podtypy zobrazuje tabulka 3.2.

- **Content-Transfer-Encoding** – Obsahuje metodu použitou pro kódování zprávy.
- **Content-ID** – Používá se pro jednoznačnou identifikaci e-mailové zprávy.
- **Content-Description** – Jedná se o nepovinné pole, které popisuje přenášený typ média.

Tab. 3.2: Příklad typů a podtypů přenášených dat protokolu MIME [27].

Typ	Podtyp	Zápis
application – binární data, která mohou být provedena nebo interpretována, nebo jejichž použití vyžaduje specifickou aplikaci.	např. pdf	application/pdf
audio – zvuková nebo hudební data.	např. mpeg	audio/mpeg
font – údaje o fontu.	např. ttf	font/ttf
image – grafická data, bitmapové, vektorové a animované obrázky.	např. jpeg	image/jpeg
text – pouze textová data.	např. html	text/html
video – video data nebo soubory.	např. mp4	video/mp4

3.3 Protokol POP3

Post Office Protocol verze 3 byl v minulosti nejběžněji používaný protokol pro stahování e-mailových zpráv z poštovního serveru. Tento protokol komunikuje přes port 110 a umožňuje e-mailovému klientu přistoupit k poštovnímu serveru, odkud si může zprávu stáhnout na lokální úložiště [28].

POP3 nenabízí možnost spravovat e-mailové zprávy na serveru. Běžně je používán tak, že po stažení zprávy na lokální zařízení se ze serveru zpráva smaže. Pro pokročilejší správu e-mailů na serveru a synchronizaci mezi různými zařízeními se preferuje protokol IMAP4 [29].

První verze POP vyšla v roce 1984, nebyla však nikterak oblíbená. Druhá verze protokolu – POP2 zavedla některé změny, ale podobně jako její předchůdce se neujala širokého rozšíření. Nynější verze POP3 prošla významnou změnou fungování a stal se z ní po dlouhou dobu používaný standard. POP3 je popsán v celé řadě RFC dokumentů, mezi nejzajímavější patří například RFC 1939, který se věnuje bezpečnosti a implementaci, nebo RFC 1734, který rozšiřuje POP3 o kryptografické ověřování a soukromí [28].

Vznikl i návrh protokolu POP4, který ale nikdy nebyl standardizován nebo široce využíván. Tento návrh přinesl několik zajímavých rozšíření, jako je přidání příkazů souvisejících se správou složek na serveru, nové příznaky související se zprávami, podpora částečného načítání zpráv atd. Tato rozšíření činí POP4 flexibilnější [30].

Celkově lze říci, že v současné době se protokol POP3 používá zřídka a je preferován protokol IMAP, který je v mnoha ohledech pokročilejší.

3.4 Protokol IMAP

Internet Message Access Protocol je alternativou k protokolu POP3. Podporuje pokročilejší funkce, zároveň klade větší požadavky na servery a základní úložiště zpráv. Protokol IMAP využívá port 143 a stejně jako POP se zabývá načítáním zpráv e-mailových klientů z poštovního serveru [28].

První dokument RFC věnující se protokolu IMAP byl RFC 1064, další standard nazývaný IMAP4 popisují dokumenty RFC 1730–1733. Pozdější doplňující dokumenty, jako jsou například dokumenty RFC 2060–2062, popisují vylepšenou verzi protokolu IMAP4 a poskytují informace o zpětné kompatibilitě [28].

Stejně jako u protokolu POP komunikují klienti a servery IMAP pomocí zpráv sestávajících z ASCII znaků. Klient IMAP odesílá na server požadavek a server posílá zpět odpověď o stavu zpracování tohoto požadavku. Od protokolu POP se liší strukturou zpráv, všechny zprávy začínají značkou, která slouží jako jedinečný identifikátor v rámci daného připojení. Díky tomuto identifikátoru je možné přiřadit odpověď ke konkrétnímu požadavku [28].

IMAP4 umožňuje klientovi ovládat a spravovat elektronickou poštu na serveru. Poskytuje možnost pracovat s e-mailovými schránkami podobně jako s lokálními složkami a umožňuje i synchronizaci mezi offline klientem a serverem. K přístupu k jednotlivým zprávám využívá buď pořadové číslo zprávy nebo unikátní identifikátor [31].

Výhodou oproti protokolu POP3 je možnost přihlášení k poštovní schránce z vícero zařízení současně. Pošta je synchronizovaná a uložená na serveru a je přístupná ze všech připojených zařízení. Přijaté a odeslané e-maily zůstávají uloženy na serveru, dokud je uživatel sám trvale nesmaže [32].

4 Použité programové moduly a knihovny

Tato kapitola představuje klíčové programové moduly a knihovny použité v rámci této bakalářské práce.

4.1 Grafické moduly a knihovny

Tato podkapitola se věnuje klíčovým grafickým modulům a knihovnám, které byly použity pro tvorbu e-mailového klienta. Podrobněji jsou zde popsány tři hlavní grafické moduly a knihovny: *Tkinter*, *PIL* a *Pygame*. Každá z těchto částí poskytuje vhled do využití těchto nástrojů v rámci samotného projektu.

Modul Tkinter

Tkinter je framework pro grafické uživatelské rozhraní a patří do standardní knihovny jazyka Python. Výhodou tohoto modulu je, že je multiplatformní a vizuální prvky se přizpůsobují operačnímu systému, na kterém aplikace napsaná v Tkinteru běží [33]. Tento framework umožňuje snadnou práci s různými grafickými prvky, jako jsou tlačítka, textová pole nebo vstupní pole. Grafické prvky jsou jednoduše modifikovatelné, a to buď základními metodami obsaženými v Tkinteru, anebo rozšiřujícími moduly, jako je například *ttk* nebo *ttkbootstrap*, které přináší modernější vzhled jednotlivých grafických prvků.

Modul PIL

Modul *PIL* poskytuje nástroje k manipulaci s obrázky a je rovněž součástí původní knihovny programovacího jazyka Python. Hlavní třídou modulu *PIL* je třída *Image*, která se stará o čtení obrázku a ukládá jej v objektu typu *Image*. Pomocí třídy *Image* lze modifikovat obrázek například oříznutím nebo změnou jeho velikosti [34].

Knihovna Pygame

Pygame je knihovna, která obsahuje sadu modulů pro programovací jazyk Python. Je převážně navržena pro tvorbu her a multimediálních programů. Pomocí *pygame* lze pracovat s grafikou, zvukem a dalšími prvky. Jedním z důležitých modulů knihovny *pygame* je modul „*mixer*“, který pracuje se zvukovými objekty. Tento modul umožňuje přehrávat hudbu nebo zvukové efekty [35, 36].

4.2 Moduly a knihovny pro práci s e-maily

Tato podkapitola představuje klíčové programové moduly a knihovny, které jsou určeny pro práci s e-maily. Následující podkapitoly popisují tři hlavní moduly a knihovny: *imaplib*, *smtplib* a *Email*.

Modul Imaplib

Modul *imaplib* v jazyce Python slouží k připojení a komunikaci s IMAP servery. IMAP je standardní protokol pro přístup k elektronické poště uložené na serveru. Tento modul definuje tři hlavní třídy, které umožňují navázání a správu spojení s těmito servery [37].

Třída IMAP4

Třída *IMAP4* se používá k inicializaci a tvorbě spojení se serverem. Lze zde nastavit parametry, jako je adresa serveru, port a časový limit pro spojení. Při inicializaci spojení je určena verze protokolu [37].

Třídá IMAP4_SSL

Jedná se o podtřídu odvozenou od třídy *IMAP4*, která se připojuje přes šifrovaný soket SSL. Navíc je rozšířena o možnost nastavení „*SSLContext*“, který umožňuje konfigurovat různé volby pro SSL, včetně certifikátů a soukromých klíčů [37].

Třída IMAP4_stream

Třída *IMAP4_stream* slouží k vytvoření spojení s IMAP serverem pomocí standardního vstupu a výstupu souboru, které jsou spojeny s procesem vytvořeným funkcí „*subprocess.Popen()*“ [37].

Modul Smtplib

Modul *smtplib* slouží k odesílání pošty na libovolný server SMTP nebo ESMTP. Tento modul je založen na normách popsaných v RFC 821 a RFC 1869, ve kterých jsou definovány operace pro přenos e-mailů přes SMTP. V *smtplib* jsou definovány tři třídy [38].

Třída SMTP

První třídou je *SMTP*, která navazuje spojení s SMTP serverem a při inicializaci umožňuje specifikovat volitelné parametry, jako je adresa serveru, port a další parametry, které budou použity při volání metody *connect()* [38].

Třída SMTP_SSL

Druhou třídou je *SMTP_SSL*, která funguje podobným způsobem, jako třída *SMTP*, navíc umožňuje zabezpečené připojení prostřednictvím SSL nebo TLS. Obsahuje *SSLContext*, který slouží k nastavení zabezpečeného spojení, včetně nastavení certifikátů a šifrovacích algoritmů [38].

Třída LMTP

Třída *LMTP* představuje klienta pro protokol LMTP (Local Mail Transfer Protocol), který je odvozený od ESMTP (Extended Simple Mail Transfer Protocol) a je podobný protokolu SMTP. Tato třída umožňuje vytvořit spojení s LMTP serverem a odesílat e-maily. LMTP může používat stejný autentizační mechanismus jako SMTP, ale obvykle autentizaci nepotřebuje nebo nepodporuje [38].

Knihovna Email

Jedná se o knihovnu pro správu e-mailových zpráv. Pro reprezentaci e-mailových zpráv v kódu se používá „objektový model“. Objektový model lze využít například k získávání informací o e-mailové zprávě nebo k vytváření nových e-mailů. Tato knihovna není navržena pro odesílání e-mailových zpráv na SMTP servery [39].

Modul header se používá v aplikacích, které potřebují kontrolovat znakové sady používané při kódování hlaviček. Mezi jeho funkce patří například *decode_header*. Tato funkce dekóduje hodnoty zprávy bez konverze znakové sady a vrací seznam dvojic, kde každá dvojice obsahuje dvě hodnoty: dekódovaný řetězec *decoded_string* a znakovou sadu *charset* [40].

4.3 Systémové moduly

Tato podkapitola se zaměřuje na klíčové systémové moduly. Systémové moduly poskytují nástroje pro manipulaci se systémovými funkcemi, regulárními výrazy, kódováním znaků, spouštěním externích procesů a správou záznamů, stejně jako práci s vlákny.

Modul os

Modul *os* pracuje s funkcemi závislými na operačním systému. Slouží ke čtení nebo zapisování souborů. Funkce *open* se používá pro práci se soubory, obsahuje tři argumenty, kterými jsou *filename*, *mode* a *encoding*. Filename specifikuje název souboru, se kterým bude manipulováno. Mode specifikuje, jakým způsobem bude soubor použit, nejběžnějšími argumenty jsou „r“ pro čtení, „w“ pro zápis a „a“ pro připojení obsahu na konec souboru. Encoding specifikuje, jak jsou řetězce zakódovány. Výchozí kódování záleží na platformě, ale standardně se používá UTF-8 [41, 42].

Modul re

Modul *re* poskytuje rozhraní pro práci s regulárními výrazy, což jsou schémata používaná k hledání shod v textu. Regulární výrazy umožňují definovat vzory pro textové řetězce, které mohou zahrnovat specifická slova, e-mailové adresy, URL adresy a mnoho dalšího. Tyto vzory lze kombinovat a použít ke hledání shod ve zdrojovém textu. Modul *re* umožňuje také provádění komplexních operací jako jsou rozdělení řetězců podle vzoru, nahrazení částí textu, nebo extrakce segmentů textu na základě regulárních výrazů [43].

Modul chardet

Modul *chardet* je nástroj používaný v Pythonu k automatické detekci kódování textu. Z posloupnosti bajtů v neznámém kódování znaků se snaží určit, jaké kódování bylo použito. Hlavní funkce modulu, *detect*, přijímá bajtový řetězec a vrací slovník s informacemi o detekovaném kódování, včetně pravděpodobnosti, s jakou bylo kódování identifikováno. Tato funkcionalita je využívána ve scénářích, kde je třeba zpracovávat textová data pocházející z různých zdrojů, což je běžné například při práci s internetovými daty [44].

Modul subprocess

Tento modul umožňuje spouštět nové procesy, interagovat s nimi a získávat jejich návratové kódy a výstupy. Funkce *run* umožňuje spustit příkazy na pozadí bez nutnosti ručního spouštění nebo otevírání terminálu. Slouží zejména k automatizaci často opakujících se úloh [45, 46].

Modul threading

Modul *threading* umožňuje vytvořit více paralelních toků, které umožňují provádět různé úkoly. Použití tohoto modulu je vhodné pro úkoly, které čekají na externí udá-

losti. Pro vytvoření nového vlákna se používá třída *Thread*, ve které lze specifikovat *target*. Target je funkce, která bude prováděna tímto vláknem [48].

Modul logging

Modul *logging* je nástroj pro sledování událostí, které se odehrávají během běhu aplikace. Hlavním účelem tohoto modulu je zaznamenávat události různé důležitosti, což umožňuje vývojářům monitorovat chování programu a diagnostikovat vzniklé problémy. Úrovně důležitosti, které modul *logging* poskytuje, zahrnují *DEBUG*, *INFO*, *WARNING*, *ERROR* a *CRITICAL*. Tyto úrovně pomáhají identifikovat vážnost zaznamenaných událostí a mohou být konfigurovány podle potřeb vývojáře, jejich popis uvádí tabulka 4.1. Modul také umožňuje konfiguraci výstupů, včetně možnosti ukládat logy do souborů v různých formátech [47].

Tab. 4.1: Popis důležitostí událostí modulu logging.

Příkaz	Význam
DEBUG	Obsahuje informace důležité pro diagnostiku problémů.
INFO	Informuje o funkčnosti aplikace.
WARNING	Informuje o problémových jevech v aplikaci.
ERROR	Oznamuje chybu při vykonávání funkce nebo při startu aplikace.
CRITICAL	Upozorňuje na závažné chyby, které by mohly způsobit pád aplikace.

4.4 Modul pro jednotkové testování

Tato kapitola se věnuje modulu pro jednotkové testování. Moduly pro jednotkové testování umožňují vývojářům efektivně testovat jednotlivé části kódu izolovaně od zbytku aplikace. Poskytují rozhraní pro definování testovacích případů, sestavování testovacích sad a automatické spouštění testů s podrobnými reporty o úspěšnosti.

Modul unittest

Modul *unittest* v jazyce Python je framework pro jednotkové testování, inspirovaný frameworkem *JUnit* z jazyka Java. Jeho klíčové vlastnosti zahrnují automatizaci testů, sdílení kódu pro *setup* a *teardown* akce před a po testech, seskupování testovacích případů do testovacích kolekcí – *suit* a nezávislost testů na systému reportování výsledků [49]. *Unittest* implementuje tyto koncepty v objektově orientované formě, některé základní koncepty jsou uvedené v tabulce 4.2.

Tab. 4.2: Klíčové koncepty modulu unittest.

Koncept	**Popis**
Test Fixture	Představuje přípravu prostředí pro provedení jednoho nebo více testů a následné úklidové akce.
Test Case	Individuální jednotka testování, která ověřuje specifickou odezvu na konkrétní vstupy.
Test Suite	Slouží k agregaci testovacích případů, aby byly provedeny společně.
Test Runner	Komponenta, která řídí provedení testů a zprostředkovává výsledky uživateli.

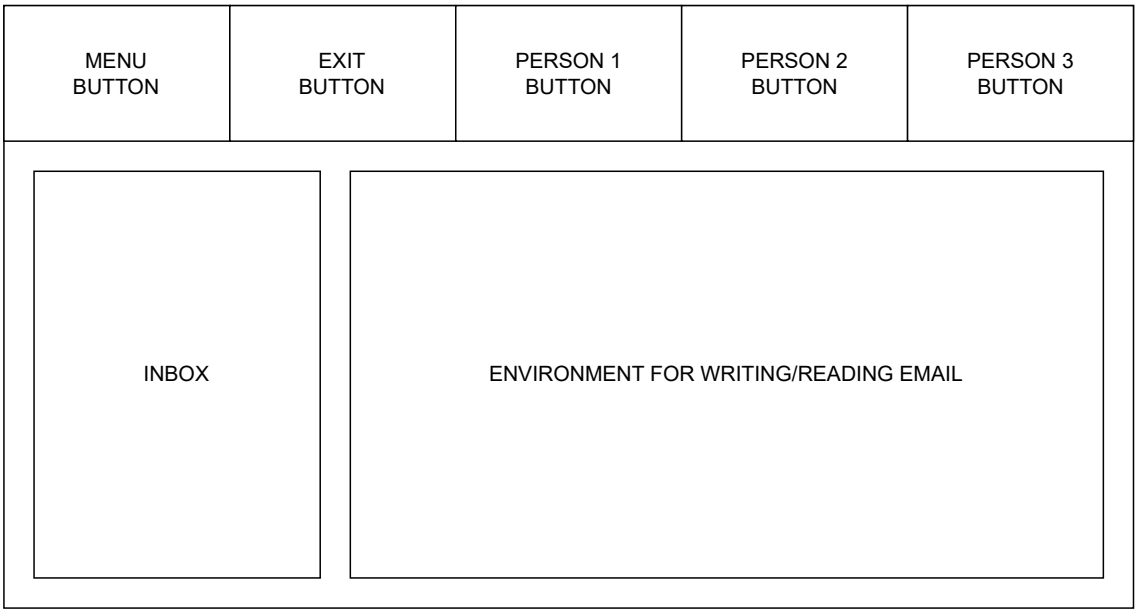
Každý test typicky obsahuje volání metod jako jsou *assertEqual()*, *assertTrue()*, *assertFalse()* nebo *assertRaises()*, které slouží k ověřování očekávaných výsledků nebo podmínek. Metody *setUp()* a *tearDown()* jsou používány k definování akcí, které se mají provést před začátkem a po skončení každého testovacího případu a umožňují udržet testovací kód dobře organizovaný a snadno udržovatelný [49].

5 Tvorba e-mailového klienta

Pro tvorbu e-mailového klienta pro seniory ve věku nad 90 let byl zvolen programovací jazyk *Python*. Tento jazyk obsahuje širokou nabídku knihoven a modulů, které umožňují vytvářet rozmanité aplikace. Jednou z klíčových knihoven pro tuto bakalářskou práci je knihovna *tkinter*, pomocí které bylo vytvořeno grafické uživatelské rozhraní.

5.1 Návrh grafického uživatelského rozhraní

Pro tvorbu grafického uživatelského rozhraní e-mailového klienta byl zvolen framework *tkinter*, převážně kvůli možnostem, které nabízí. Aplikace byla navržena k provozu v režimu celé obrazovky a je rozdělena do tří segmentů, jak je znázorněno na obrázku 5.1. První segment představuje prostor pro ovládání aplikace umístěný v horní části návrhu. Následuje prostor vyhrazený pro zobrazení doručené pošty v levé části aplikace a proměnná část pro psaní nebo čtení e-mailových zpráv v pravé části aplikace. Pozadí aplikace tvoří hlavní rámec, který spojuje všechny tyto části a pokrývá celé pozadí.



Obr. 5.1: Návrh grafického uživatelského rozhraní.

Část pro ovládání aplikace

V první části aplikace se nachází programovatelná tlačítka, která byla implementována pomocí předlohy. Tato předloha zajišťuje jednotný vzhled všech aplikací projektu *Senior-OS* a umožňuje vytvoření sad programovatelných tlačítek. V procesu návrhu bylo rozhodnuto implementovat dvě hlavní menu, mezi kterými se lze přepínat pomocí prvního tlačítka v každém menu. Každé menu obsahuje čtyři programovatelná tlačítka, jejichž funkce byly přeprogramovány podle specifických požadavků.

Tlačítka jsou vytvořena pomocí knihovny *tkinter*. Základ tvoří tzv. *master frame*, na kterém jsou rozmístěna. První tlačítko nesoucí název „*MENU 1*“ má textovou podobu a slouží k navigaci mezi menu. Během vývoje aplikace dále došlo k předělání dalších tlačítek na grafická. Druhé tlačítko, reprezentované bílým křížkem na červeném pozadí, slouží k ukončení aplikace. Následuje sada s obrázky osob, symbolizující členy domácnosti seniora nebo jeho přátele. Poslední tlačítko v druhém menu je textové a nese název *Komu*. Vizuální podobu těchto tlačítek lze vidět na obr. 5.2 a 5.3. Princip fungování těchto tlačítek popisuje podkapitola 5.2.

Obr. 5.2: Tlačítka obsažená v menu 1. Obrázky dostupné z: [50, 51, 52].

Obr. 5.3: Tlačítka obsažená v menu 2. Obrázky dostupné z: [53, 54, 55].

Část pro zobrazení doručených e-mailových zpráv

Druhá část se nachází v levém oddílu aplikace a slouží pro zobrazení doručených e-mailových zpráv do schránky seniora. Při běhu aplikace nemění svojí podobu.

Základem této části je rámec, který v sobě obsahuje dva prvky: *ScrolledListbox* a *Label*. *ScrolledListbox* slouží jako seznam, do kterého jsou po spuštění aplikace vloženy jednotlivé e-mailové zprávy. Součástí seznamu je posuvník umístěný v jeho

pravé části. *Label* je nápis, který se přizpůsobuje jazykovému nastavení aplikace a popisuje seznam doručených zpráv.

Při konfiguraci vlastností seznamu byl nejprve zvolen „rodičovský prvek“, do kterého byl seznam vložen. Dále se specifikoval font, který se získává z konfiguračního souboru a je stejný pro všechny aplikace v *Senior-OS*. Následně se seznamu nastavila jeho výška a vypnutí vizuálních efektů při najetí myší na jakýkoliv prvek v něm zobrazený. Nakonec se nastavil režim výběru položky, v tomto případě je povolen výběr pouze jedné. Vytvoření seznamu popisuje výpis 5.1.

Výpis 5.1: Vytvoření seznamu pro doručenou poštu.

```
self.inbox_list = ScrolledListbox(
    self.frame, font=font_config(),
    height=self.number_of_lines_listbox,
    activestyle="none", selectmode=tk.SINGLE
    )
```

Část pro psaní a čtení e-mailových zpráv

Poslední částí je pravý oddíl aplikace, který se mění v závislosti na stisknutém prvku na obrazovce. Zobrazuje se buď textové pole pro čtení e-mailové zprávy, anebo několik vstupních polí pro psaní e-mailové zprávy. Pro každé prostředí je vytvořen samostatný rámec.

Prostředí pro čtení e-mailové zprávy je tvořeno prvky: *Label* a *ScrolledText*. *Label* slouží jako popisek. Prvek *ScrolledText* představuje textové pole s posuvníkem na pravé straně. Textovému poli se stejně jako u seznamu nastavuje výška. Protože se jedná o pole pro čtení e-mailové zprávy, dochází k jeho „uzamčení“, aby v něm obsažený text nemohl být editován.

Prostředí pro psaní e-mailové zprávy obsahuje několik vstupních polí – v knihovně *tkinter* mají název *Entry*. Dále prostředí obsahuje několik *Label* prvků pro popis vstupních polí. Pro samotné psaní e-mailové zprávy byl přidán prvek *ScrolledText*. Vstupní pole jsou celkem dvě, slouží pro vyplnění příjemce a předmětu. Jednotlivé popisky se nacházejí v těsné blízkosti těchto vstupních polí. Na rozdíl od textového pole *ScrolledText* použitého u čtení e-mailové zprávy, nedochází v tomto prostředí k uzamčení editace.

Dílčí prvky každého prostředí jsou uspořádány na svůj vlastní rámec. Prostředí pro čtení e-mailové zprávy zobrazuje obrázek 5.5. Prostředí pro psaní e-mailové zprávy lze vidět na obrázku 5.4.

Obr. 5.4: Prostředí pro psaní e-mailové zprávy.

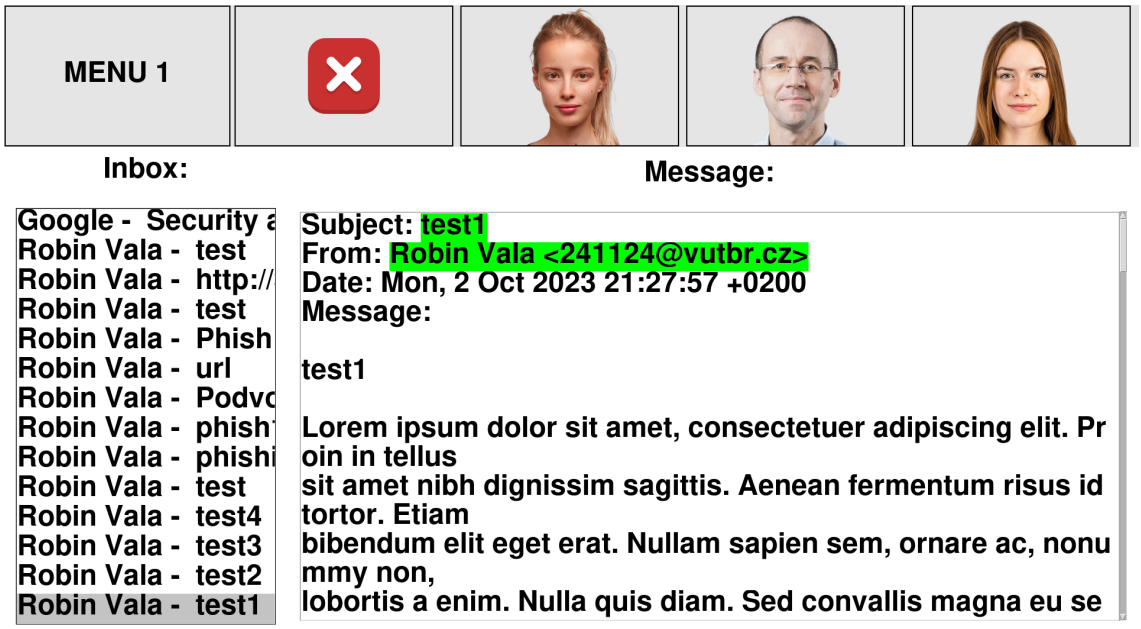
Vložení jednotlivých částí na hlavní rámec aplikace

Hlavní rámec aplikace – *master frame* slouží pro uspořádání všech prvků v prostoru aplikace, tento rámec byl implementován pomocí šablony společně s tlačítky. Pro aplikaci e-mailového klienta byl pak vytvořený nový rámec – *rámec aplikace SMAIL*, který pokrývá plochu pod tlačítky a umožňuje v tomto prostoru zobrazení dalších rámců, které obsahují prvky aplikace. Tyto rámce byly vytvořeny pro jednotlivá prostředí – jednalo se o rámce pro prostředí pro psaní e-mailových zpráv, pro čtení e-mailových zpráv a pro zobrazení doručené pošty. Prostředí pro zobrazení doručené pošty je fixní a při běhu aplikace nemění svoji podobu. Prostředí v pravé části aplikace je proměnné.

Plocha *rámce aplikace SMAIL* má nastavené uspořádání, které lze vidět ve výpisu 5.2. Uspořádání je rozděleno do dvou sloupců, první má váhu 1, druhý má váhu 3. Sloupec s váhou 1 je umístěn v levé části aplikace a obsahuje rámec pro zobrazení přijatých zpráv. Ve sloupci s váhou 3 jsou zobrazována proměnlivá prostředí. Výslednou podobu grafického uživatelského rozhraní lze vidět na obrázku 5.5.

Výpis 5.2: Konfigurace uspořádání rámců v aplikaci.

```
1 self.columnconfigure(0, weight=1, uniform="a")
2 self.columnconfigure(1, weight=3, uniform="a")
```



Obr. 5.5: Grafické uživatelské rozhraní. Obrázky dostupné z: [50, 51, 52].

5.2 Princip ovládání aplikace

Aplikace je ovládána pomocí tlačítek, jejichž grafickou podobu popisuje předchozí kapitola 5.1. Každé z těchto tlačítek má specifickou funkci, která byla nastavena s cílem usnadnit seniorům interakci s aplikací.

Prvním tlačítkem v každé sadě je tlačítko označené jako *MENU*, které slouží pouze k přepínání mezi různými menu. První programovatelné tlačítko v *MENU 1*, reprezentované pomocí bílého křížku na červeném pozadí, bylo vytvořeno za účelem ukončení aplikace, to je zajištěno funkcí *exit_app()*, kterou lze vidět ve výpisu 5.3. Funkce má za úkol zavřít hlavní okno grafického uživatelského rozhraní a následně ukončit proces Pythonu pomocí *SIGKILL*.

Výpis 5.3: Funkce pro ukončení aplikace.

```
1 def exit_app(self):
2     self.master.destroy()
3     subprocess.run(["kill", "-9", str(os.getpid())])
```

Následuje šest tlačítek, která představují jednotlivé členy rodiny seniora, popřípadě jeho přátele a známé. V každém tlačítku je zobrazena fotografie osoby, po kliknutí na tlačítko je změněno rozložení aplikace. V pravé části aplikace se objeví prostředí pro psaní e-mailové zprávy a vstupní pole pro zadání příjemce a předmětu. Pro jednodušší ovladatelnost je vyplněno vstupní pole pro zadání příjemce, a to

e-mailovou adresou zvolené osoby. Po opětovném stisknutí tlačítka dochází k odeslání e-mailové zprávy. Posledním tlačítkem ve druhém menu je textové tlačítko s názvem „Komu“, toto tlačítko umožňuje poslat zprávu na libovolnou e-mailovou adresu, kterou si senior sám zvolí. Funkci, starající se o odeslání e-mailové zprávy, popisuje podkapitola 5.5.

O modifikaci tlačítek se v kódu stará funkce *redefine_template_buttons()*. Tato funkce nejprve zjišťuje počet tlačítek v menu, na základě konfiguračního souboru, a následně vypočítává výšku a šířku každého tlačítka pro uspořádání do menu. Pomocí konfiguračního souboru se také určí, jaký jazyk bude použit v aplikaci. Následně již probíhá konfigurace jak obrázkových tlačítek, tak textových. Tuto konfiguraci zobrazuje výpis 5.4, zahrnuje následující:

- načtení tlačítek z předlohy do lokálních proměnných,
- vyhledání obrázků z konfiguračních souborů a přiřazení obrázků tlačítkům,
- vymazání textu, případně přepsání textu, který byl původně navržený v předloze,
- nastavení šířky tlačítka,
- přiřazení funkce tlačítku,
- přiřazení barvy zobrazované při stisknutí tlačítka.

Výpis 5.4: Modifikace tlačítka.

```
# image configuration
self.person1_image = image_config(
    "Person1", self.six_height)
# access menu 1
self.options_buttons_crt1 = (
    self.menu.menuFrameCreateButtonsVal.optButtons1)
# saving buttons to local variables
self.send_mail_person1 = (
    self.options_buttons_crt1.button_dict[2])
# button redefinition
self.send_mail_person1.config(
    command=lambda: self.fillRecipient(1),
    image=self.person1_image,
    text="",
    width=self.button_width,
    activebackground = active
)
```

5.3 Konfigurace e-mailového klienta

Konfigurace e-mailového klienta je řešena prostřednictvím konfiguračních souborů, které se nachází ve složce *sconf*. Konfigurační soubor *SMAIL_config.json* zajišťuje následující:

- nastavení e-mailového účtu a hesla pro přihlášení k e-mailové schránce,
- nastavení e-mailového účtu opatrovníka seniora,
- nastavení přeposílání e-mailových zpráv na adresu opatrovníka seniora,
- nastavení zobrazování odkazů v e-mailové zprávě,
- nastavení smtp a imap připojení,
- nastavení e-mailových adres a obrázků jednotlivých osob, viz výpis 5.5,
- nastavení zvuků přehrávaných při podržení kurzoru na prvcích v aplikaci a jejich různé jazykové verze,
- nastavení textového překladu jednotlivých prvků v aplikaci.

Výpis 5.5: Nastavení obrázků jednotlivých osob v souboru SMAIL_config.json.

```
"images": {
        "Person1": "../sconf/images/SMAIL_PERSON_1.png",
        "Person2": "../sconf/images/SMAIL_PERSON_2.png",
        "Person3": "../sconf/images/SMAIL_PERSON_3.png",
        "Person4": "../sconf/images/SMAIL_PERSON_4.png",
        "Person5": "../sconf/images/SMAIL_PERSON_5.png",
        "Person6": "../sconf/images/SMAIL_PERSON_6.png"
    }
```

Druhým konfiguračním souborem je *config.json*, který upravuje vzhled více aplikací projektu *Senior-OS*. Aplikace *SMAIL*, využívá především tato nastavení:

- barvu pozadí hlavní aplikace a tlačítek,
- velikost písma, font, tloušťku písma,
- počet tlačítek v menu,
- nastavení monitoru, na kterém se aplikace spustí,
- nastavení jazyka textu a hlasové asistence,
- nastavení barvy upozorňující na bezpečnostní událost,
- nastavení barvy pro zvýraznění prvků v aplikaci,
- nastavení barvy pozadí aplikace,
- nastavení zvukového časovače,
- nastavení fontu.

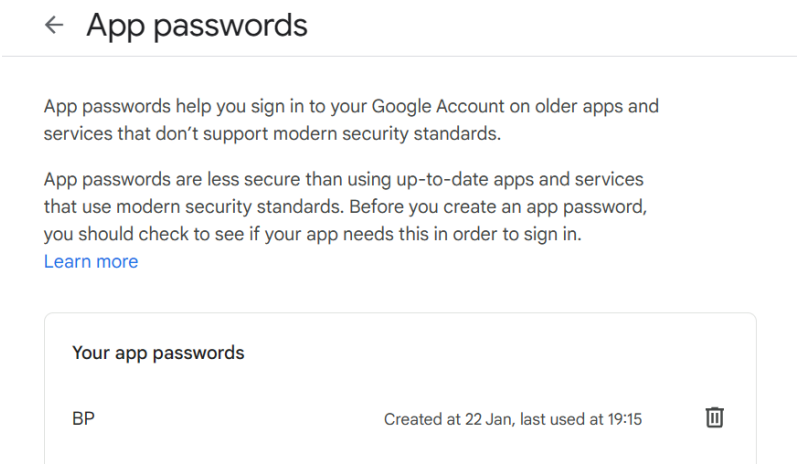
Aplikace pro opatrovníka seniora – *SGIVE* umožňuje hodnoty uložené v těchto souborech editovat pomocí grafického uživatelského rozhraní.

Prvotní vygenerování konfiguračních souborů

Při spuštění aplikace *SMAIL* je provedeno ověření přítomnosti konfiguračních souborů. V případě, kdy tyto soubory nejsou detekovány, systém automaticky generuje nové konfigurační soubory s přednastavenými výchozími parametry. Proces generování globálního konfiguračního souboru je realizován s využitím funkce z importované předlohy ve složce *template*. Pro vytvoření konfiguračního souboru aplikace *SMAIL* je použita funkce *smail_config_default()*.

Nastavení e-mailového účtu a hesla

Pro účely testování aplikace a pro simulaci e-mailové schránky seniora byl vytvořen e-mailový účet *Gmail*. Pro připojení k e-mailové schránce z jiného než webového *Gmail* prostředí bylo zapotřebí vygenerovat heslo pro aplikaci. Toto heslo lze získat v nastavení účtu *Google* v záložce zabezpečení, pod sekcí dvoufázové ověření po zvolení možnosti: *hesla aplikací*, viz obrázek 5.6. Možnost generování hesla pro aplikace je dostupná převážně pro starší aplikace nebo služby. Takto získané heslo bylo vloženo do konfiguračního souboru e-mailového klienta.



Obr. 5.6: Vygenerování hesla pro aplikaci [56].

Nastavení e-mailu opatrovníka seniora a přeposílání e-mailů

Konfigurační soubor obsahuje záznam e-mailové adresy opatrovníka seniora a záznam, který aktivuje odeslání doručených e-mailových zpráv ze schránky seniora na adresu opatrovníka – tento záznam je v konfiguračním souboru označen jako „*resend_email*“. Pokud je u tohoto záznamu nastavena hodnota „1“, dochází při načtení aplikace ke zkopírování veškerých přijatých e-mailů ze schránky seniora do těla e-mailové zprávy, která se následně odešle opatrovníkovi.

Nastavení připojení k e-mailovému serveru

Podle dostupných materiálů bylo nakonfigurováno připojení *smtp* a *imap*. Pro připojení k *imap* serveru je použit port 993 s protokolem *SSL*. Pro připojení k *smtp* serveru používá aplikace port 587 s protokolem *TLS* [57].

Nastavení e-mailových adres a obrázků osob

V aplikaci se nachází celkem 6 tlačítek, která reprezentují jednotlivé osoby. Každé osobě je přidělena e-mailová adresa a obrázek. Obrázky reprezentující osoby byly staženy z databáze obrázků [58].

Po stažení prošly obrázky drobnými modifikacemi, jednalo se především o oříznutí a odstranění pozadí. V rámci aplikace se pak obrázky přizpůsobují velikosti tlačítka.

Nastavení jazyka aplikace a hlasové asistence

Aplikace je přeložena do 3 jazyků, jedná se o češtinu, angličtinu a němčinu. Podle nastavení jazyka v konfiguračním souboru se mění překlad jednotlivých prvků v prostředí aplikace.

Pro jednodušší orientaci v aplikaci byla rovněž přidána hlasová asistence, která byla přeložena také do 3 jazyků. Hlasové nahrávky byly vytvořeny pomocí online nástroje pro generování řeči [59].

Nastavení zobrazení odkazů v těle e-mailové zprávy

Konfigurační soubor *SMAIL_config.json* umožňuje vypnutí respektive zapnutí možnosti kliknout na odkaz v těle e-mailové zprávy. Pokud je tato možnost zapnuta, dochází při vkládání obsahu e-mailu do prostředí textového pole k hledání vzorů odkazů. Funkce *mark_email()*, starající se o tento proces, používá regulární výraz, který hledá řetězce začínající s „*http://*“, „*https://*“ nebo „*www.*“ a pomocí metody *mark_and_link_url()* je označuje.

Samotná funkce *mark_and_link_url()*, viz výpis 5.6, provede následující kroky pro každý nalezený výskyt odkazu:

- určí začátek a konec odkazu v textovém poli,
- vytvoří jedinečné jméno značky pro daný odkaz,
- přidá tuto značku k odkazu a nastaví barvu a podtržení,
- naváže událost kliknutí na značku, která spustí otevření prohlížeče s daným odkazem.

Výpis 5.6: Funkce pro označení odkazů v těle e-mailové zprávy.

```
def mark_and_link_url(self, url):
    start_pos = "1.0"
    while True:
        start_index = self.message_area.search(url,
            start_pos, tk.END)
        if not start_index:
            break
        end_index = f"{start_index}+{len(url)}c"
        tag_name = f"clickable_{start_index.replace('.',
            '_')}"
        self.message_area.tag_add(tag_name,
        start_index, end_index)
        self.message_area.tag_config(tag_name,
            foreground="blue", underline=True)
        self.message_area.tag_bind(tag_name,"<Button-1>",
            lambda event, u=url: self.open_browser(event,
            u))

        start_pos = end_index
```

Nastavení barev aplikace

Při importování šablony jsou nastaveny výchozí barvy aplikace, které lze následně měnit pomocí konfiguračního souboru *config.json*. Pomocí aplikace pro opatrovníka seniora lze například změnit barvu, která upozorňuje na bezpečnostní události. Při zobrazení zprávy s nebezpečným odkazem se pak tato barva zobrazí na pozadí tlačítek. Dalším užitečným nastavením je změna barvy pro zvýraznění prvků v aplikaci, která je využita například pro zvýraznění odesílatele a předmětu při čtení přijaté pošty.

Nastavení fontu

Soubor *config.json* obsahuje kromě nastavení rozložení aplikace a barev také specifikaci fontu a jeho velikosti. Tyto informace jsou použité pro přizpůsobení grafického uživatelského rozhraní, což zahrnuje mimo jiné i dynamické úpravy velikostí prvků. Aplikace využívá informace o velikosti fontu a rozlišení obrazovky k přepočtu velikosti polí, čímž zajišťuje, že se rozhraní přizpůsobí na základě rozlišení obrazovky.

5.4 Zobrazení přijatých e-mailových zpráv

Pro zajištění co nejjednoduššího přístupu k přijatým zprávám byl do aplikace přidán levý sloupec, který zobrazuje e-mailové zprávy. V tomto sloupci se nachází seznam, který obsahuje určitý počet zpráv v závislosti na velikosti fontu a velikosti displeje obrazovky. E-maily jsou seřazeny od nejnovějšího k nejstaršímu. V tomto seznamu je možné se pohybovat pomocí kolečka myši nebo pomocí posuvníku. Každá e-mailová zpráva má název začínající jménem odesílatele, za kterým následuje textový název předmětu přijatého e-mailu.

Připojení k IMAP serveru

O připojení k IMAP serveru se stará funkce *imap_connection()*, ve které se sestavuje spojení pomocí modulu *imaplib*. Konkrétně je použita třída *IMAP4_SSL*, která se připojuje přes šifrovaný soket SSL, jak je vidět ve výpisu 5.7. Ve funkci *read_mail()* dále probíhá dekódování jednotlivých e-mailových zpráv ze složky doručená pošta.

Výpis 5.7: Připojení k IMAP serveru.

```
def imap_connection(login, password, imap_server,
imap_port):
    ...
    mail = imaplib.IMAP4_SSL(
        imap_server, imap_port,
        ssl_context=ssl.create_default_context()
    )
    mail.login(login, password)
    logger.info("Successful connection to IMAP server.")
    return mail
```

Před samotným navázáním spojení nejprve funkce *read_mail()* načítá, na základě nastaveného jazyka v konfiguračním souboru, specifické řetězce pro předmět, odesílatele, datum a samotnou zprávu. Následuje samotné volání funkce *imap_connection()* společně s parametry jako je uživatelské jméno, heslo, imap server a port. Funkce *read_mail()* poté zvolí složku „INBOX“ na straně serveru a vyhledá v ní všechny e-mailové zprávy. Poté probíhá získávání obsahu každé e-mailové zprávy, to zahrnuje:

- získání informací o odesílateli,
- získání data odeslání e-mailu,
- získání názvu předmětu a jeho dekódování,
- získání těla e-mailu a jeho dekódování.

Funkce prochází obsah e-mailové zprávy po částech. Nejprve kontroluje typ obsahu a zjišťuje zda se jedná o čistý text nebo HTML. Následně získává kódování pro jednotlivé části. Pokud je k dispozici informace o kódování, dekóduje se tato část e-mailu podle uvedeného kódování *(charset)*. V případě, že informace o kódování nejsou dostupné, použije se knihovna *chardet*, která se pokusí kódování detekovat. Pokud detekce kódování selže, použije se kódování *latin-1*. Proces dekódování zachycuje výpis 5.8.

Výpis 5.8: Dekódování částí e-mailové zprávy.

```
def read_mail():
  ...
  for part in email_message.walk():
      if part.get_content_type() in ["text/plain",
                                     "text/html"]:
          # get charset from the email part
          charset = part.get_content_charset()
          message = part.get_payload(decode=True)
          if charset:
              message_decode = message.decode(charset)
          else:
              # use chardet
              detect_charset = chardet.detect(message)
              char = detect_charset["encoding"]
              if char:
                  message_decode = message.decode(char)
              else:
                  # if charset fails, use latin-1
                  message_decode = message.decode(
                                       "latin-1")
```

Posledním krokem je zkonstruování obsahu e-mailu, kde dochází k přidání získaných informací do textového řetězce. Výsledný řetězec je poté přidán do seznamu e-mailů. Funkce *read_mail()* následně předává tento seznam další funkci.

Vložení e-mailových zpráv do seznamu

Po úspěšném získání zpráv z IMAP serveru dochází k uspořádání zpráv do seznamu v levé části aplikace. K tomu slouží funkce *insert_emails()*. Funkce nejprve získává dekódované e-mailové zprávy od funkce *read_mail()* a následně je řadí v reverzním

pořadí, aby se nové e-maily zobrazovaly na začátku seznamu. Po přeorganizování pořadí zpráv jsou veškeré e-maily vloženy do seznamu.

Pokud je zjištěna změna mezi počtem přijatých zpráv a počtem zpráv zobrazovaných v aplikaci e-mailového klienta nebo se jedná o prvotní načtení e-mailových zpráv, dochází k filtrování obsahu těla jednotlivých e-mailů pomocí funkce *check_email_for_spam()*. Tato funkce je blíže popsaná v kapitole 5.7. Výsledkem filtrace jsou dva seznamy – seznam bezpečných a nebezpečných zpráv, které kromě těla e-mailu obsahují rovněž index reprezentující pořadí přijetí zprávy a informaci identifikující bezpečnost. Oba tyto seznamy jsou následně sloučeny a seřazeny na základě informace o indexu přijetí. Následně dochází ke vložení jednotlivých e-mailů do seznamu, při tomto procesu se vytváří zobrazovaný název, který se skládá z: *jména odesílatele* a *předmětu zprávy*. Každé vložené položce je nakonec přiřazena událost, která zajišťuje zobrazení zvolené zprávy v prostředí pro čtení e-mailové zprávy. Jednotlivé kroky funkce *insert_emails()* zobrazuje výpis 5.9.

Výpis 5.9: Vložení e-mailů do seznamu pomocí funkce insert_emails().

```
1  def insert_emails(self):
2      ...
3      try:
4          self.safe_emails, self.phish_emails =
               check_email_for_spam(self.reversed_list)
5      ...
6      self.all_emails = (self.safe_emails +
7                         self.phish_emails)
8      self.all_emails.sort(key=lambda x: x[1])
9
10     for email_content, index, safe in self.all_emails:
11         ...
12         self.inbox_list.listbox.insert(tk.END, f"{name}
               - {sub}")
13
14     self.inbox_list.listbox.bind("<<ListboxSelect>>",
           self.show_email)
```

Funkce *insert_emails()* je volána každých 10 sekund, jejím primárním cílem je zkontrolovat doručené zprávy na straně serveru a porovnat je se současně uloženými zprávami v e-mailovém klientu. Výhodou této funkce je, že k aktualizaci doručené pošty v prostředí e-mailového klienta dochází pouze v případě zjištění změny, nestává se tedy, že by se seznam doručených zpráv aktualizoval každých 10 sekund, což by způsobovalo viditelné problikávání v prostředí grafického uživatelského rozhraní.

Zobrazení e-mailové zprávy

Po kliknutí na e-mail se na pravé straně aplikace zobrazí celý text e-mailu, který je možné posouvat posuvníkem. V těle e-mailu se nachází „předmět“, „od“, „datum“ a samotná zpráva. O zobrazení e-mailové zprávy v textovém poli se stará funkce *show_email()*, viz výpis 5.10.

Výpis 5.10: Zobrazení e-mailové zprávy v textovém poli.

```
def show_email(self, event):
    ...
    self.switch_to_reading_mail()

    if not self.inbox_list.listbox.curselection():
        if (self.last_selected_index is not None
            and self.last_selected_email is not None):
            self.configure_message_area(
                 self.last_selected_email)
            return

    try:
        selected_index =
           self.inbox_list.listbox.curselection()[0]
    ...
    selected_email = self.all_emails[selected_index]
    if selected_email[2] == "phish":
        self.alert_buttons()
    else: self.stop_alert()

    self.configure_message_area(selected_email[0])
    self.last_selected_index = selected_index
    self.last_selected_email = selected_email[0]
```

Funkce *show_email()* nejprve zajistí přepnutí na prostředí pro čtení e-mailové zprávy pomocí funkce *switch_to_reading_mail()*. Následuje jednoduché rozhodování o tom, jaká e-mailová zpráva se zobrazí. Řeší se zde mimo jiné problém, který nastane pokud při běhu aplikace uživatel klikne mimo seznam přijatých zpráv, v tomto případě by totiž program ztratil přehled o aktuálně zvoleném e-mailu. Tento problém je řešen pomocí ukládání posledního otevřeného e-mailu a jeho indexu do proměnných.

Rozhodování o zobrazeném e-mailu probíhá následovně:

- Funkce zjišťuje, zda-li je vybrán nějaký e-mail ze seznamu.
- Pokud není aktuálně žádný e-mail vybrán, použije se poslední uložený e-mail pomocí proměnných *last_selected_index* a *last_selected_email*.
- Pokud byl uživatelem zobrazený nový e-mail ze seznamu, aktualizují se hodnoty proměnných *last_selected_index* a *last_selected_email*.

V rámci funkce *show_email()* také dochází k rozlišování e-mailových zpráv na bezpečné a nebezpečné. Podle zvoleného e-mailu se pak mění chování aplikace. Pokud dojde ke zvolení e-mailové zprávy, která je označená jako bezpečná, aplikace se chová standardně a nedochází k žádné změně. Pokud dojde ke zvolení e-mailové zprávy, která je označená jako nebezpečná, dojde ke změně vzhledu a chování aplikace, viz kapitola 5.7.

Vložení obsahu e-mailu do textového pole probíhá následovně:

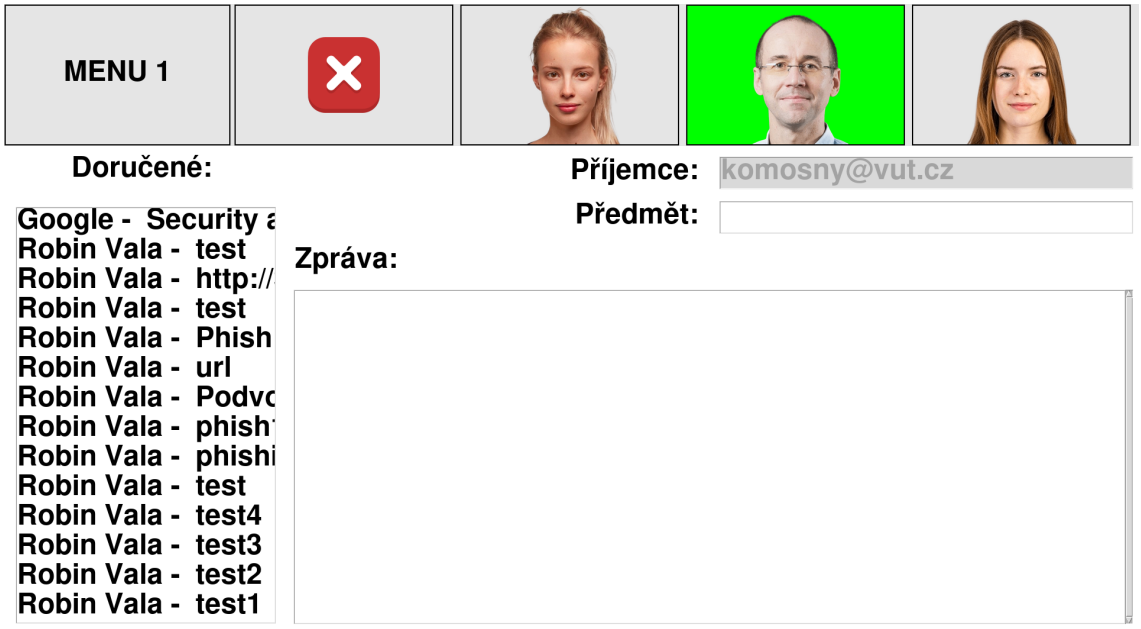
- V základním stavu je textové pole v režimu, kdy nelze měnit jeho obsah.
- Dochází k přepnutí režimu na režim úprav.
- Vymaže se aktuální obsah v textové oblasti.
- Vloží se obsah nově zvoleného e-mailu.
- Dojde k barevnému označení odesílatele e-mailu a předmětu zprávy.
- V těle e-mailu se označkují URL odkazy.
- Textové pole se přepne do režimu, který zakazuje jeho úpravy.

Funkce *mark_important_data()*, jež má na starosti barevné označení odesílatele e-mailu a předmětu zprávy, nejprve zjistí pomocí konfiguračního souboru barvu, která má být použita. Následně prochází vložený obsah zprávy v textovém poli a aplikuje zvolenou barvu na pozadí textu.

Textové pole je rovněž prohledáno pomocí funkce *mark_email()*. Účelem této funkce je najít v textovém poli URL odkazy a označit je. Tato funkce je popsána v podkapitole 5.3.

5.5 Prostředí pro psaní e-mailových zpráv

Pro psaní e-mailové zprávy byla vytvořena samostatná část, která je přepínatelná s částí pro čtení e-mailu. Část pro psaní e-mailové zprávy v sobě obsahuje vstupní pole pro zadání příjemce, předmětu a samotné zprávy. Jednotlivá vstupní pole lze vidět na obr. 5.7. Při kliknutí na fotografii libovolné osoby je zajištěno automatické doplnění e-mailové adresy. Velikost vstupních polí se přizpůsobuje velikosti písma a obrazovce, na které je aplikace spuštěna. Pro odeslání e-mailu stačí vyplnit všechna pole a stisknout tlačítko osoby, které je barevně odlišené.



Obr. 5.7: Prostředí pro psaní e-mailu. Obrázky dostupné z: [50, 51, 52].

Přepnutí na prostředí pro psaní e-mailové zprávy

Přepínání na prostředí pro psaní e-mailových zpráv a odesílání e-mailu má na starost funkce *fill_recipient()*. Při prvním stisknutí tlačítka funkce zjišťuje, zda-li je zobrazeno prostředí pro čtení e-mailových zpráv nebo prostředí pro psaní e-mailových zpráv. Pokud je zobrazeno nesprávné prostředí, funkce prostředí přepne a zobrazí tři vstupní pole pro psaní e-mailové zprávy.

Pokud bylo stisknuto tlačítko s osobou, dochází k automatickému doplnění adresy příjemce. Při opakovaném stisknutí tlačítka proběhne porovnání vyplněných vstupních polí. Pokud jsou všechna pole vyplněna a došlo ke stlačení stejného tlačítka podruhé, odešle se e-mail na adresu osoby zobrazené na tlačítku. Tuto situaci lze

vidět ve výpisu 5.11. Funkce v tomto případě zjišťuje, zda-li jsou vyplněna vstupní pole pro příjemce, předmět a obsah e-mailu. Následně je e-mail odeslán pomocí funkce *send_email_status()*.

Výpis 5.11: Odeslání e-mailu v případě vyplnění všech polí.

```
if (self.recipient_entry.get()
    and self.subject_entry.get()
    and self.content_entry.get("1.0", tk.END).strip()
    and id == self.button_state):
        self.send_email_status()
```

Pokud naopak není alespoň jedno pole vyplněné, dojde k upozornění na nevyplněné pole pomocí zbarvení pozadí. Nedojde k vymazání obsahu polí a senior může pokračovat ve psaní e-mailové zprávy.

Pokud dojde ke stisknutí tlačítka jiné osoby než vepsané do vstupního pole pro zadání příjemce, obsah vstupních polí se vymaže a do políčka adresáta se vloží nová e-mailová adresa. Tuto situaci zachycuje výpis 5.12. Nejprve funkce porovnává, zda-li je vyplněné vstupní pole pro zadání předmětu nebo těla e-mailové zprávy. Dále zjišťuje jestli existuje shoda mezi *id* aktuálně stlačeného tlačítka a předchozím stlačeným tlačítkem. Pokud není zjištěna shoda, dojde k nalezení nové e-mailové adresy v konfiguračním souboru, která se shoduje s osobou na stlačeném tlačítku.

Výpis 5.12: Přepsání adresy přijemce v případě kliknutí na tlačítko jiné osoby.

```
if id != self.button_state:
    email = search_mail(id)
    recipient = self.switch_to_write_mail()
    recipient.delete(0, tk.END)
    recipient.insert(0, email)
    recipient.configure(state="disabled")
```

V případě stisknutí tlačítka „Komu“ nedochází k vyplnění vstupního pole pro zadání adresáta a pro úspěšné odeslání e-mailu musí být vyplněna všechna tři pole. Po prvním stisknutí tohoto tlačítka dojde k vymazání obsahu vstupního pole pro zadání adresáta, jak je vidět ve výpisu 5.13. Při druhém stisknutí tlačítka „Komu“ a při splnění požadavků, dochází k odeslání e-mailové zprávy, jak zachycuje výpis 5.11.

Výpis 5.13: Vymazání vstupního pole pro zadání adresáta.

```
if id == 0 and id != self.button_state:
    recipient = self.switch_to_write_mail()
    recipient.delete(0, tk.END)
```

Funkce *fill_recipient()* tedy volá funkci pro odeslání e-mailové zprávy pouze v případě, kdy je stisknuto stejné tlačítko dvakrát a jsou vyplněna všechna vstupní pole.

Odeslání e-mailové zprávy se děje za pomocí funkce *send_email_status()*. Tato funkce volá funkci *send_email()* a předává ji textové řetězce obsahující e-mailovou adresu příjemce, předmět a samotné tělo zprávy. Pokud předání informací a následné odeslání e-mailu proběhne v pořádku, vymaže se obsah všech vstupních polí. Funkce *send_email_status()* je zobrazena ve výpisu 5.14.

Výpis 5.14: Funkce send_email_status() a předání informací.

```
def send_email_status(self):
    ...
    for recipient in self.recipient_list:
        success = send_email(
            recipient.strip(), self.subject_entry.get(),
            self.content_entry.get("1.0", tk.END),
               login, password, smtp_server, smtp_port)
            if success != 1:
                status = success
        if status == 1:
            self.send_email_success()
```

Připojení k SMTP serveru a odeslání e-mailu

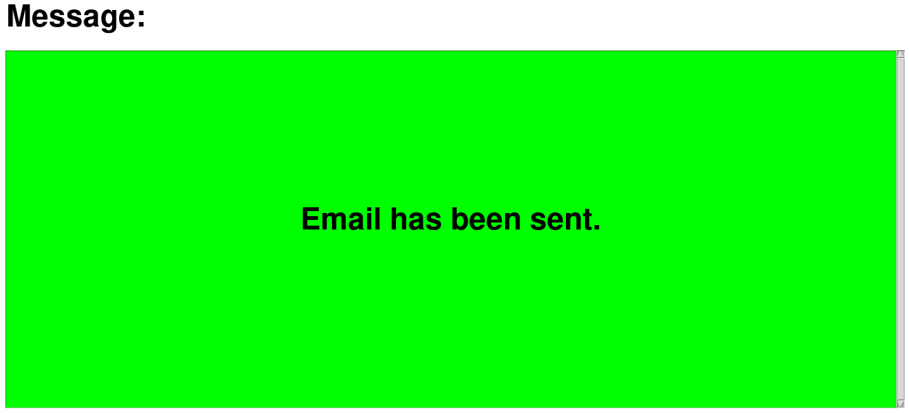
Funkce *send_email()* realizuje připojení k SMTP serveru a následné odeslání e-mailové zprávy pomocí modulu *smtplib*, konkrétně pomocí třídy *SMTP*, která umožňuje zabezpečené spojení pomocí TLS, viz výpis 5.15. Funkce přijímá jako vstupní parametry příjemce, předmět a obsah zprávy. Tyto parametry následně využívá k sestavení objektu e-mailové zprávy pomocí třídy *MIMEText*. Tímto způsobem funkce připravuje e-mailovou zprávu s danými parametry, kterou je poté možné odeslat prostřednictvím SMTP serveru.

Následuje spojení s SMTP serverem, dochází k zadání uživatelského jména a hesla, poté je e-mailová zpráva odeslána. Pokud vše proběhne bez problému, vrací funkce hodnotu „1“.

Výpis 5.15: Připojení k SMTP serveru.

```
def send_email(recipient, subject, content):
    msg = MIMEText(content)
    msg['Subject'] = subject
    msg['From'] = login
    msg['To'] = recipient
    try:
        with smtplib.SMTP(smtp_server, smtp_port
        ) as server:
            server.starttls(context=sslContext)
            server.login(login, password)
            server.sendmail(login, recipient,
                            msg.as_string())
            ...
        return 1
```

V případě, že proces odesílání e-mailu proběhne úspěšně, dojde k vizuální indikaci tohoto stavu změnou barvy textového pole. Současně se v tomto poli zobrazí informační zpráva „E-mail byl úspěšně odeslán“, viz obr. 5.8. Tato změna slouží jako zpětná vazba pro uživatele, signalizující, že akce byla provedena úspěšně a e-mail byl zpracován a odeslán příjemci. Tuto barevnou signalizaci má na starosti funkce *send_email_success()*, strukturu lze vidět ve výpisu 5.16.



Obr. 5.8: Potvrzení odeslání e-mailu.

Na začátku funkce *send_email_success()* nejprve zjišťuje nastavené barvy z konfiguračního souboru a ukládádá je do proměnných *default_color* a *selected_color*. Následně jsou vymazána všechna vstupní pole a do textového pole pro psaní zprávy je, opět pomocí konfiguračního souboru, vložen text informující uživatele o úspěšném odeslání e-mailové zprávy a pozadí je změněno na barvu uloženou v proměnné

selected_color. Posledním krokem funkce je přepnutí pozadí pole na výchozí hodnoty, tedy pomocí využití proměnné *default_color* je nastaveno pozadí zpět na bílou barvu a je vymazána zpráva potvrzující odeslání e-mailu. Tento poslední krok je realizován, po uběhnutí 5 sekund, pomocí funkce *clear_content_entry()*.

Výpis 5.16: Funkce potvrzující úspěšné odeslání e-mailu.

```
def send_email_success(self):
    ...
    # clear all entries
    self.recipient_entry.delete(0, tk.END)
    self.subject_entry.delete(0, tk.END)
    self.content_entry.delete("1.0", tk.END)
    ....
    self.content_entry.insert("end",
        self.text[f"smail_{self.language}_email_sent"])
    self.content_entry.tag_configure("center",
        justify="center")
    self.content_entry.tag_add("center", "1.0", "end")
    self.content_entry.config(bg=selected_color)
    self.content_entry.configure(state="disabled")
    self.master.after(5000, self.clear_content_entry,
        bg_default_color)
```

Přeposlání e-mailu opatrovníkovi při odpovědi na podvodný e-mail

Ve funkci *send_email()* se nachází kontrolní mechanismus, který porovnává e-mailovou adresu příjemce se seznamem adres, z nichž přišly seniorovi podvodné e-maily. Pokud tedy senior pošle e-mailovou zprávu podvodníkovi, odešle se tato zpráva také opatrovníkovi seniora. Proces odeslání e-mailu opatrovníkovi zobrazuje výpis 5.17.

Výpis 5.17: Přeposlání e-mailové zprávy opatrovníkovi seniora.

```
def resend_reply(recipient, content, server):
    content = f"Senior send reply email to phishing email
            ({recipient}) with content:\n" + content
    msg = MIMEText(content)
    msg['Subject'] = f"Reply to phish email by {login}"
    msg['From'] = login
    server.sendmail(login, get_guardian_email(),
            msg.as_string())
```

Nejprve je vytvořen nový obsah zprávy. Před původní text e-mailu se přidá upozornění o tom, že senior odeslal e-mail na podvodnou e-mailovou adresu. V tomto upozornění je také uvedena podvodná e-mailová adresa. Předmět e-mailu je změněn na informační zprávu, která opatrovníkovi oznamuje, ze kterého e-mailu senior zprávu odeslal. Následně se vytvoří nová e-mailová zpráva pomocí třídy *MIMEText*. Nakonec dojde k odeslání e-mailu opatrovníkovi seniora.

Přeposlání veškeré komunikace opatrovníkovi seniora

V případě nastavení hodnoty „1“ u položky *resend_email* v konfiguračním souboru aplikace SMAIL, dochází při spuštění e-mailového klienta k přeposlání veškeré přijaté pošty na adresu opatrovníka seniora. Jako předmět e-mailové zprávy je nastavený řetězec s názvem: „Email report from *emailová adresa seniora*, date: *datum*“. V těle zprávy jsou pod sebou vypsané údaje o jednotlivých doručených e-mailových zprávách společně se samotným obsahem. O celý tento proces se stará funkce *resend_mail_to_guardian()*, kterou lze vidět ve výpisu 5.18.

Výpis 5.18: Funkce pro přeposlání e-mailových zpráv na e-mail opatrovníka seniora.

```
def resend_mail_to_guardian(emails):
    active, smail, gmail = resend_active()
    if active:
        date = datetime.now().strftime("%d.%m.%Y")
        email_subject = f"Email report from {smail},
            date: {date}"
        email_content = ""
        logger.info(f"Sending emails from seniors address
            {smail} to guardians email address {gmail}.")
        for e in emails: email_content += e
        ...
        send_email(gmail, email_subject, email_content,
            login, password, smtp_server, smtp_port)
```

Funkce *resend_mail_to_guardian()* využívá pro načtení hodnoty záznamu *resend_email* a e-mailových účtů seniora a opatrovníka funkci *resend_active()*. V případě nalezení hodnoty „1“ u patřičného záznamu v konfiguračním souboru, dochází k přípravě na odeslání e-mailových zpráv na adresu opatrovníka seniora. Zjišťuje se aktuální datum, které se připojí k předmětu zprávy. Funkce následně prochází všechny přijaté e-mailové zprávy a vytváří řetězec, který je využit jako tělo nového e-mailu. Posledním krokem je odeslání nově vzniklé zprávy na adresu opatrovníka. Výslednou zprávu odeslanou opatrovníkovi zobrazuje obrázek 5.9.

Obr. 5.9: Zpráva odeslaná na e-mailovou adresu opatrovníka.

Tato funkce přeposílající e-mailovou komunikaci může být užitečná v situacích, kdy je senior nesvéprávný a nemůže plně spravovat své záležitosti kvůli omezením spojeným s věkem nebo zdravotním stavem. Pro opatrovníka to může představovat efektivní nástroj k monitorování veškeré e-mailové komunikace seniora, což mu umožňuje chránit jeho zájmy a zasáhnout v případě potřeby.

Přeposílání e-mailů může také hrát zásadní roli při identifikaci potenciálně škodlivých zpráv, které by mohly seniora ohrozit. Tato funkce poskytuje opatrovníkovi průběžný přehled o komunikaci, kterou senior obdržel. To může být důležité, pokud je senior často kontaktován různými organizacemi, které mohou ovlivňovat jeho rozhodování.

5.6 Hlasová asistence

Při vývoji tohoto e-mailového klienta byl brán ohled také na zrakově postižené seniory, pro které je obtížné zorientovat se v prostředí aplikace. Pro tyto případy byly vytvořeny zvukové nahrávky, které napomáhají seniorovi orientovat se, kde se nachází jeho kurzor myši.

Všechny nahrávky byly vytvořené ve třech jazycích, jedná se o češtinu, němčinu a angličtinu. Mechanismus funguje tím způsobem, že po najetí myši na určitý prvek se začne odpočítávat čas. Pokud je čas kurzoru nad prvkem větší než 5 sekund, přehraje se příslušný zvuk. Pokud senior opustí prvek před uběhnutím 5 sekund, časovač se vynuluje a zvuk se nepřehraje. Tento mechanismus je aplikován nejen na jednotlivá tlačítka, ale také na pole, která se nachází pod nimi.

V závislosti na nastaveném jazyce v konfiguračním souboru *config.json*, jsou nastaveny jednotlivé zvukové nahrávky. Přiřazení nahrávek tlačítkům a jednotlivým polím v prostředí aplikace zobrazuje výpis 5.19. Přiřazení těchto zvukových nahrávek má na starosti funkce *audio_configure()*, jejímiž parametry jsou: pole nebo tlačítko, kterému má být nastavena nahrávka a název nahrávky.

Výpis 5.19: Příklad přiřazení zvukové nahrávky.

```
self.audio_configure(self.recipient_entry, "recipient")
self.audio_configure(self.subject_entry, "subject")
self.audio_configure(self.content_entry, "write_message")
```

Funkce *audio_configure()* definuje seznam uchovávající čas vstupu myši na tlačítko nebo prvek a dvě události, jak lze vidět ve výpisu 5.20. První událost je spuštěna, když kurzor myši přejede na tlačítko nebo pole v aplikaci, tato událost volá funkci *button_hover()*. Druhá událost je spuštěna, když kurzor myši opustí dané tlačítko nebo pole, volá se funkce *button_leave()*. Tyto dvě události dále pracují s parametry tlačítka nebo pole a časem vstupu.

Výpis 5.20: Funkce pro přiřazení zvukové nahrávky.

```
def audio_configure(self, button, button_name):
    enter_time = [None]
    button.bind("<Enter>",
                lambda event, name=button_name,
                et=enter_time:
                button_hover(button, name, et))
    button.bind("<Leave>",
                lambda event, et=enter_time:
                button_leave(button, et))
```

Spuštění nahrávky po najetí na prvek v aplikaci

Funkce *button_hover()* nejprve zjišťuje nastavení jazyka aplikace, umístění audia a čas, po kterém se má přehrát zvuková nahrávka. Následně dochází k naplánování události pomocí metody *after* po určeném časovém okamžiku. Identifikátor naplánované události je uložen do proměnné *enter_time[0]*. Pokud dojde k naplnění časového okamžiku vyvolá se funkce *play_sound()*.

Funkce *play_sound()* nejprve inicializuje knihovnu *Pygame*, která slouží pro přehrávání zvuku. Dále zjistí potřebné zvukové informace z konfiguračního souboru a načte zvukový soubor, který odpovídá konkrétnímu prvku a jazyku. Nakonec se přehraje zvuková nahrávka pomocí metody *music_play()*. Funkce *button_hover()* a *play_sound()* jsou zobrazeny ve výpisu 5.21.

Výpis 5.21: Funkce pro spuštění audio nahrávky.

```
def button_hover(button, button_name, enter_time):
    language, audio, timer = get_audio()
    enter_time[0] = button.after(timer,
        lambda: play_sound(button_name))

def play_sound(button_name):
    pygame.mixer.init()
    language, audio, timer = get_audio()
    pygame.mixer.music.load(
        audio[f"smail_{language}_{button_name}"])
    pygame.mixer.music.play()
```

Zrušení přehrávání nahrávky po opuštění prvku

Pokud kurzor myši opustí prvek před uběhnutím časovače, zavolá se funkce *button_leave()*. Tato funkce zkontroluje, jestli je naplánované přehrání zvuku pomocí identifikátoru naplánované události – *enter_time[0]*. Pokud je přehrávání naplánováno, provede jeho zrušení a zvuk se nepřehraje. Funkci *button_leave()* zobrazuje výpis 5.22.

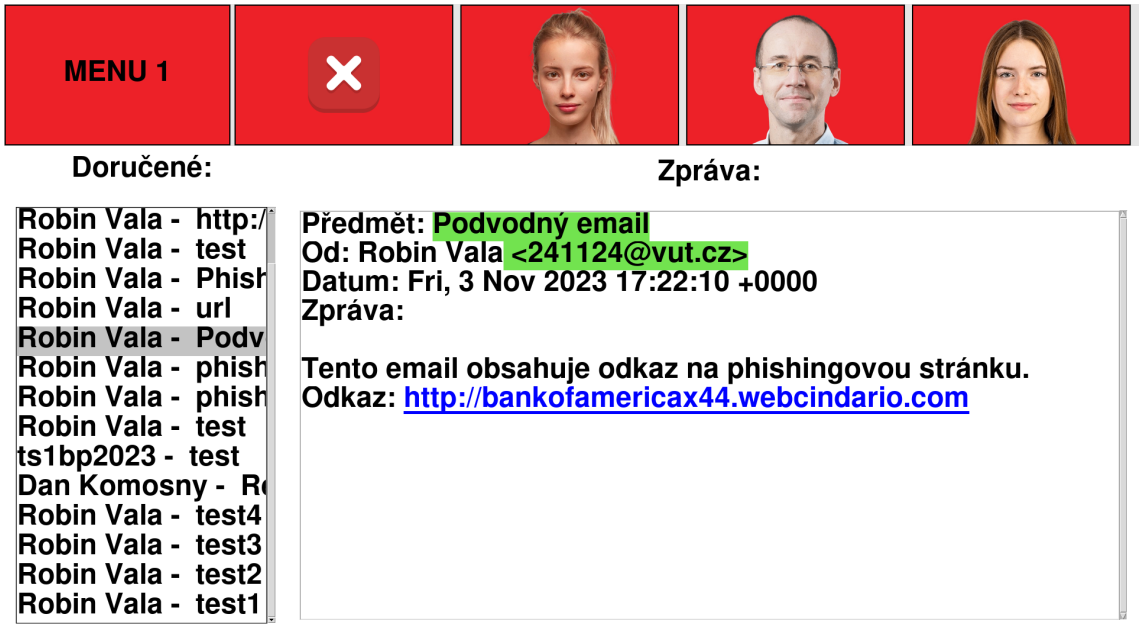
Výpis 5.22: Funkce pro zrušení spuštění audio nahrávky.

```
def button_leave(button, enter_time):
    if enter_time[0]:
        button.after_cancel(enter_time[0])
```

5.7 Ochrana před podvodnými e-maily

Při přijetí e-mailu prochází zpráva procesem kontroly. Obsah těla zprávy je prozkoumán a porovnává se, jestli neobsahuje nebezpečný URL odkaz. Pokud je ve zprávě odkaz nalezen, zapíše se hlášení s úrovní *warning*, toto hlášení je pak viditelné v aplikaci pro opatrovníka seniora. Pokud je zpráva bez odkazu nebo je odkaz neškodný, považuje se za bezpečnou.

Všechny přijaté zprávy jsou zapsány do seznamu, který je zobrazuje v levé části obrazovky. Senior si pak může jednotlivé zprávy přečíst. Pokud klikne na e-mailovou zprávu, která obsahuje nalezený nebezpečný odkaz, přehraje se upozornění „*Pozor, tento e-mail obsahuje nebezpečný odkaz*“ a pozadí tlačítek se změní podle konfiguračního souboru na varovnou barvu, jak ukazuje obrázek 5.10.



Obr. 5.10: E-mail s podvodným odkazem. Obrázky dostupné z: [50, 51, 52].

Pro kontrolu URL odkazů je využívána databáze aktivních podvodných stránek, která je pravidelně aktualizovaná [60]. Databáze je stažena při prvním spuštění aplikace e-mailového klienta pomocí funkce *get_DB()* a ukládá se do společné složky *sconf*. Funkce *get_DB()* se také stará o její pozdější aktualizaci, která probíhá při stáří databáze větší než 14 dnů.

Prozkoumávání obsahu e-mailových zpráv

Před vložením e-mailů do seznamu nejprve probíhá filtrování obsahu jednotlivých zpráv. K tomu slouží funkce *check_email_for_spam()*. Tato funkce definuje dva typy seznamů, do kterých jsou e-mailové zprávy roztříděny. Těmito seznamy jsou:

- *safe_emails* – pro uložení bezpečných e-mailových zpráv,
- *phish_emails* – pro uložení nebezpečných e-mailových zpráv.

Následně funkce prochází každou e-mailovou zprávu a rozděluje její obsah na jednotlivé části. Z jednotlivých částí poté určuje samotnou zprávu a adresu odesílatele, kterou nejprve zbavuje případných závorek, aby byl dosažen čistý formát adresy, viz výpis 5.23. Po úpravách dochází k předání získaných informací funkci *check_content_of_email()*, která provádí kontrolu obsahu e-mailové zprávy.

Výpis 5.23: Získání potřebných informací z těla e-mailové zprávy.

```
def check_email_for_spam(email_messages):
    safe_emails = []
    phish_emails = []
    for email_content in email_messages:
        email_parts = email_content.split("\n")
        sender = email_parts[1].replace("From: ", "")
        message = "".join([s.strip("\r") for s in
           email_parts[4:]])
        if '<' in sender and '>' in sender:
            start_index = sender.find('<')
            end_index = sender.find('>')
            if start_index < end_index:
                modified_sender = sender[start_index +
                                         1:end_index]
        else:
            modified_sender = sender
        contentBlock = check_content_of_email(message,
           modified_sender, email_parts[0])
        ...
```

Funkce *check_content_of_email()* pracuje s databází aktivních podvodných stránek. Z databáze načítá záznamy podvodných URL odkazů do seznamu *phish_urls*. Dále stanovuje regulární výraz pro identifikaci URL adres. Tento regulární výraz je použit pro nalezení všech URL odkazů v obsahu e-mailových zpráv (včetně předmětu zprávy). Pokud dojde k nalezení odkazu ve zprávě, uloží se do seznamu *urls*.

Po zkontrolování celého obsahu e-mailové zprávy dochází k porovnávání nalezených odkazů s podvodnými odkazy, jak lze vidět ve výpisu 5.24. Pokud dojde k nalezení podvodného odkazu, přidá se odesílatel e-mailové zprávy do seznamu podvodných odesílatelů. Funkce nakonec vrací zpět funkci *check_email_for_spam()* hodnotu podle zjištěné závadnosti:

- pokud byla nalezena alespoň jedna podvodná URL adresa, vrací *False*,
- pokud nebyla nalezena žádná závadnost, vrací *True*.

Výpis 5.24: Filtrace podvodných URL odkazů v těle e-mailové zprávy.

```
def check_content_of_email(content, sender, subject):
    ...
    url_pattern = r"https?://(?:www\.)?\S+|www\.\S+"

    content_urls = re.findall(url_pattern, content)
    subject_urls = re.findall(url_pattern, subject)
    urls = content_urls + subject_urls
    found_phishing_url = False

    for url in urls:
        clean_url = url.strip().lower()
        for p in phish_urls:
            if clean_url.startswith(p):
                found_phishing_url = True
                phish_senders.append(sender)
                break

    if found_phishing_url:
        return False
    else:
        return True
```

Funkce *check_email_for_spam()* nakonec provede přiřazení e-mailové zprávy do seznamu na základě získané hodnoty. Do seznamu je kromě samotného obsahu e-mailové zprávy přidán i index, který identifikuje pořadí zprávy a označení bezpečnosti zprávy, viz výpis 5.25. Po vyfiltrování všech e-mailových zpráv dochází k předání obou seznamů funkci, která se stará o umístění e-mailových zpráv do seznamu přijatých zpráv. Vložení a zobrazení e-mailových zpráv popisuje kapitola 5.4.

Výpis 5.25: Přiřazení e-mailové zprávy do seznamu podle bezpečnosti.

```
if contentBlock:
    safe_emails.append((email_content, index, "safe"))
    index = index + 1
else:
    phish_emails.append((email_content, index, "phish"))
    index = index + 1
```

Upozornění seniora na nebezpečnou zprávu

V případě zobrazení nebezpečné zprávy byl do aplikace zahrnut mechanismus, který změní vzhled ovládacích tlačítek a aktivuje hlasové upozornění. Přepnutí vzhledu aplikace je realizováno pomocí funkce *alert_buttons()*. Funkce je volána v momentě, kdy senior klikne na nebezpečnou zprávu v seznamu přijatých zpráv. Změna barvy všech tlačítek na nastavenou barvu probíhá změnou jejich konfigurace, viz výpis 5.26. Funkce *alert_buttons()* taktéž volá funkci *play_sound()*, která spouští zvukové upozornění.

Výpis 5.26: Změna barvy tlačítek v případě zobrazení nebezpečné zprávy.

```
def alert_buttons(self):
    alert_bg = get_alert_color()
    self.exit_button.config(bg = alert_bg)
    self.send_mail_person1.config(bg = alert_bg)
    ...
    play_sound("alert")
```

Pokud následně dojde k zobrazení e-mailové zprávy, která se nenachází v seznamu nebezpečných zpráv, dochází ke zpětné konfiguraci pomocí funkce *stop_alert()*. Funkce načte původní hodnoty barev z konfiguračního souboru a takto získaná data aplikuje v konfiguraci tlačítek obdobným způsobem jako u předešlé funkce. Dochází také ke zrušení přehrávání zvukového upozornění na podvodný odkaz.

Databáze aktivních podvodných stránek

Při každém spuštění aplikace je volána funkce *get_DB()*, která zajišťuje stažení databáze podvodných stránek a její následnou aktualizaci. Nejprve funkce zjišťuje, zda soubor obsahující data s podvodnými odkazy existuje. Pokud soubor existuje proběhne výpočet stáří souboru, získá se datum poslední modifikace a odečte se od aktuálního data. Pokud stáří souboru přesáhne nastavený čas pro aktualizaci souboru nebo soubor ještě nebyl stažen, stáhne se nová aktualizovaná verze databáze.

Pro stažení databáze se používá příkaz uložený v proměnné *command*, který za použití nástroje *wget* stahuje soubor z internetové adresy do místa definovaného pomocí proměnné *file_path*. Tento příkaz je spouštěn v příkazovém řádku pomocí podprocesu, viz výpis 5.27.

Výpis 5.27: Stažení databáze aktivních podvodných stránek.

```
def get_DB():
    file_path = os.path.join(
                os.getcwd().split("smail")[0],
                "sconf/phish/SMAIL_PHISH_1.txt")
    command = f"wget -O {file_path} https://raw.
                githubusercontent.com/mitchellkrogza/
                Phishing.Database/master/
                ALL-phishing-links.txt"
    ...
    try:
        subprocess.run(command, shell=True, check=True,
        executable='/bin/bash', stdout=subprocess.PIPE)
```

5.8 Zaznamenávání vzniklých událostí

V rámci aplikace *SMAIL* byl zaveden systém, který slouží pro upozorňování na události, které nastávají při běhu aplikace. Systém zaznamenává nejen různé stavy aplikace pro zajištění správného fungování, ale i bezpečnostní hrozby v podobě přijetí podvodných e-mailových zpráv do schránky seniora. Záznam událostí je zapisován do souboru *SMAIL.log* ve složce *sconf* a je dále využíván v aplikaci pro opatrovníka seniora. Nastavení upozornění na události je viditelné ve výpisu 5.28.

Výpis 5.28: Nastavení záznamové funkce.

```
logging.basicConfig(
      level=logging.INFO,
      filename=os.path.join(os.getcwd().split("smail")[0],
         "sconf/logs/SMAILlog.log"),
      filemode="a",
      format="%(asctime)s:SMAIL-%(levelname)s-%(funcName)s:
         %(message)s",
      datefmt="%b %d %H:%M:%S",
)
```

V konfiguraci je nejprve nastavena úroveň záznamů na „*INFO*“, což znamená, že budou zapisovány všechny události na úrovni „*INFO*“ a vyšší. Následuje nastavení cesty a názvu záznamového souboru. Pomocí *filemode* je stanoveno, že soubor bude otevřen pro zápis a nový obsah se bude připojovat na konec souboru.
Formát zápisu je složen z několika částí:

- **%(asctime)s** – vkládá čas záznamu,
- **SMAIL-%(levelname)s** – vkládá úroveň záznamu společně s názvem aplikace (pro lepší přehlednost v aplikaci pro opatrovníka seniora),
- **%(funcName)s** – vkládá název funkce, která vedla k vytvoření události,
- **%(message)s** – vkládá zprávu, která popisuje vzniklou událost.

Posledním krokem je nastavení formátu data a času. Nejprve se zobrazuje měsíc, poté den v měsíci a nakonec čas ve formátu: *hodina:minuta:sekunda*.

Při spuštění aplikace *SMAIL* dochází nejprve k ověření stáří záznamového souboru pomocí funkce *check_logfile()*. Tento proces začíná otevřením souboru, aby bylo možné určit datum prvního záznamu. Po získání data funkce porovnává tento údaj s aktuálním datem. Jestliže se zjistí, že soubor neobsahuje záznamy odpovídající současnému datu, funkce soubor smaže ze souborového systému. Následně je vytvořen nový záznamový soubor, který se používá pro zaznamenávání dat aktuálního dne. Díky této funkci je možné každodenně uchovávat záznamy bez potřeby ukládání dat z předchozích dnů.

Zaznamenávání kritických událostí

Záznamy s kritickými událostmi vytváří aplikace v případě, kdy nefunguje požadovaným způsobem a hrozí riziko pádu aplikace. K uložení záznamu s kritickou událostí dochází například v těchto případech:

- aplikace se vůbec nespustí nebo se spustí způsobem, který vyvolá chybu,
- při neúspěšném realizování filtrace obsahu e-mailových zpráv, viz výpis 5.29.

Výpis 5.29: Vyvolání záznamu při kritické události.

```
1 logger.critical("Failed to apply anti-phishing filters."
2                 "Omitting security steps.", exc_info=True))
```

Zaznamenávání chybových událostí

Záznamy s chybovými událostmi vznikají v případě, kdy dojde k chybě při vykonávání funkce. Příklady vzniku chybových událostí v aplikaci *SMAIL*:

- při nenalezení konfiguračního souboru, viz výpis 5.30,
- při nenalezení požadovaného záznamu v konfiguračním souboru,

- při chybné inicializaci spojení s *SMTP* serverem, včetně špatného zadání přihlašovacích údajů k e-mailovému účtu,
- při chybné inicializaci spojení s *IMAP* serverem.

Výpis 5.30: Vyvolání záznamu při chybové události.

```
logging.error(f"File not found: {file_path}")
```

Zaznamenávání varovných událostí

Záznamy s varovnými událostmi informují o problémových jevech v aplikaci, nejčastěji se jedná o jevy spojené s podvodnými e-mailovými zprávami. Příklady vzniku záznamů s varovnými událostmi:

- nalezení podvodného URL odkazu v obsahu e-mailové zprávy, viz výpis 5.31,
- odeslání e-mailové zprávy na e-mailovou adresu podvodníka.

Výpis 5.31: Vyvolání záznamu při varovné události.

```
logger.warning(f"Found a phishing URL from {sender},
                                        url: {url}")
```

Zaznamenávání informačních událostí

Záznamy s informačními událostmi informují o funkčnosti aplikace. Většinou se jedná o potvrzení úspěšnosti jednotlivých kroků funkce. Mezi zaznamenávané události patří:

- informování o stáří databáze podvodných odkazů a o aktualizaci databáze,
- informování o úspěšném odeslaní e-mailové zprávy,
- informování o úspěšném připojení k *IMAP* serveru, viz výpis 5.32,
- informování o počtu načtených e-mailových zpráv do schránky uživatele,
- informování o nalezení *bezpečného* URL odkazu v těle e-mailové zprávy,
- informování o přeposlání e-mailových zpráv na e-mailovou adresu opatrovníka seniora.

Výpis 5.32: Vyvolání informačního záznamu.

```
logger.info("Successful connection to IMAP server.")
```

Výsledná podoba souboru se záznamy

Do souboru *SMAIL.log* se zaznamenávají všechny události po celou dobu běhu aplikace. Soubor obsahuje události všech úrovní počínaje úrovní „INFO“, tyto události jsou řazeny podle času pod sebou na řádcích. Výslednou podobu tohoto souboru ukazuje obr. 5.11.

```
May 05 22:19:30:SMAIL-INFO-get_DB: Phishing database file is 5 days, 23 hours, and 37 minutes old.
May 05 22:19:32:SMAIL-INFO-redefine_template_buttons: Buttons successfully redefined.
May 05 22:19:33:SMAIL-INFO-imap_connection: Successful connection to IMAP server.
May 05 22:19:35:SMAIL-INFO-read_mail: 13 emails successfully loaded
May 05 22:19:36:SMAIL-WARNING-check_content_of_email: Found a phishing URL from robin.valu@seznam.cz
May 05 22:19:36:SMAIL-INFO-check_content_of_email: Found URL from robin.vala02@gmail.com
May 05 22:19:36:SMAIL-WARNING-check_content_of_email: Found a phishing URL from 241124@vut.cz
May 05 22:19:36:SMAIL-INFO-check_content_of_email: Found URL from 241124@vutbr.cz
```

Obr. 5.11: Soubor se záznamy událostí.

6 Testování e-mailového klienta

Cílem této praktické části bylo zanalyzovat a otestovat e-mailový klient za účelem identifikace potenciálních chyb, slabých míst a nedostatků v jeho funkčnosti. Testování bylo realizováno jak autorem, tak dobrovolnými uživateli.

6.1 Jednotkové testování

Jednotkové testování bylo zaměřeno na ověření spolehlivosti aplikace při různých situacích, které mohou při běhu aplikace nastat. Byly zkoumány tři klíčové funkce aplikace: *send_email()*, *imap_connection()* a *load_json_file()*. Hlavním úkolem testů bylo zjistit, zda jednotlivé části kódu vykazují očekávané chování a vrací očekávané návratové hodnoty.

Testování funkce send_email()

Tato sekce se věnuje testům, které byly zaměřeny na funkci *send_email()*. Funkce *send_email()* je zodpovědná za odesílání e-mailů pomocí SMTP protokolu. Celkem byly provedeny čtyři testy, každý z nich zkoumal odlišný scénář chování.

Pro všechny testy byly vytvořeny simulace funkcí pomocí dekorátoru `@patch`, které umožnily vytvoření náhradních objektů pro sledování chování těchto funkcí během testování. Prvním náhradním objektem byl `mock_smtp`, který simuloval chování třídy *SMTP*. Tato třída byla zodpovědná za připojení a komunikaci s SMTP serverem. Dalším náhradním objektem byl `mock_logger`, který sloužil jako náhradní objekt pro modul `logger`.

Test úspěšného odeslání e-mailu

Testovací funkce *test_send_email_success()* byla navržena tak, aby ověřila chování funkce *send_email()* při úspěšném odeslání e-mailu přes SMTP server, její strukturu lze vidět ve výpisu 6.1.

Testovací funkce volala funkci *send_email()* se vstupy, jako je příjemce, předmět a obsah e-mailu. Náhradní objekty byly využity k simulaci úspěšné interakce s externími závislostmi, například s SMTP serverem. Po provedení funkce *send_email()* bylo ověřeno, že nedošlo k neočekávaným chybám při zaznamenávání událostí pomocí ověření `mock_logger.error.assert_not_called()`. Dále bylo ověřeno, že funkce *info* byla na objektu `logger` volána právě jednou s očekávanou zprávou, která potvrzovala úspěšné odeslání e-mailu. Testovací funkce také obsahovala ověření, že byla metoda *smtplib* volána s očekávanými parametry *smtp_server* a *smtp_port*,

což zajišťovalo, že funkce *send_email()* vytvořila spojení s SMTP serverem pomocí správných údajů o serveru a portu. Toto ověření bylo realizováno pomocí metody `assert_called_with()` na náhradním objektu `mock_smtp`. Nakonec byl porovnán výsledek volání funkce *send_email()* s očekávaným výsledkem pomocí metody `assertEqual(result, 1)`. Tímto způsobem byla zajištěna validace funkce *send_email()* při úspěšném odesílání e-mailu.

Výpis 6.1: Testování úspěšného odeslání e-mailu.

```
def test_send_email_success(self,mock_logger, mock_smtp):
    result = send_email(self.recipient, self.subject,
                        self.content, self.login,
                        self.password, self.smtp_server,
                        self.smtp_port)
    mock_logger.error.assert_not_called()
    mock_logger.info.assert_called_once_with(f"An email
                    has been sent to {self.recipient}.")
    mock_smtp.assert_called_with(self.smtp_server,
                        self.smtp_port)
    self.assertEqual(result, 1)
```

Test chyby při připojení k SMTP serveru

Testovací funkce *test_smtp_connect_error()* byla navržena k ověření chování funkce *send_email()* v případě chyby připojení k SMTP serveru, kód funkce lze vidět ve výpisu 6.2.

Samotná testovací funkce volala funkci *send_email()* s očekávanými vstupy. Náhradní objekt `mock_smtp` následně simuloval chybné spojení s SMTP serverem pomocí výjimky *SMTPConnectError*. Tento test se navíc zaměřil na ověření, že metoda *error* na objektu `logger` byla volána právě jednou s očekávanou chybovou zprávou a informacemi o vyvolané výjimce. V důsledku chyby připojení k SMTP serveru bylo rovněž ověřeno, že metoda *info* na objektu `logger` nebyla volána, což odpovídá očekávání, že se žádné úspěšné odeslání e-mailu nekonalo. Nakonec byl porovnán výsledek volání funkce *send_email()* s očekávaným výsledkem pomocí metody `assertEqual(result, 0)`, což zajistilo validaci správného zpracování chyby při připojení k SMTP serveru.

Výpis 6.2: Testování chyby při připojení k SMTP serveru.

```
def test_smtp_connect_error(self,mock_logger, mock_smtp):
    result = send_email(self.recipient, self.subject,
                self.content, self.login, self.password,
                self.smtp_server, self.smtp_port)
    mock_smtp.assert_called_once_with(self.smtp_server,
                self.smtp_port)
    mock_logger.error.assert_called_once_with("SMTP
                connection error. Check your SMTP
                server and port.", exc_info=True)
    mock_logger.info.assert_not_called()
    self.assertEqual(result, 0)
```

Test chyby autentizace při připojování k SMTP serveru

Testovací funkce *test_smtp_authentication_error()* byla navržena k ověření reakce funkce *send_email()* na chybu autentizace při pokusu o připojení k SMTP serveru, kód funkce zachycuje výpis 6.3.

Stejně jako v předchozích testech, i zde byla funkce *send_email()* volána s definovanými vstupy. Pro simulaci autentizační chyby byl použit náhradní objekt `mock_smtp`, který vyvolal chybu *SMTPAuthenticationError*. Po volání funkce bylo potvrzeno správné chování metod na objektu `logger`: metoda *error* byla volána s očekávaným popisem chyby, zatímco metoda *info* nebyla volána, protože nedošlo k úspěšnému odeslání e-mailu. Výsledek testu byl porovnán s očekávanou hodnotou -1, což potvrdilo správné zpracování chyby při připojení k SMTP serveru.

Výpis 6.3: Testování chyby autentizace při připojení k SMTP serveru.

```
def test_smtp_authentication_error(self, mock_logger,
mock_smtp):
    result = send_email(self.recipient, self.subject,
            self.content, self.login, self.password,
            self.smtp_server, self.smtp_port)
    mock_smtp.assert_called_once_with(self.smtp_server,
            self.smtp_port)
    mock_logger.error.assert_called_once_with(
            "Authentication error. Check your email and
            password.", exc_info=True)
    mock_logger.info.assert_not_called()
    self.assertEqual(result, -1)
```

Test neočekávané chyby při připojování k SMTP serveru

Poslední testovací funkce, *test_exception_during_send_email()*, byla navržena k ověření funkce *send_email()* v případě, že během odesílání e-mailu dojde k neočekávané výjimce. Kód funkce je zdokumentovaný ve výpisu 6.4.

Testovací funkce volala funkci *send_email()* s konkrétními vstupy. Náhradní objekt `mock_smtp` byl následně využit k simulaci obecné výjimky *Exception* během odesílání e-mailu. Po provedení funkce *send_email()* bylo ověřeno, že metoda *error* na objektu `logger` byla volána právě jednou s očekávanou chybovou zprávou a informacemi o výjimce, která byla vyvolána. Vzhledem k povaze testu nebylo očekáváno volání metody *info*. Nakonec byl porovnán výsledek volání funkce *send_email()* s očekávanou hodnotou -2, což zajistilo validaci správného zpracování výjimky během neúspěšného pokusu o odeslání e-mailu.

Výpis 6.4: Testování neočekávané chyby při odesílání e-mailu.

```
def test_exception_during_send_email(self,mock_logger,
mock_smtp):
    result = send_email(self.recipient, self.subject,
                self.content, self.login, self.password,
                self.smtp_server, self.smtp_port)
    mock_smtp.assert_called_once_with(self.smtp_server,
                self.smtp_port)
    mock_logger.error.assert_called_once_with("Error
                occurred when trying to send email. ",
                exc_info=True)
    mock_logger.info.assert_not_called()
    self.assertEqual(result, -2)
```

Testování funkce imap_connection()

V této části jsou popsány testy pro funkci *imap_connection()*. Každý test byl vytvořen s cílem ověřit správnost a spolehlivost této funkce, která je zodpovědná za navazování spojení s IMAP serverem.

Pro všechny testovací funkce byly pomocí dekorátoru `@patch` vytvořeny simulace dvou funkcí. Prvním náhradním objektem byl `mock_imap4_ssl`, který simuloval třídu *IMAP4_SSL*. Druhým náhradním objektem byl `mock_logger`, který sloužil jako náhradní objekt pro modul `logger`.

Test úspěšného navázání spojení s IMAP serverem

První test byl definován pomocí funkce *test_imap_connection_success()*, která ověřovala správné chování funkce *imap_connection()* při úspěšném navázání spojení s IMAP serverem pomocí SSL. Testovací funkci zachycuje výpis 6.5.

Ve funkci byl nejprve vytvořen náhradní objekt `mock_mail` pomocí `MagicMock()`. Tento objekt byl nastaven jako návratová hodnota volání `mock_imap4_ssl` a umožnil tak simulovat úspěšné navázání spojení s IMAP serverem. Samotná testovací funkce volala funkci *imap_connection()* se vstupy jako je uživatelské jméno, heslo, adresa IMAP serveru a port. Náhradní objekty byly následně využity k simulaci úspěšného navázání spojení s IMAP serverem. Po provedení funkce *imap_connection()* bylo ověřeno, že metoda *IMAP4_SSL* byla volána právě jednou s očekávanými parametry – adresou serveru a portem, včetně parametru *ssl_context=ANY*. Dále bylo ověřeno, že metoda *login* na objektu `mock_mail` byla volána právě jednou s očekávaným uživatelským jménem a heslem. Následně došlo k ověření, že metoda *info* na objektu `logger` byla volána s očekávanou zprávou, která potvrzuje úspěšné navázání spojení s IMAP serverem. Vzhledem k úspěšnému navázání spojení nebylo očekáváno volání metody *error* na objektu `logger`. Nakonec byl porovnán výsledek volání funkce *imap_connection()* s očekávaným výsledkem, v tomto případě odpovídajícím objektu `mock_mail`.

Výpis 6.5: Testování úspěšného navázání spojení s IMAP serverem.

```
def test_imap_connection_success(self, mock_logger,
mock_imap4_ssl):
    mock_mail = MagicMock()
    mock_imap4_ssl.return_value = mock_mail
    result = imap_connection(self.login, self.password,
            self.imap_server, self.imap_port)
    mock_imap4_ssl.assert_called_once_with(
            self.imap_server, self.imap_port,
            ssl_context=ANY)
    mock_mail.login.assert_called_once_with(self.login,
            self.password)
    mock_logger.info.assert_called_once_with("Successful
            connection to IMAP server.")
    mock_logger.error.assert_not_called()
    self.assertEqual(result, mock_mail)
```

Test chyby při navázání spojení s IMAP serverem

Testovací funkce *test_imap_connection_imap_error()* byla určena k ověření chování funkce *imap_connection()* v případě chyby při navázání spojení s IMAP serverem, její kód lze vidět ve výpisu 6.6.

Funkce *imap_connection()* byla volána s definovanými parametry uživatele a serveru. Použití náhradního objektu `mock_imap4_ssl` umožnilo simulaci výjimky *imaplib.IMAP4.error* při pokusu o připojení k IMAP serveru. Po provedení funkce *imap_connection()* bylo zkontrolováno, zda byla metoda *error* na objektu `logger` volána právě jednou s očekávanou chybovou zprávou. Protože došlo k chybě při připojení, volání metody *info* na `logger` nebylo očekáváno. Nakonec byl výsledek funkce *imap_connection()* porovnán s očekávanou hodnotou 0, což ukázalo, že funkce správně identifikovala a zpracovala chybu připojení.

Výpis 6.6: Testování chyby při navázání spojení s IMAP serverem.

```
def test_imap_connection_imap_error(self,mock_logger,
mock_imap4_ssl):
    result = imap_connection(self.login, self.password,
                self.imap_server, self.imap_port)
    mock_imap4_ssl.assert_called_once_with(
                self.imap_server, self.imap_port,
                ssl_context=ANY)
    mock_logger.error.assert_called_once_with(
                "IMAP Error: Failed to connect to the
                IMAP server.", exc_info=True)
    mock_logger.info.assert_not_called()
    self.assertEqual(result, 0)
```

Test chyby spojení s IMAP serverem

Testovací funkce *test_imap_connection_connection_error()* byla určena k ověření správného chování funkce *imap_connection()* v případě, že dojde k chybě při navazování spojení s IMAP serverem. Obsah funkce je zobrazen ve výpisu 6.7.

Testovací funkce volala funkci *imap_connection()* s definovanými vstupy. Náhradní objekt `mock_imap4_ssl` byl využit k simulaci výjimky *ConnectionError* při navazování spojení s IMAP serverem. Po volání funkce *imap_connection()* bylo ověřeno, že metoda *error* na objektu `logger` byla volána právě jednou s příslušnou chybovou zprávou a informacemi o vyvolané výjimce, zatímco volání metody *info* nebylo, dle očekávání, provedeno. Nakonec došlo k porovnání výsledku funkce

imap_connection() s očekávanou hodnotou −1, což potvrdilo, že se spojení s IMAP serverem neuskutečnilo.

Výpis 6.7: Testování chyby spojení IMAP.

```
def test_imap_connection_connection_error(self,
mock_logger, mock_imap4_ssl):
    result = imap_connection(self.login, self.password,
                self.imap_server, self.imap_port)
    mock_imap4_ssl.assert_called_once_with(
                self.imap_server, self.imap_port,
                ssl_context=ANY)
    mock_logger.error.assert_called_once_with(
                "Connection Error: Failed to establish
                a connection to the IMAP server.",
                exc_info=True)
    mock_logger.info.assert_not_called()
    self.assertEqual(result, -1)
```

Test neočekávané chyby

Poslední funkce, *test_imap_connection_unexpected_error()*, testovala chování funkce *imap_connection()* při neočekávané chybě spojení s IMAP serverem. Kód testovací funkce zachycuje výpis 6.8.

Stejně jako v předešlých případech, testovací funkce volala *imap_connection()* s konkrétními vstupy. Náhradní objekt `mock_imap4_ssl` byl v tomto případě využit k simulaci neočekávané chyby *Exception* při připojení k IMAP serveru. Po provedení funkce *imap_connection()* bylo ověřeno, že metoda *error* na objektu `logger` byla volána právě jednou s očekávanou chybovou zprávou a informacemi o vyvolané výjimce. Vzhledem k chybě při připojení k IMAP serveru nebylo očekáváno volání metody *info* na objektu `logger`. Nakonec byl porovnán výsledek volání funkce *imap_connection()* s očekávaným výsledkem, který v tomto případě odpovídal hodnotě −2, což indikovalo, že došlo k chybě během připojení k IMAP serveru.

Výpis 6.8: Testování neočekávané chyby spojení IMAP.

```
def test_imap_connection_unexpected_error(self,
mock_logger, mock_imap4_ssl):
    result = imap_connection(self.login, self.password,
            self.imap_server, self.imap_port)
    mock_imap4_ssl.assert_called_once_with(
            self.imap_server, self.imap_port,
            ssl_context=ANY)
    mock_logger.error.assert_called_once_with(
            "An unexpected error occurred.",
            exc_info=True)
    mock_logger.info.assert_not_called()
    self.assertEqual(result, -2)
```

Testování funkce load_json_file()

V této části jsou popsány testy funkce *load_json_file()*, která se v aplikaci *SMAIL* stará o načítání obsahu konfiguračních souborů. Každý test byl vytvořen s cílem ověřit, zda tato funkce správně reaguje na situace, jako je úspěšné načtení souboru, neexistující soubor nebo neočekávaná chyba během čtení.

Pro jednotlivé testy byla prostřednictvím dekorátoru `@patch` vytvořena simulace funkce *open* pomocí objektu `mock_open`, který umožňoval simulovat čtení obsahu souboru. Tato simulace byla nastavena tak, aby funkce *open* vracela předem definovaný obsah JSON souboru. Dále byl vytvořen náhradní objekt `mock_logger`, který sloužil jako náhrada pro modul `logger`.

Test úspěšného načtení JSON souboru

Tento test, ověřující správné chování funkce *load_json_file()* při úspěšném načtení obsahu JSON souboru, byl realizován pomocí funkce *test_load_json_file_success()*, jejíž obsah lze vidět ve výpisu 6.9.

V testovací funkci byla nejprve volána funkce *load_json_file()* s cestou k souboru. Simulace funkce *open* byla využita k načtení obsahu souboru. Po provedení funkce *load_json_file()* došlo k ověření, že funkce *open* byla volána právě jednou s očekávanými parametry, tj. cestou k souboru a režimem „*r*“. Dále byl porovnán výsledek volání funkce *load_json_file()* s očekávaným výsledkem, který v tomto případě odpovídal načtenému obsahu JSON souboru.

Výpis 6.9: Testování úspěšného načtení JSON souboru.

```
def test_load_json_file_success(self, mock_open_file):
    result = load_json_file(self.file_path)
    mock_open_file.assert_called_once_with(
            self.file_path, "r")
    self.assertEqual(result, {"key": "value"})
```

Test chyby při nenalezení JSON souboru

Druhý test, definovaný funkcí *test_load_json_file_success()*, ověřoval správné chování funkce *load_json_file()* v případě, že se JSON soubor nepodaří nalézt, struktura testu je viditelná ve výpisu 6.10.

V testovací funkci byla volána funkce *load_json_file()* s cestou k souboru. Simulace funkce *open* pomocí náhradního objektu `mock_open_file` vyvolala výjimku *FileNotFoundError*, čímž byla simulována situace, kdy cílový soubor není nalezen. Po provedení funkce *load_json_file()* došlo k ověření, že metoda *error* na objektu `logger` byla volána s očekávanou chybovou zprávou o nenalezení cílového souboru. Nakonec byl porovnán výsledek volání funkce *load_json_file()* s očekávaným výsledkem, který v tomto případě odpovídal hodnotě 0, což indikovalo, že cílový soubor nebyl nalezen.

Výpis 6.10: Testování chyby při nenalezení JSON souboru.

```
def test_load_json_file_file_not_found(self,
mock_logger, mock_open_file):
    result = load_json_file(self.file_path)
    mock_open_file.assert_called_once_with(
        self.file_path, "r")
    mock_logger.error.assert_called_once_with(f"File not
        found: {self.file_path}")
    self.assertEqual(result, 0)
```

Test neočekávané chyby

Poslední test ověřoval správné chování funkce *load_json_file()* v případě neočekávané chyby během načítání JSON souboru. Pro účely testování byla vytvořena funkce *test_load_json_file_unexpected_error()*, viz výpis 6.11.

V testovací funkci byla volána funkce *load_json_file()* s cestou k souboru. Simulace funkce *open* pomocí náhradního objektu `mock_open_file` vyvolala obecnou chybu *Exception* při načítání obsahu souboru. Po provedení funkce *load_json_file()*

došlo k ověření, že metoda *error* na objektu `logger` byla volána s očekávanou chybovou zprávou a informacemi o výjimce, která byla vyvolána. Nakonec byl výsledek volání funkce *load_json_file()* porovnán s očekávanou hodnotou −1, což indikovalo, že došlo k neočekávané chybě během načítání obsahu souboru.

Výpis 6.11: Testování neočekávané chyby při při práci s JSON souborem.

```
def test_load_json_file_unexpected_error(self,
mock_logger, mock_open_file):
    result = load_json_file(self.file_path)
    mock_open_file.assert_called_once_with(
        self.file_path, "r")
    mock_logger.error.assert_called_once_with(f"An
        unexpected error occurred while loading data
        from {self.file_path}", exc_info=True)
    self.assertEqual(result, -1)
```

Spouštění a výsledky jednotkových testů

Pro souhrnné provedení jednotkových testů byl v rámci vývoje vytvořen soubor *run_test.py*. Tento skript obsahuje testovací balíček – *suite*, do kterého jsou začleněny všechny jednotkové testy. Skript umožňuje centralizované spouštění všech testů, což zjednodušuje proces ověřování jednotlivých funkcí. Soubor *run_test.py* je společně se soubory obsahujícími jednotkové testy umístěn v podsložce *testing*.

Jednotkové testování potvrdilo, že všechny testované funkce pracují podle očekávání a nebyly zjištěny žádné nedostatky ani pochybnosti o správnosti jejich implementace. Výsledky testování jsou ilustrovány na obrázku 6.1.

```
test_imap_connection_connection_error (test_imap_connection.TestImapConnection.test_imap_connection_connection_error) ... ok
test_imap_connection_imap_error (test_imap_connection.TestImapConnection.test_imap_connection_imap_error) ... ok
test_imap_connection_success (test_imap_connection.TestImapConnection.test_imap_connection_success) ... ok
test_imap_connection_unexpected_error (test_imap_connection.TestImapConnection.test_imap_connection_unexpected_error) ... ok
test_exception_during_send_email (test_send_mail.TestSendEmail.test_exception_during_send_email) ... ok
test_send_email_success (test_send_mail.TestSendEmail.test_send_email_success) ... ok
test_smtp_authentication_error (test_send_mail.TestSendEmail.test_smtp_authentication_error) ... ok
test_smtp_connect_error (test_send_mail.TestSendEmail.test_smtp_connect_error) ... ok
test_load_json_file_file_not_found (test_json_file.TestLoadJsonFile.test_load_json_file_file_not_found) ... ok
test_load_json_file_success (test_json_file.TestLoadJsonFile.test_load_json_file_success) ... ok
test_load_json_file_unexpected_error (test_json_file.TestLoadJsonFile.test_load_json_file_unexpected_error) ... ok

----------------------------------------------------------------------
Ran 11 tests in 0.123s

OK
```

Obr. 6.1: Výsledek jednotkového testování.

6.2 Testování načítání aplikace

Proces načítání aplikace hraje důležitou roli v celkovém uživatelském zážitku. Uživatelé očekávají rychlé načítání, bez ohledu na různé podmínky, kterým může aplikace čelit. Testování tohoto aspektu aplikace je zásadní pro zajištění optimálního výkonu, spolehlivosti a uživatelské spokojenosti.

Následujících pár řádků se bude věnovat manuálnímu testování, které zkoumá dobu načítání aplikace v různých situacích. Pro každou ze tří testovaných situací (spuštění aplikace, načítání e-mailů do doručené složky a odesílání e-mailů) bylo provedeno 10 měření času. Výsledky těchto měření jsou prezentovány v tabulce 6.1:

Spuštění Aplikace: Měření doby, kterou aplikace potřebuje k načtení aplikace. To zahrnuje inicializaci, načítání základních prvků a připravenost k interakci (bez načtení doručených e-mailových zpráv).

Načtení e-mailových zpráv do doručené složky: Měření doby, kterou aplikace potřebuje k načtení aplikace a zobrazení všech e-mailových zpráv v seznamu doručených zpráv.

Odeslání e-mailové zprávy: Měření doby od kliknutí na tlačítko pro odeslání e-mailové zprávy po kladné potvrzení o odeslání zprávy.

Tab. 6.1: Testování načítání prvků aplikace.

Měření	Spuštění aplikace [s]	Načtení doručených zpráv [s]	Odeslání zprávy [s]
1	1,5	3,8	2,4
2	1,8	4	2,1
3	1,9	4	2
4	1,8	3,4	2,1
5	1,6	3,2	1,6
6	1,5	3,4	1,5
7	1,4	3,4	1,6
8	1,4	4,5	1,5
9	1,8	3,8	1,5
10	1,8	4,1	1,4

6.3 Testování realizované uživateli

Testování realizované uživateli bylo rozděleno do 3 částí. První část se soustředila na testování ovládání aplikace, rychlost načítání aplikace a na hlasovou asistenci, včetně různých jazykových variant. V druhé části se uživatelé seznámili s prostředím pro čtení e-mailové zprávy, s funkcí otevírající webový prohlížeč po kliknutí na odkaz v těle e-mailu a s varováním při zobrazení nebezpečné zprávy. V poslední části bylo uživatelům představeno prostředí pro psaní e-mailové zprávy.

Mezi uživateli byli jak uživatelé bez technických dovedností (celkem 2), tak uživatelé s technickými dovednostmi (celkem 3). Pro každou část testování aplikace byly vytvořeny pokyny, podle kterých uživatelé měli aplikaci vyzkoušet a následně zhodnotit. Nejprve všichni účastnici testování prostudovali pokyny, následovalo samotné vyzkoušení aplikace, kde si uživatelé mohli zároveň vyzkoušet i další funkční prvky aplikace. Nakonec každý účastník vyplnil dotazník.

Návod na použití aplikace

Pro uživatele byl vytvořen dvoustránkový návod, který představuje ovládání aplikace a seznamuje uživatele s její funkcionalitou. Tento návod obsahuje postup pro stažení e-mailového klienta, nainstalování potřebných knihoven a spuštění aplikace. Jsou v něm rovněž popsány jednotlivé operace, pomocí kterých lze aplikaci ovládat. Návod pro ovládání aplikace lze vidět na obr. 6.2 a 6.3.

Dále byly uživatelům poskytnuty informace, jak provést testování. Mezi informacemi byly zahrnuty konkrétní úkoly, které měly být provedeny během testování, včetně odeslání e-mailu, otevření odkazu, testování hlasové asistence, zobrazení varování a další. Cílem těchto návodů bylo poskytnout rámec pro konzistentní a opakovatelné testování, zahrnující klíčové funkce a scénáře, které reflektují reálné uživatelské interakce s aplikací.

Návod na použití aplikace Smail

Stažení repozitáře: • https://github.com/forsenior/senior-os

Instalace aplikace: • nainstalovat knihovny a moduly uvedené v textovém souboru requirements.txt, který se nachází ve složce smail

Spuštění aplikace: • přes integrované vývojové prostředí (IDE) spustit soubor smail_app.py ve složce smail

Ovládání aplikace

2 sady tlačítek:

Hlavní ovládací panel:

- Přepínání mezi jednotlivými menu pomocí tlačítka „MENU“.
- Ukončení aplikace pomocí červeného tlačítka s bílým křížkem.
- Tlačítka osob slouží pro přepnutí na prostředí pro psaní zprávy, po vyplnění všech polí a opětovném stisknutí tlačítka dochází k odeslání e-mailové zprávy.
- Poslední tlačítko v menu 2 funguje podobným způsobem jako předešlá tlačítka, není vyplněno políčko pro zadání příjemce.

Zobrazení přijatých zpráv:

- Pro zobrazení e-mailové zprávy stačí kliknout na libovlonou položku v levém sloupci aplikace
- Vybraná zpráva se zobrazí v pravé části aplikace.

Obr. 6.2: Návod pro ovládání aplikace strana 1.

Návod na použití aplikace Smail

Otevření odkazu z těla e-mailové zprávy:

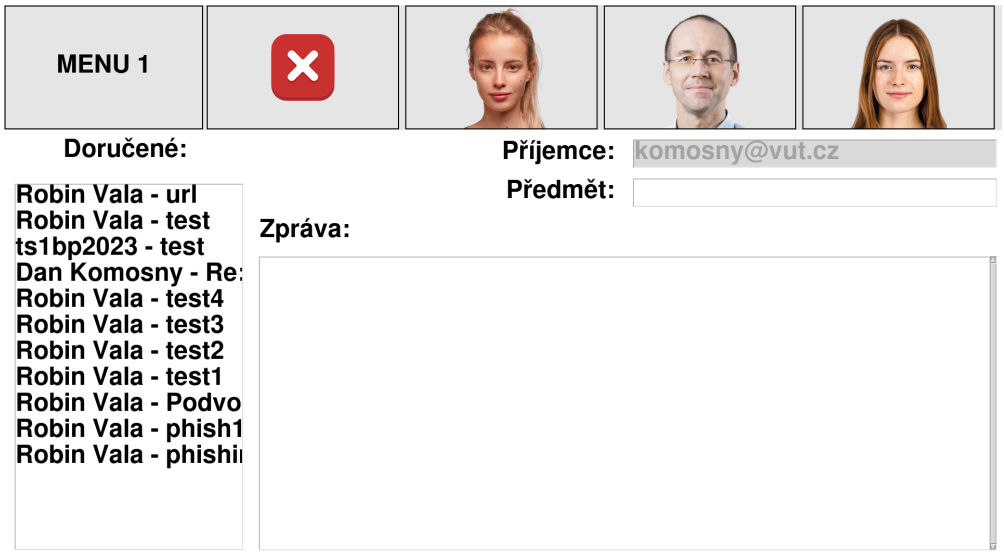
- Pro otevření odkazu stačí kliknout na modrý text v těle zprávy.
- Otevře se webový prohlížeč.

Zpráva:

https://github.com/forsenior/senior-os

Psaní e-mailu:

- Pro psaní e-mailové zprávy stačí kliknout na tlačítko libovolné osoby nebo na tlačítko s názvem „KOMU“.
- Při kliknutí na tlačítko s osobou dojde k automatickému vyplnění políčka pro zadání příjemce.
- Po vyplnění všech polí lze odeslat e-mailovou zprávu pomocí stlačení stejného tlačítka.



Obr. 6.3: Návod pro ovládání aplikace strana 2.

Testování ovládání a rychlosti aplikace

V této části testování uživatelé hodnotili ovládání aplikace, rychlost načtení aplikace a hlasovou asistenci. Hodnocení bylo realizováno pomocí dotazníku, který obsahoval několik otázek věnujících se dané oblasti. Veškeré otázky zahrnuté v dotazníku zobrazuje tabulka 6.2.

Tab. 6.2: Dotazník s výsledky testování ovládání a rychlosti aplikace.

Otázka	Ano	Ne	Slovní odpověď
Bylo z návodu jasné, co každé tlačítko v aplikaci vykonává?	5	0	×
Pokud ne specifikujte, které části návodu byly nejasné, a co by mohlo být vylepšeno.	×	×	–
Jak byste popsali svou zkušenost s ovládáním této aplikace? Působilo ovládání autenticky?	×	×	
Byl čas načítání aplikace přijatelný?	5	0	×
Bylo přepínání mezi různými prostředími v aplikaci plynulé a bezproblémové?	4	1	×
Pokud ne specifikujte, kde a jak často jste zažívali problémy s přepínáním prostředí.	×	×	
Byla hlasová asistence v aplikaci funkční?	5	0	×
Měli jste nějaké obtíže s rozpoznáváním hlasových příkazů nebo nějaké další problémy týkající se hlasové interakce? Pokud ano, poskytněte konkrétní příklady.	×	×	
Celkové hodnocení jazykových překladů (škála 1-5): 1 - Nesrozumitelné, 5 - Výborně srozumitelné	×	×	4×5 1×4
Dodatečné poznámky:	×	×	

Připomínky uživatelů

1. Jak byste popsali svou zkušenost s ovládáním této aplikace? Působilo ovládání autenticky?

- 4×Ano
- Ne, podle mě by bylo užitečné, kdyby existovalo jak zvukové tak vizuální (s textem „odeslat“) oddělení, aby bylo poznat jestli jdu psát zprávu a nebo ji už odesílám, případně, pokud ji odesílám, proč nejde odeslat.

2. Bylo přepínání mezi různými prostředími v aplikaci plynulé a bezproblémové? Pokud ne specifikujte, kde a jak často jste zažívali problémy s přepínáním prostředí.

- Občas trvalo moc dlouho, než se načetly doručené e-maily nebo tlačítka s příjemci.

3. Měli jste nějaké obtíže s rozpoznáváním hlasových příkazů nebo nějaké další problémy týkající se hlasové interakce? Pokud ano, poskytněte konkrétní příklady.

- Některá slova jsou občas zvláštní, např. u tlačítka „odejít“, ale rozumět jim jde dobře. Anglicky je mnohem srozumitelnější.
- Očekávala jsem, že mi v doručených zprávách přečte i text předmětu, protože vždy není vidět celý, případně i od koho je.
- Text není čten pokud uživatel najede na text „Zpráva“ a „Doručené“.
- Problémy jsem neměl, ale některé překlady by mohly být lepší, např. „odejít“ u tlačítka pro opuštění aplikace by mohlo být předěláno na „ukončit aplikaci“, „KOMU“ by šlo nahradit za „Poslat zprávu na vlastní e-mailovou adresu“.

4. Dodatečné poznámky:

- Bylo by vhodné ukončit hlasovou asistenci při překliknutí ze zprávy s podvodným odkazem.
- Při prvotním načtení aplikace se zobrazí prázdné pole „zpráva“ a lze do něj psát, budí to dojem, že uživatel píše e-mailovou zprávu, bylo by vhodné toto pole nezobrazovat vůbec nebo jej alespoň zneplatnit.

Testování prostředí pro čtení e-mailové zprávy

V této části testování uživatelé hodnotili prostředí pro čtení e-mailové zprávy, otevření webového prohlížeče po kliknutí na odkaz v těle e-mailové zprávy a upozornění u nebezpečné zprávy. Hodnocení bylo opět realizováno pomocí dotazníku. Otázky zahrnuté v dotazníku zobrazuje tabulka 6.3.

Tab. 6.3: Dotazník s výsledky testování prostředí pro čtení e-mailové zprávy.

Otázka	Ano	Ne	Slovní odpověď
Bylo z návodu jasné, jak v aplikaci otevřít e-mailovou zprávu ke čtení?	5	0	×
Zobrazil se obsah e-mailové zprávy správně a čitelně?	5	0	×
Jak hodnotíte prostředí pro čtení e-mailových zpráv z pohledu uživatelské přívětivosti a vizuální prezentace e-mailů?	×	×	
Byl text v e-mailové zprávě dostatečně čitelný a měl vhodnou velikost?	5	0	×
Prostor pro připomínky ohledně čitelnosti textu.	×	×	
Bylo možné otevřít odkazy v e-mailové zprávě pomocí prohlížeče?	5	0	×
Prostor pro připomínky týkající se otevírání obsahu v prohlížeči.	×	×	
Jak hodnotíte účinnost funkce upozornění v případě otevření zprávy s nebezpečným odkazem? Považujete tuto funkci za dostatečně srozumitelnou a alarmující?	5	0	×
Prostor pro názor na upozornění o nebezpečné zprávě.	×	×	
Jak vnímáte možnost přeposílání doručené pošty **svéprávného** seniora na e-mail opatrovníka? Považujete tuto funkci za užitečnou či nežádoucí a proč?	×	×	

Pokračování na další straně

Tab. 6.3 – pokračování na další straně

Otázka	Ano	Ne	Slovní odpověď
Jak vnímáte možnost přeposílání doručené pošty **nesvéprávného** seniora na e-mail opatrovníka? Považujete tuto funkci za užitečnou či nežádoucí a proč?	×	×	
Došlo k zobrazení nově přijaté e-mailové zprávy?	5	0	×
Byla nově zachycená e-mailová zpráva ve stejném formátu jako ostatní e-mailové zprávy?	4	1	×
Dodatečné poznámky:	×	×	

Připomínky uživatelů

1. Jak hodnotíte prostředí pro čtení e-mailových zpráv z pohledu uživatelské přívětivosti a vizuální prezentace e-mailů?

- Pro seniora dostačující.
- Ve sloupci doručených e-mailů chybí posuvník k vertikálnímu pohybu výpisu. Drobná pouze vizuální chyba.
- Nešťastné umístění tlačítka pro ukončení aplikace. Otravný a neintuitivní design.
- Aktualizace sloupce doručených e-mailů může některým starším uživatelům působit neklid, nebo jiné zdravotní obtíže. Chybné technické řešení.
- Chybí zvýraznění důležitých částí e-mailu, např. odesílatel, předmět.
- Časové údaje jsou zobrazovány pouze v anglickém jazyce.

2. Připomínky týkající se otevírání obsahu v prohlížeči.

- Při otevírání prvního odkazu bylo trochu zpoždění, ale ostatní fungovaly v pořádku.

3. Názor na upozornění o nebezpečné zprávě.

- Zrušil bych možnost kliknutí na odkaz v těle e-mailové zprávy s nebezpečným e-mailem.
- Senior může na odkaz kliknout dřív než se přehraje hlasové upozornění, dobré by bylo odkaz na nějaký čas zablokovat a potom se znovu zeptat jestli ho chce otevřít v případě, že by se jednalo o špatně vyhodnocený podvodný e-mail.

4. Jak vnímáte možnost přeposílání doručené pošty *svéprávného* seniora na e-mail opatrovníka? Považujete tuto funkci za užitečnou či nežádoucí a proč?

- Spíš ne, pod možností schválení klidně.
- V případě seniora, který je svéprávný bych tuto funkci považoval za žádoucí pouze v případě, kdyby s tím senior souhlasil.
- Považoval bych to za narušení soukromí seniora, jestliže se jedná o osobní zprávy, které mohou obsahovat citlivé údaje.
- Užitečná, v případě příchozího podvodného e-mailu. K přeposílání veškeré pošty svéprávného seniora by ale nemělo docházet.

5. Jak vnímáte možnost přeposílání doručené pošty *nesvéprávného* seniora na e-mail opatrovníka? Považujete tuto funkci za užitečnou či nežádoucí a proč?

- Pokud ho opatrovník zastupuje v legálních věcech tak by to mohlo být užitečné.
- Ano určitě. Tato funkce by neměla sloužit ke sledování seniora, pouze k jeho dohledu.
- V pořádku, nesvéprávný by neměl mít možnost odpovídat na nebezpečné e-maily.
- Dle mého názoru je tato funkce velmi užitečná.

6. Dodatečné poznámky:

- Přijatá zpráva z portálu *Seznam.cz* neobsahovala vyplněného odesílatele ve sloupci s doručenými e-maily.
- Jak je to s podporou zobrazení e-mailů přijatých od jiných poskytovatelů e-mailových služeb? Námět k prozkoumání.
- Jak jsou ošetřeny odkazy na podvodné stránky v předmětu e-mailu? Námět k prozkoumání.

Testování prostředí pro psaní e-mailové zprávy

V poslední části testování uživatelé hodnotili prostředí pro psaní e-mailové zprávy. Hodnocení bylo rovněž realizováno pomocí dotazníku. Otázky zahrnuté v dotazníku zobrazuje tabulka 6.4.

Tab. 6.4: Dotazník s výsledky testování prostředí pro psaní e-mailové zprávy.

Otázka	Ano	Ne	Slovní odpověď
Bylo z návodu jasné, jak se dostat do prostředí pro psaní e-mailové zprávy?	5	0	×
Bylo z návodu jasné, jak odeslat e-mailovou zprávu?	5	0	×
Byl pokus o odeslání e-mailové zprávy na adresu osoby na obrázku úspěšný?	5	0	×
Byl pokus o odeslání e-mailové zprávy na libovolnou adresu (tlačítko komu) úspěšný?	5	0	×
Prostor pro připomínky týkající se odesílání e-mailů.	×	×	
Hodnocení intuitivnosti prostředí pro psaní e-mailové zprávy? Stupnice 1-5 (1 značí nejméně intuitivní a 5 značí nejvíce intuitivní)	×	×	
Jak hodnotíte možnost přeposílání e-mailové zprávy, kterou napsal senior, opatrovníkovi, v případě, že senior odpovídá na e-mail podvodníka? (škála 1-5); 1 - velmi neúčinná, 5 - velmi užitečná	×	×	
Myslíte si, že je tato funkce nezbytná a přináší hodnotu v kontextu bezpečnosti komunikace seniora prostřednictvím e-mailu?	×	×	
Dodatečné poznámky:	×	×	

Připomínky uživatelů

1. Připomínky týkající se odesílání e-mailů

- V případě načteného doručeného emailu, při vytváření nové zprávy dochází po dvojkliku na políčko předmětu, prostoru pro zprávy nebo odesílatele k přepnutí na původně zobrazovány e-mail. Nová zpráva je tedy zahozena.

- Potřebovala bych rozlišení mezi tím, kdy vstupuji do pole, pro napsání zprávy a kdy odesílám – jak zvukové tak vizuální (text se slovem odeslat). Sice to je řečeno v návodu, ale snadno se ztratí přehled, co dělám, komu zprávu odesílám atd.
- Když píšu e-mail jakékoli osobě na obrázcích, tak překliknutím na jiné tlačítko osoby se zpráva smaže, ale pokud chci komukoli napsat zprávu a pak na to samé tlačítko znovu kliknu, i když chci nakonec použít jinou e-mailovou adresu se e-mail odešle.
- Nejsem upozorněna, proč e-mail nejde odeslat a že se neodeslal, když se o to snažím. Zároveň pak uživatel neví, jestli se zpráva odeslala, nebo se jen zaseklo prostředí.
- Zvážil bych zvýraznění tlačítka komu píšu zprávu.
- Nějaké potvrzení, že zpráva byla opravdu odeslána.
- Při pokusu o odeslání e-mailu na více adres je pokus neúspěšný.

2. Myslíte si, že je funkce přeposílání e-mailové zprávy na adresu opatrovníka nezbytná a přináší hodnotu v kontextu bezpečnosti komunikace seniora prostřednictvím e-mailu?

- Nedostatečná, jsou potřeba další opatření.
- Funkce je užitečná, ale mohlo by se stát, že senior poskytne podvodníkovi např. přístupové údaje do bankovnictví. V tomto případě už by bylo na upozornění pozdě.
- V případě, že senior sdílel své osobní údaje, opatrovník by neměl být schopný zprávu přečíst, jen být upozorněný, že byla odeslaná. Opatrovník by neměl být schopný vidět obsah zprávy pokud mu to senior nedovolí. Každopádně si myslím že celkově by senior neměl být schopný odpovídat na podvodný e-mail předtím, než by konzultoval s opatrovníkem. Tzn. opatrovník by měl být informovaný o tom když se ji snaží poslat, pokud je informován až poté, co ji pošle tak už je pozdě.
- Ano. Lidé v pokročilém věku mají problém rozeznat spravedlivé zacházení. Proto se také aktuálně stávají největším terčem na podvody ve všech směrech moderního světa.
- Pokud by byl e-mail poslán jen opatrovníkovi a ten by musel potvrdit jeho odeslání, byla by funkce účinější, takhle se jen ví, pokud senior nějak sdílel informace, co neměl.
- Senior by neměl mít možnost odpovídat na nebezpečnou zprávu, minimálně by měl být upozorněn. Spíše by neměl mít možnost vůbec.

6.4 Odladění aplikace na základě výsledků testování

Při testování přijímání nových e-mailových zpráv nezobrazil e-mailový klient správně jméno odesílatele ve dvou případech. Prvním případem byl řetězec tvaru „*Od: <email@example.cz>*“, kdy aplikace nedokázala rozeznat jméno odesílatele kvůli ostrým závorkám. Druhým případem byl řetězec tvaru „*Od: "Jméno Příjmení" <email@example.cz>*“, kde aplikace označila jméno a příjmení odesílatele včetně uvozovek. Tyto situace byly vyřešeny pomocí přidání podmínek, které správně identifikují a oddělují jméno odesílatele od zbytečných znaků ve formátovaném řetězci. Tím je zaručeno, že e-mailový klient nyní korektně zobrazuje jména odesílatelů bez ohledu na přítomnost speciálních znaků nebo formátování.

Na základě zpětné vazby z dotazníku byl přidán vertikální posuvník do seznamu doručených zpráv. Původní „*Listbox*“ byl nahrazen entitou „*ScrolledListbox*“, která má posuvník implementován ve výchozím nastavení.

Při běhu aplikace docházelo každých deset sekund k obnovení seznamu přijatých zpráv. Při obnovení seznamu byl viditelný problik způsobený smazáním zpráv a vložením nově načtených. Na základě podnětu z dotazníku byla tato funkce upravena, aby docházelo k obnovení pouze v případě, kdy je přijata nová e-mailová zpráva.

Během testování grafického uživatelského rozhraní byla objevena chyba, která způsobovala, že po kliknutí na e-mailovou zprávu v seznamu přijatých zpráv a následném dvou-kliknutí do libovolného políčka v prostředí pro psaní e-mailové zprávy, došlo ke znovu zobrazení přijaté e-mailové zprávy. Tato chyba byla opravena pomocí implementace proměnné *self.allow_show_email*, která je přepínána na základě spuštěného prostředí. Pokud je aktivní prostředí pro čtení e-mailové zprávy, je proměnná nastavena na hodnotu *True*, což umožňuje funkci „show_email()“ zapamatování posledního zvoleného e-mailu ze seznamu přijatých e-mailů. Pokud dojde k přepnutí na prostředí pro psaní e-mailové zprávy, je hodnota nastavena na *False*, tímto způsobem je zajištěno, že nedojde k zobrazení posledního zvoleného e-mailu v textovém poli.

V původní verzi aplikace byly e-maily vkládány do seznamu přijatých zpráv podle toho, zda se jednalo o bezpečný nebo nebezpečný e-mail. Jako první se zobrazovaly bezpečné e-maily, až poté následovaly podvodné e-maily. V současné verzi jsou e-maily zobrazovány podle data, tzn. ve vrchní části seznamu se zobrazují nejnovější e-mailové zprávy. Došlo ke sloučení seznamů bezpečných a nebezpečných zpráv. Původní seznamy byly doplněny o informaci, zda-li se jedná o bezpečný či nebezpečný e-mail a o pořadové číslo. Na základě těchto informací se pak ve funkci, která má na starosti vložení zpráv do seznamu, přiřadila upozornění na základě typu e-mailové zprávy.

Na základě podnětu z dotazníkového šetření bylo také zjištěno, že aplikace nedokáže detekovat podvodný odkaz v předmětu e-mailu. Na základě tohoto zjištění byla do funkce kontrolující obsah e-mailu *check_content_of_email()* přidána kontrola odkazů v předmětu e-mailu. Bylo rovněž přizpůsobeno porovnávání odkazů tak, aby byl detekován odkaz, který má odlišný formát, např. přidané znaky jako jsou: „\“nebo *libovolný text*.

Pro lepší orientaci seniora v aplikaci byla do prostředí pro psaní e-mailové zprávy přidána funkce, která barevně informuje seniora o jeho možnostech při psaní e-mailové zprávy. Při kliknutí na tlačítko dojde ke zbarvení jeho pozadí, což může být nápomocné pro pozdější rozhodování o odeslání e-mailové zprávy. Pokud senior klikne na tlačítko před vyplněním všech polí je rovněž informován pomocí zbarvení prázdných polí, viz obrázek 6.4. Při splnění všech podmínek pro odeslání e-mailové zprávy a opětovném stisknutí zvýrazněného tlačítka, dojde k vymazání všech polí a k potvrzení odeslání zprávy pomocí zbarvení textového pole pro psaní zprávy. V tomto poli se také objeví text „E-mail byl úspěšně odeslán.“. Dalším vylepšením v této oblasti bylo upravení kódu pro případ, kdy se uživatel snaží odeslat zprávu na více e-mailových adres naráz.

Obr. 6.4: Upozornění v případě nevyplnění pole.

Na základě podnětu z dotazníku bylo do prostředí pro čtení e-mailové zprávy doplněno zvýraznění podstatných informací o přijaté zprávě. Konkrétně došlo ke zvýraznění *předmětu* a *odesílatele* pomocí změny barvy pozadí.

V oblasti hlasové navigace došlo k přidání hlasových nahrávek v případě podržení kurzoru myši nad nápisy *doručené*, *zpráva*, *předmět* a *příjemce*. Rovněž byly předělány hlasové nahrávky dvou prvků a to: Z původního *„Komu“* na *„Poslat zprávu na vlastní adresu.“* a z původního *„Odejít“* na *„Ukončit aplikaci.“*.

6.5 Návrhy na vylepšení e-mailového klienta

E-mailový klient by mohl být v budoucnu upraven jak z funkčního, tak z vizuálního hlediska, aby přinesl seniorům i opatrovníkovi příjemnější zážitek z užívání. Z funkčního hlediska by bylo vhodné se zaměřit na rozšíření stávajících nástrojů pro správu e-mailů. Z vizuálního hlediska by mohlo dojít k modernizaci uživatelského rozhraní, aby bylo intuitivnější a vizuálně přitažlivější.

Implementace ukládání přijatých i odeslaných e-mailových zpráv do databáze by přinesla možnost trvalé archivace komunikace. Na rozdíl od dočasného ukládání zpráv v proměnné by databáze mohla umožnit vyhledávání, třídění a filtraci e-mailů. To by mohlo ulehčit správu e-mailů a umožnilo by to snadný přístup k historii seniorovi komunikace. Navíc by mohla být přidána možnost ukládání odeslaných zpráv, která by dovolila opatrovníkovi seniora lépe monitorovat a chránit seniorovu komunikaci.

Rozšíření funkčnosti klienta o možnost prohlížet si přílohy přímo v aplikaci by výrazně zlepšilo uživatelský komfort.

Předělání klienta na *custom* verzi modulu *tkinter* by umožnilo přechod od standardního vzhledu a chování aplikace k více přívětivějšímu grafickému uživatelskému rozhraní. Tento přístup by zvýšil vizuální atraktivitu aplikace, ale také by umožnil implementaci specifických funkcí a interaktivních prvků, které by mohli být přizpůsobeny potřebám a preferencím seniorů.

7 Zveřejnění kódu a instalace aplikace

Výsledný kód e-mailového klienta *SMAIL* je zveřejněn v GitHub repozitáři pod organizací *forsenior* [61]. Hlavní část kódu se nachází ve složce *smail*. Kód je rozdělen do několika souborů, jednotlivé soubory se nacházejí v příslušných složkách. Ve složce *antiphishing* se nachází soubor *get_DB.py* pro práci s databází podvodných odkazů. Složka *connection* obsahuje soubor *mail_connection.py* pro komunikaci s e-mailovými servery IMAP a SMTP a soubor *command_line_mail.py* s funkcí umožňující odeslat e-mailovou zprávu prostřednictvím příkazového řádku. Ve složce *screens* se nacházejí snímky aplikace. Další je složka *template*, která obsahuje šablony jednotného vzhledu aplikace – soubory *configActions.py* a *guiTemplate.py*. Poslední složkou je *testing*, tato složka obsahuje jednotkové testy pro kritické funkce aplikace *SMAIL*. Mimo složku je pak umístěn soubor *smail_app.py*, který slouží ke spuštění aplikace. Dále zde nalezneme soubor *configuration.py*, který obsahuje výchozí konfiguraci aplikace *SMAIL* a nastavení pro záznamovou funkci, soubor *style.py* s doplňkovými funkcemi aplikace a *layout.py*, obsahující zdrojový kód pro grafické uživatelské rozhraní. Navíc jsou mimo složky umístěné textové soubory *README.md* a *requirements.txt*. Celou strukturu repozitáře zobrazuje obrázek 7.1.

Ve složce *sconf* se nacházejí konfigurační soubory, vytvořené zvukové nahrávky, soubor se záznamy a obrázky použité v tlačítkách.

Name	Last commit message	Last commit date
..		
antiphishing	Code cleanup	last week
connection	code cleanup	last week
screens	add logo screenshot	3 weeks ago
template	UPD confirm sent email	1 hour ago
testing	Updated testing	2 weeks ago
.gitignore	new button logic + logging_v1	5 months ago
README.md	UPD readme.md	3 days ago
configuration.py	UPD confirm sent email	1 hour ago
layout.py	UPD confirm sent email	1 hour ago
requirements.txt	UPD send email from command line	29 days ago
smail_app.py	UPD confirm sent email	1 hour ago
style.py	UPD confirm sent email	1 hour ago

Obr. 7.1: Struktura repozitáře GitHub.

Instalace a spuštění aplikace SMAIL

Aplikace *SMAIL* je optimalizována pro provoz na operačních systémech Linux a vyžaduje Python ve verzích 3.11 nebo 3.12. Instalační proces začíná stažením zdrojového kódu projektu *Senior-os*. Následně je klíčové nainstalovat všechny potřebné závislosti, které umožní bezproblémový běh aplikace. Seznam těchto závislostí je k nalezení ve složce *smail*, v souboru *requirements.txt*. Pro instalaci závislostí lze využít nástroj *pip*, jak je popsáno ve výpisu 7.1. Dále je nutné mít nainstalovaný nástroj *wget*, který je používán pro stahování databáze podvodných odkazů. Po dokončení uvedených kroků je aplikace úspěšně nainstalována a připravena na další konfiguraci.

Výpis 7.1: Stažení repozitáře a instalace závislostí e-mailového klienta.

```
1 git clone https://github.com/forsenior/senior-os
2 cd smail
3 pip install -r requirements.txt
```

Před prvním spuštěním e-mailového klienta je klíčové vytvořit nebo nechat automaticky vygenerovat konfigurační soubory. To lze provést buď přímo v aplikaci *SGIVE*, nebo spuštěním aplikace *SMAIL*, která proces generování konfiguračních souborů zahájí automaticky. Po tomto procesu je nutné vložit e-mailovou adresu a heslo do konfiguračního souboru *SMAIL_config.json*

Pro připojení k Gmail účtu z ne-webového prostředí, jako je aplikace *SMAIL*, je nutné vygenerovat heslo pro aplikaci. Pro získání hesla pro aplikaci lze postupovat následujícím způsobem:

1. V nastavení Google účtu se přistoupí k záložce „Zabezpečení“.
2. V sekci „Jak se přihlašujete do Googlu“ se vybere „Dvoufázové ověření“.
3. V podsekci „Hesla pro aplikace“ se vygeneruje nové heslo.
4. Zvolí se aplikace nebo zařízení, pro které má být heslo vygenerováno.
5. Vygenerované heslo se následně zkopíruje a použije společně s e-mailovou adresou pro konfiguraci v aplikaci *SGIVE*.

Pro spuštění aplikace je vyžadováno otevření vývojového prostředí, například *PyCharm*, navigace do adresáře *smail* a otevření souboru *smail_app.py*. Následně lze aplikaci spustit kliknutím na tlačítko *Spustit* ve vývojovém prostředí. Program lze však spustit i pomocí příkazového řádku, jak ukazuje výpis 7.2.

Výpis 7.2: Spuštění aplikace SMAIL pomocí příkazového řádku.

```
1 cd smail
2 python smail_app.py
```

Závěr

Cílem této bakalářské práce bylo vytvořit uživatelsky přívětivý a snadno ovladatelný e-mailový klient, speciálně navržený pro potřeby seniorů ve věku 90 let a více.

Teoretická část této práce se věnovala popisu e-mailové komunikace, kde byly popsány formáty a struktury e-mailových zpráv, proces přenosu zpráv od odesílatele k příjemci a stručný popis systému DNS a doménových jmen. V teoretické části byla rovněž zkoumána problematika sociálního inženýrství, přičemž hlavní pozornost byla věnována fázím a typům útoků, s důrazem na podvodné e-mailové zprávy a situaci v České republice. Dále byly popsány protokoly používané pro práci s e-maily, jmenovitě protokol *SMTP* pro odesílání pošty, protokol *IMAP* umožňující přístup k e-mailům z různých zařízení, protokol *POP* pro stahování pošty a protokol *MIME* pro odesílání jiných než textových datových souborů e-mailem. Závěr teoretické části se věnoval popisu modulů a knihoven implementovaných do aplikace.

Praktická část bakalářské práce se soustředila na vývoj e-mailového klienta a jeho testování. V rámci práce byl využit programovací jazyk *Python* pro postupný vývoj aplikace. Popis začíná návrhem a implementací grafického uživatelského rozhraní, následuje vysvětlení ovládání klienta prostřednictvím tlačítek a implementace tohoto ovládání. Poté se práce věnuje konfiguraci parametrů. Následně je rozebrán vývoj prostředí pro psaní a čtení e-mailových zpráv, včetně dalších funkcí souvisejících s těmito prostředími. Samotný text také obsahuje sekci o vývoji hlasové asistence pro nedoslýchavé seniory a závěr se zaměřuje na popis bezpečnostních opatření proti podvodným e-mailům a systému pro zaznamenávání událostí, což přispívá k lepší ochraně a přehlednosti o bezpečnostních událostech.

V další sekci textu se práce zabývala testováním vytvořeného e-mailového klienta. Byly popsány jednotkové testy klíčových funkcí aplikace a testování načítání aplikace. Následoval popis testování uživatelů včetně výsledků z dotazníkového šetření. Poslední část praktické části se věnovala odladění aplikace na základě výsledků testování a návrhům na zlepšení.

V závěru bakalářské práce bylo popsáno zveřejnění výsledného kódu a postup při instalaci a spuštění aplikace.

Literatura

[1] BOIARSKY, Carolyn. *The impact of emailing and texting on effective written communication: Changes in reading patterns, convergence of subgenres, confusion between social and business communication.* Online. 2015 IEEE International Professional Communication Conference (IPCC). 2015, s. 1-6. ISBN 978-1-4799-3375-4. Dostupné z: https://doi.org/10.1109/IPCC.2015.7235822. [cit. 2024-01-30].

[2] YASAR, Kinza. *Email.* Online. TECHTARGET. October 2022. Dostupné z: https://www.techtarget.com/whatis/definition/e-mail-electronic-mail-or-email. [cit. 2024-01-30].

[3] MICROSOFT. *Change the message format to HTML, Rich Text Format, or plain text.* Online. Dostupné z: https://support.microsoft.com/en-us/office/change-the-message-format-to-html-rich-text-format-or-plain-text-338a389d-11da-47fe-b693-cf41f792fefa. [cit. 2024-01-31].

[4] CROCKER, Dave. *Internet Mail Architecture.* Online. RFC 5598. 2009. Dostupné z: https://doi.org/10.17487/RFC5598. [cit. 2023-11-09].

[5] ALBITZ, Paul a LIU, Cricket. *DNS and BIND.* Fourth Edition. California: O'Reilly & Associates, 2001. ISBN 0-596-00158-4. Dostupné z: https://archive.org/details/dnsbind00albi. [cit. 2023-11-23].

[6] CZ.NIC. *CZ.NIC - O doménách a DNS.* Online. Dostupné z: https://www.nic.cz/page/312/o-domenach-a-dns/. [cit. 2023-11-09].

[7] SEZNAM. *Typy DNS záznamů | Seznam Nápověda.* Online. Dostupné z: https://napoveda.seznam.cz/cz/typy-dns-zaznamu/. [cit. 2023-11-10].

[8] SALAMA, Ramiz; AL-TURJMAN, Fadi; BHATLA, Shruti a YADAV, Satya Prakash. *Social engineering attack types and prevention techniques- A survey.* Online. 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN). 2023, s. 817-820. ISBN 979-8-3503-3802-7. Dostupné z: https://doi.org/10.1109/CICTN57981.2023.10140957. [cit. 2024-01-28].

[9] OVEH, R.O. a AZIKEN, G.O. *Mitigating Social Engineering Attack: A Focus on the Weak Human Link.* Online. 2022 5th Information Technology for Education and Development (ITED). 2022, s. 1-4. ISBN 978-1-6654-9370-3. Dostupné z: https://doi.org/10.1109/ITED56637.2022.10051202. [cit. 2024-01-28].

[10] AL-YOZBAKY, Rian Sh. a ALANEZI, Mafaz. *Detection and Analyzing Phishing Emails Using NLP Techniques.* Online. In: 2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). Istanbul, 2023, s. 1-6. Dostupné z: https://doi.org/10.1109/HORA58378.2023.10156738. [cit. 2023-11-10].

[11] LEE, Jaeil; LEE, Yongjoon; LEE, Donghwan; KWON, Hyukjin a SHIN, Dongkyoo. *Classification of Attack Types and Analysis of Attack Methods for Profiling Phishing Mail Attack Groups.* Online. IEEE Access. 2021, roč. 9, s. 80866-80872. ISSN 2169-3536. Dostupné z: https://doi.org/10.1109/ACCESS.2021.3084897. [cit. 2023-11-11].

[12] LEONOV, Pavel Y.; VOROBYEV, Alexander V.; EZHOVA, Anastasia A.; KOTELYANETS, Oksana S.; ZAVALISHINA, Aleksandra K. et al. *The Main Social Engineering Techniques Aimed at Hacking Information Systems.* Online. 2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). 2021, s. 0471-0473. ISBN 978-1-7281-7691-8. Dostupné z: https://doi.org/10.1109/USBEREIT51232.2021.9455031. [cit. 2024-01-27].

[13] BAIG, Muhammad Sawood; AHMED, Faisal a MEMON, Ali Mobin. *Spear-Phishing campaigns: Link Vulnerability leads to phishing attacks, Spear-Phishing electronic/UAV communication-scam targeted.* Online. 2021, s. 1-6. ISBN 978-1-6654-9441-0. Dostupné z: https://doi.org/10.1109/ICCIS54243.2021.9676394. [cit. 2024-01-27].

[14] SETH, Prerna a DAMLE, Madhavi. *A Comprehensive Study of Classification of Phishing Attacks with its AI/I Detection.* Online. 2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC). 2022, s. 370-375. ISBN 978-1-6654-5687-6. Dostupné z: https://doi.org/10.1109/IIHC55949.2022.10060305. [cit. 2024-01-27].

[15] STOUFFER, Clare. *20 types of phishing attacks + examples and prevention tips.* Online. Norton. 2022. Dostupné z: https://us.norton.com/blog/online-scams/types-of-phishing. [cit. 2024-01-22].

[16] GOOGLE. *Zabezpečení a soukromí v Gmailu – Centrum bezpečnosti Google.* Online. Dostupné z: https://safety.google/gmail/. [cit. 2023-11-11].

[17] DAVIS, Chris; CHAKRABARTI, Ratula a SIMPSON, Daniel. *Anti-phishing protection in Microsoft 365.* Online. MICROSOFT. 2023. Dostupné z: https://learn.microsoft.com/en-us/microsoft-365/security/office-365-s

ecurity/anti-phishing-protection-about?view=o365-worldwide#anti-phishing-protection-in-eop. [cit. 2023-11-11].

[18] SEZNAM. *Nevyžádaná pošta a SPAM | Seznam Nápověda.* Online. Dostupné z: https://napoveda.seznam.cz/cz/email/nevyzadana-posta-a-spam/. [cit. 2023-11-11].

[19] KUMARAN, Neil. *Understanding Gmail's spam filters.* Online. GOOGLE WORKSPACE. Dostupné z: https://workspace.google.com/blog/identity-and-security/an-overview-of-gmails-spam-filters. [cit. 2023-11-11].

[20] GOOGLE. *Open & download attachments in Gmail.* Online. Dostupné z: https://support.google.com/mail/answer/30719?hl=en&ref_topic=3394652&sjid=16733253694919550592-EU#zippy=. [cit. 2023-11-11].

[21] PIDRMNAOVÁ, Zuzana. *ZPRAVODAJSTVÍ 2022 #nePINdej!.* Online. Policie České republiky. 2022. Dostupné z: https://www.policie.cz/clanek/nepindej.aspx. [cit. 2024-02-04].

[22] TURZOVÁ, Martina. *Policie České republiky – KŘP Ústeckého kraje: Pozor na „Nigerijské dopisy".* Online. Policie České republiky. 2023. Dostupné z: https://www.policie.cz/clanek/pozor-na-nigerijske-dopisy.aspx. [cit. 2024-02-04].

[23] KLENSIN, John C. *Simple Mail Transfer Protocol. Online.* RFC 5321. 2008. Dostupné z: https://doi.org/10.17487/RFC5321. [cit. 2023-11-10].

[24] GEEKSFORGEEKS. *Multipurpose Internet Mail Extension (MIME) Protocol.* Online. 19 Dec, 2023. Dostupné z: https://www.geeksforgeeks.org/multipurpose-internet-mail-extension-mime-protocol/. [cit. 2024-01-23].

[25] BORENSTEIN, Nathaniel a FREED, Ned. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.* Online. RFC 2045. 1996. Dostupné z: https://doi.org/10.17487/RFC2045. [cit. 2024-01-23].

[26] AWATI, Rahul. *MIME (Multipurpose Internet Mail Extensions).* Online. TECHTARGET. October 2021. Dostupné z: https://www.techtarget.com/whatis/definition/MIME-Multi-Purpose-Internet-Mail-Extensions. [cit. 2024-01-23].

[27] MDN. *MIME types (IANA media types).* Online. Mdn web docs. Dec 18, 2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types. [cit. 2024-01-23].

[28] LAWRENCE, Hughes. *Internet E-mail: protocols, standards, and implementation*. Boston: ARTECH HOUSE, 1998. ISBN 0-89006-939-5. Dostupné z: https://archive.org/details/internetemailpro0000hugh. [cit. 2023-11-23].

[29] ROSE, Marshall T. a MYERS, John G. *Post Office Protocol - Version 3*. Online. RFC 1939. 1996. Dostupné z: https://doi.org/10.17487/RFC1939. [cit. 2023-11-10].

[30] POP4. *Post Office Protocol Version 4*. Online. [2003?]. Dostupné z: http://www.pop4.org/. [cit. 2023-11-10]..

[31] MELNIKOV, Alexey a LEIBA, Barry. *Internet Message Access Protocol (IMAP) - Version 4rev2*. Online. RFC 9051. 2021. Dostupné z: https://doi.org/10.17487/RFC9051. [cit. 2023-11-10].

[32] MICROSOFT. *What is the difference between POP and IMAP? - Microsoft Support*. Online. Dostupné z: https://support.microsoft.com/en-au/office/what-is-the-difference-between-pop-and-imap-85c0e47f-931d-4035-b409-af3318b194a8. [cit. 2023-11-10].

[33] AMOS, David. *Python GUI Programming With Tkinter*. Online. REAL PYTHON. Dostupné z: https://realpython.com/python-gui-tkinter/. [cit. 2023-11-11].

[34] CLARK, Jeffrey A. *Tutorial - Pillow (PIL Fork) 10.1.0 documentation*. Online. PILLOW READ THE DOCS. Dostupné z: https://pillow.readthedocs.io/en/stable/handbook/tutorial.html. [cit. 2023-11-10].

[35] PYGAME. *About - pygame wiki*. Online. Dostupné z: https://www.pygame.org/wiki/about. [cit. 2023-11-10].

[36] PYGAME. *Pygame v2.6.0 documentation*. Online. Dostupné z: https://www.pygame.org/docs/ref/mixer.html. [cit. 2023-11-10].

[37] PYTHON SOFTWARE FOUNDATION. *Imaplib — IMAP4 protocol client — Python 3.12.0 documentation*. Online. Dostupné z: https://docs.python.org/3/library/imaplib.html. [cit. 2023-11-10].

[38] PYTHON SOFTWARE FOUNDATION. *Smtplib — SMTP protocol client — Python 3.12.0 documentation*. Online. Dostupné z: https://docs.python.org/3/library/smtplib.html. [cit. 2023-11-10].

[39] PYTHON SOFTWARE FOUNDATION. *Email — An email and MIME handling package; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/library/email.html`. [cit. 2023-11-10].

[40] PYTHON SOFTWARE FOUNDATION. *Email.header: Internationalized headers; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/library/email.header.html`. [cit. 2023-11-10].

[41] PYTHON SOFTWARE FOUNDATION. *Os — Miscellaneous operating system interfaces; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/library/os.html#files-and-directories`. [cit. 2023-11-10].

[42] PYTHON SOFTWARE FOUNDATION. *7. Input and Output; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/tutorial/inputoutput.html#tut-files`. [cit. 2023-11-10].

[43] PYTHON SOFTWARE FOUNDATION. *Re — Regular expression operations; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/library/re.html`. [cit. 2023-11-10].

[44] PILGRIM, Mark; BLANCHARD, Dan a CORDASCO, Ian. *Usage; chardet 5.0.0 documentation.* Online. CHARDET READ THE DOCS. Dostupné z: `https://chardet.readthedocs.io/en/latest/usage.html`. [cit. 2023-11-10].

[45] PYTHON SOFTWARE FOUNDATION. *Subprocess — Subprocess management; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/library/subprocess.html`. [cit. 2023-11-10].

[46] SILVEIRA, Otávio Simões. *Python Subprocess: The Simple Beginners Tutorial (2023).* Online. DATAQUEST. Dostupné z: `https://www.dataquest.io/blog/python-subprocess/`. [cit. 2023-11-10].

[47] PYTHON SOFTWARE FOUNDATION. *Logging HOWTO; Python 3.12.0 documentation.* Online. Dostupné z: `https://docs.python.org/3/howto/logging.html#logging-basic-tutorial`. [cit. 2023-11-10].

[48] ANDERSON, Jim. *An Intro to Threading in Python. Online.* REAL PYTHON. Dostupné z: `https://realpython.com/intro-to-python-threading/`. [cit. 2023-11-10].

[49] PYTHON SOFTWARE FOUNDATION. *Unittest — Unit testing framework.* Online. Dostupné z: `https://docs.python.org/3/library/unittest.html`. [cit. 2024-03-30].

[50] MOQACCINO RESOURCES - FLATICON. *Button free icon. Online*. In: Flaticon.com. Dostupné z: https://www.flaticon.com/free-icons/exit. [cit. 2023-12-02].

[51] WAYHOMESTUDIO. *Portrait of young woman wearing striped blouse*. Online. In: Freepik.com. Dostupné z: https://www.freepik.com/free-photo/portrait-young-woman-wearing-striped-blouse_11014693.htm. [cit. 2023-11-22].

[52] FREEPIK. *Portrait of a smiling blonde woman*. Online. In: Freepik.com. Dostupné z: https://www.freepik.com/free-photo/portrait-smiling-blonde-woman_5701128.htm. [cit. 2023-11-22].

[53] NAKARIDORE. *Photo as passport young man with stylish hair*. Online. In: Freepik.com. Dostupné z: https://www.freepik.com/free-photo/photo-as-passport-young-man-with-stylish-hair-cut_10543604.htm. [cit. 2023-11-22].

[54] FREEPIK. *Portrait business man wearing formal suit*. Online. In: Freepik.com. Dostupné z: https://www.freepik.com/free-photo/portrait-business-man-wearing-formal-suit_13698352.htm. [cit. 2023-11-22].

[55] SHURKIN_SON. *Expressive senior woman posing*. Online. In: Freepik.com. Dostupné z: https://www.freepik.com/free-photo/expressive-senior-woman-posing_11196440.htm. [cit. 2023-11-22].

[56] GOOGLE. *Sign in with app passwords*. Online. Gmail Help. Dostupné z: https://support.google.com/mail/answer/185833?hl=en. [cit. 2024-03-21].

[57] GOOGLE. *IMAP, POP, and SMTP*. Online. Google Workspace. Aktualizace 2024-03-05. Dostupné z: https://developers.google.com/gmail/imap/imap-smtp. [cit. 2024-05-15].

[58] FREEPIK COMPANY. *Freepik*. Online. Dostupné z: https://www.freepik.com/. [cit. 2024-03-21].

[59] TTSFREE. *Text To Speech Free*. Online. Dostupné z: https://ttsfree.com/. [cit. 2024-03-21].

[60] KROG, Mitchell a CHABABY, Nissar. *Phishing Domain Database*. Online. GitHub. Dostupné z: https://github.com/mitchellkrogza/Phishing.Database [cit. 2024-03-21].

[61] FORSENIOR. *Senior OS*. Online. GitHub. 2023. Dostupné z: `https://github.com/forsenior/senior-os` [cit. 2024-05-04]

Seznam symbolů a zkratek

DNS	Domain Name System
IMAP	Internet Message Access Protocol
MDA	Mail Delivery Agent
MIME	Multipurpose Internet Mail Extension
MSA	Mail Submission Agent
MTA	Mail Transfer Agent
MS	Message Store
MUA	Mail User Agent
POP3	Post Office Protocol version 3
SMTP	Simple Mail Transfer Protocol

A Obsah elektronické přílohy

Elektronická příloha obsahuje zdrojový kód aplikace *SMAIL* včetně výchozích konfiguračních souborů. Pro běh aplikace je nutné nainstalovat závislosti uvedené v textovém souboru *requirements.txt*, který je umístěný ve složce *smail*. Z důvodu odevzdání pouze kódu aplikace *SMAIL*, není možné spustit webový prohlížeč *SWEB* ani měnit konfiguraci prostřednictvím aplikace *SGIVE*. Konfigurace je možná manuální úpravou přímo v konfiguračních souborech. Pro plné využití schopností e-mailového klienta je nutné stáhnout a nainstalovat celý repozitář z platformy *GitHub*, podle pokynů uvedených v závěru bakalářské práce.

Aplikace je optimalizována pro provoz na operačních systémech Linux a vyžaduje Python ve verzích 3.11 nebo 3.12. Pro správné fungování všech částí je také nutné nainstalovat nástroj *wget*. Aplikace se spouští pomocí souboru *smail_app.py*.

```
senior-os
├── sconf ..... složka s konfigurací projektu
│   ├── audio ..... složka s audio soubory
│   ├── images ..... složka s obrázky
│   ├── logs ..... složka se záznamovými soubory
│   ├── phish ..... složka s databází podvodných stránek
│   ├── config.json ..... globální konfigurační soubor
│   └── SMAIL_config.json ..... konfigurační soubor aplikace SMAIL
├── smail ..... složka se zdrojovým kódem aplikace SMAIL
│   ├── antiphishing
│   │   └── get_DB.py ..... soubor pro práci s databází podvodných odkazů
│   ├── connection
│   │   ├── command_line_mail.py .... soubor pro odeslání e-mailu z příkazové řádky
│   │   └── mail_connection.py ..... soubor pro práci s e-mailovými servery
│   ├── screens ..... složka se snímky obrazovky
│   ├── template ..... složka se soubory implementované šablony
│   │   ├── configActions.py
│   │   └── guiTemplate.py
│   ├── testing ..... složka se soubory jednotkového testování
│   │   ├── run_tests.py
│   │   ├── test_imap_connection.py
│   │   ├── test_json_file.py
│   │   └── test_send_email.py
│   ├── configuration.py ..... soubor obsahující výchozí konfiguraci
│   ├── layout.py ..... soubor se zdrojovým kódem grafického uživatelského rozhraní
│   ├── README.md
│   ├── requirements.txt ..... soubor se závislostmi
│   ├── smail_app.py ..... soubor pro spuštění aplikace SMAIL
│   └── style.py ..... soubor s doplňkovými funkcemi aplikace
└── LICENSE
```