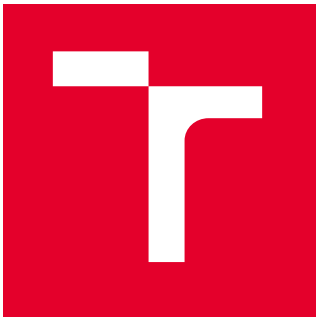


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marek Fiala**

AUTHOR

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

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Marek Fiala ***ID:*** 211477

Ročník: 2 ***Akademický rok:*** 2024/25

NÁZEV TÉMATU:

Emailový klient pro seniory

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte základního emailového klienta, který bude přizpůsoben pro seniory vyššího věku a mentálně znevýhodněné uživatele. Emailový klient bude jednoduše ovladatelný a bude umožňovat spolupráci s opatrovníkem seniora. Implementujte varování při odpovědi seniorem na podvodný email, který žádá o zaslání osobních a jiných citlivých údajů, např. údajů k bankovní kartě. V případě ignorování varování přepošlete text odpovědi opatrovníkovi seniora. Emailového klienta zhotovte v programovacím jazyce Python. Výsledky práce publikujte na repozitáři GitHub pod licencí MIT.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 10.2.2025 ***Termín odevzdání:*** 27.5.2025

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

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Diplomová práce se věnuje návrhu a vývoji e-mailového klienta SMAIL, který je přizpůsoben potřebám seniorů ve věku 90 let a starším. Cílem bylo vytvořit jednoduché a přehledné uživatelské prostředí, které obsahuje vhodné funkce potřebné pro tuto cílovou skupinu. Hlavními vylepšeními oproti předchozí verzi jsou sjednocení uživatelského rozhraní pomocí knihovny PyQt5, zvýšení zabezpečení proti ztrátě rozepsaných e-mailů a přidání validace e-mailových adres. Aplikace dokáže detekovat citlivé údaje, jako jsou hesla nebo čísla platebních karet, a při pokusu o jejich odeslání na ně uživatele vizuálně upozorní. Součástí je také systém vícestupňové ochrany, který podle zvolené úrovně kontroluje například odesílatele a příjemce podle seznamu schválených kontaktů. Tento projekt klade důraz na přístupnost a bezpečnost aplikace. Kompletní zdrojový kód je volně dostupný v repozitáři na platformě GitHub.

KLÍČOVÁ SLOVA

E-mailový klient, SMAIL, Senior, PyQt5, Poetry, SMTP, IMAP

ABSTRACT

The thesis focuses on the design and development of the SMAIL email client tailored to the needs of seniors aged 90 and above. The objective was to create a simple and clear user interface that includes appropriate features required by this target group. The main improvements compared to the previous version include unifying the user interface using the PyQt5 library, enhancing protection against the loss of drafted emails, and adding email address validation.The application is able to detect sensitive information, such as passwords or credit card numbers, and provides a visual warning to users when they attempt to send it. It also incorporates a multi-level protection system that, at higher levels, verifies both the sender and the recipient against a predefined list of approved contacts. This project emphasizes accessibility and security. The complete source code is freely available in a repository on the GitHub platform.

KEYWORDS

E-mail client, SMAIL, Senior, PyQt5, Poetry, SMTP, IMAP

Vysázeno pomocí balíčku `thesis` verze 4.11; `https://latex.fekt.vut.cz/`

FIALA, Marek. *Emailový klient pro seniory*. Diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2025. Vedoucí práce: prof. Ing. Dan Komosný, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Marek Fiala
VUT ID autora:	211477
Typ práce:	Diplomová práce
Akademický rok:	2024/25
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 diplomové práce panu prof. Ing. Danu Komosnému, Ph.D., za odborné vedení, konzultace, trpělivost a cenné rady, které významně přispěly k realizaci této práce. Zvláště si cením důsledného pracovního režimu, který nastavil, a jenž pro mě byl motivací a současně i důvodem, proč mě práce na projektu skutečně bavila a proč se nám jej podařilo úspěšně dotáhnout do konce. Poděkování patří také mým kolegům za příjemnou spolupráci, vzájemnou podporu a sdílení zkušeností při řešení náročných úkolů v rámci společného projektu. A na závěr bych rád poděkoval Standovi za jeho neocenitelnou mentální podporu a pomoc s korekturou, které mi výrazně pomohly zvládnout celé studium i úspěšně dokončit tuto práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Diplomová práce se zaměřuje na návrh a implementaci e-mailového klienta, který je součástí projektu Senior-OS. Tento projekt sdružuje více aplikací navržených pod společnou iniciativou zaměřenou na podporu seniorů vyššího věku a mentálně znevýhodněných uživatelů. Hlavním cílem e-mailového klienta je nabídnout intuitivní a snadno ovladatelné uživatelské rozhraní, které minimalizuje složitost a poskytuje pouze základní funkce nezbytné pro bezpečnou e-mailovou komunikaci.

Jedním z klíčových požadavků je implementace varování při pokusu o odpověď na podvodný e-mail, který žádá o osobní či přístupové údaje. V případě, že uživatel varování ignoruje, aplikace automaticky přepošle text odpovědi opatrovníkovi, čímž přispěje k minimalizaci potenciálních rizik. Výsledná aplikace je vyvíjena v programovacím jazyce Python a zveřejněna na platformě GitHub.

Diplomová práce je rozdělena do osmi kapitol, které podrobně popisují proces vývoje, implementace a e-mailového klienta přizpůsobeného potřebám seniorů. Níže je stručně popsán obsah jednotlivých kapitol.

První kapitola se věnuje popisu současného stavu vývoje aplikace. Nejprve je představena původní verze e-mailového klienta včetně jeho základních funkcí. Následně jsou analyzována omezení, které byly identifikovány při používání předchozí verze. Tato analýza slouží jako základ pro návrh a implementaci vylepšení, která jsou rovněž stručně popsána v této kapitole.

Druhá kapitola se zabývá základy e-mailové komunikace. Vysvětluje klíčové komponenty e-mailu, popisuje nejběžnější formáty používané při práci s e-maily a popisuje proces přenosu zpráv od odesílatele k příjemci. Dále se tato kapitola zaměřuje na přehled e-mailových protokolů. Podrobně jsou zde popsány protokoly Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP), Multipurpose Internet Mail Extensions (MIME) a Post Office Protocol (POP3). Tato kapitola poskytuje nezbytný teoretický rámec, který čtenáři usnadňuje pochopení funkcionality e-mailového klienta.

Třetí kapitola se věnuje aktuálním hrozbám v oblasti e-mailové komunikace. Shrnuje nejčastější typy útoků, jako je phishing a sociální inženýrství. Cílem kapitoly je přiblížit rizika spojená s e-mailovou komunikací a zdůraznit nutnost zavedení ochranných opatření při návrhu e-mailového klienta.

Čtvrtá kapitola představuje technologie a nástroje, které byly použity při vývoji aplikace. Podrobně popisuje výběr programovacího jazyka Python a klíčových knihoven, jako je PyQt5, které byly využity pro návrh grafického uživatelského rozhraní. Kapitola dále popisuje externí nástroj Poetry, který byl použit pro správu závislostí a balíčků, a vysvětluje jeho význam v procesu vývoje. Následně je popsáno používání regulárních výrazů v rámci detekce citlivých informací.

Pátá kapitola se věnuje praktické realizaci e-mailového klienta. Popisuje návrh a implementaci uživatelského rozhraní s důrazem na přívětivost pro seniory. Dále se zabývá funkcemi aplikace, jako je ochrana proti ztrátě rozepsaných e-mailů, upozornění na potenciálně podvodné zprávy a použití úrovně ochrany pro zlepšení bezpečnosti. Kapitola rovněž podrobně rozebírá strukturu projektu, včetně uspořádání adresářů a souborů.

Šestá kapitola obsahuje návody k použití aplikace. Uživatelé zde naleznou podrobné instrukce k používání e-mailového klienta.

Sedmá kapitola shrnuje hlavní výzvy, kterým bylo čeleno v průběhu vývoje aplikace. Popisuje například problémy spojené s přechodem z knihovny `Tkinter` na `PyQt5`, návrh správné logiky ovládání tlačítek, implementaci periodického načítání e-mailů bez blokování uživatelského rozhraní nebo odstranění skrytých cyklických závislostí v kódu.

Osmá kapitola se zaměřuje na distribuci aplikace. Popisuje proces zveřejnění zdrojového kódu na platformě GitHub a uvádí kroky potřebné pro zpřístupnění aplikace širší veřejnosti. Jsou zde rovněž popsány výhody zvoleného způsobu distribuce. Kapitola zahrnuje i postup pro generování hesel pro Gmail, který je nezbytný pro propojení aplikace s e-mailovým účtem uživatele.

Závěrečná kapitola shrnuje výsledky práce a hodnotí splnění stanovených cílů.

1 Popis současného stavu vývoje

Tato práce navazuje na předchozí bakalářskou práci „Emailový klient pro seniory“ [31]. Minulý vývoj zahrnoval vytvoření základní verze e-mailového klienta, který byl přizpůsobený pro seniory ve věkové skupině 90 let a více. Hlavním cílem bylo vytvořit uživatelsky přívětivé rozhraní, minimalizovat komplexitu aplikace a zahrnout pouze základní funkce potřebné pro e-mailovou komunikaci seniorů. Klient byl navržen tak, aby mohl fungovat jako součást širšího projektu Senior-OS. Vývoj zahrnoval také prvky, jako jsou hlasová asistence a ochrana proti podvodným (phishingovým) e-mailům.

1.1 Funkce původní aplikace

Původní verze e-mailového klienta zahrnovala několik klíčových funkcí, které byly zaměřeny na zjednodušení používání a zabezpečení e-mailové komunikace:

- **Hlasová asistence:** Aplikace obsahovala zvukovou asistenci, která usnadňovala navigaci v prostředí programu a zlepšovala celkovou uživatelskou zkušenost.
- **Jednoduché ovládání:** Byla zahrnuta podpora pro programovatelná tlačítka, která umožňovala rychlý výběr kontaktů a snadné psaní e-mailů.
- **Ochrana proti podvodným e-mailům:** Implementována ochrana varující uživatele před potenciálně nebezpečnými odkazy na podvodné webové stránky.
- **Podpora spolupráce s opatrovníkem**: Byla zahrnuta možnost přeposlání komunikace opatrovníkovi seniora, včetně zaznamenávání e-mailových adres, ze kterých přišly podvodné zprávy.

1.2 Plánovaná a realizovaná vylepšení

V rámci pokračujícího vývoje aplikace byly identifikovány oblasti, které nabízejí prostor pro rozšíření, zpřesnění a zefektivnění funkcionality s důrazem na udržitelnost, uživatelský komfort a hladkou integraci do systému Senior-OS.

- **Přechod na PyQt5:** Cílem bude sjednotit knihovny použité v celém projektu Senior-OS, což bude zahrnovat nejen aplikaci SMAIL, ale také další komponenty. Tento krok přispěje k jednotnému vzhledu aplikací a snazší údržbě kódu. Díky tomu budou prvky sdíleny mezi aplikacemi, technická složitost bude snížena a vývoj zefektivněn.
- **Ochrana citlivých dat:** Aplikace bude rozšířena o detekci potenciálně citlivých informací v obsahu e-mailu (např. čísla účtů, hesla, rodná čísla apod.). Pokud bude takový obsah rozpoznán, uživatel bude nejprve varován a zpráva

bude odeslána až po jeho potvrzení. Tímto způsobem bude minimalizováno riziko nechtěného sdílení citlivých údajů, což bude zvláště důležité u zranitelných skupin uživatelů.

- **Zabezpečení rozpracovaných e-mailů:** Bude implementována funkce, která zajistí ochranu rozpracovaných e-mailů před nechtěným smazáním. K odstranění rozepsané zprávy bude nutné potvrdit akci dvakrát. Tím bude zvýšena bezpečnost práce a eliminovány náhodné chyby, které by mohly být pro uživatele frustrující.
- **Validace e-mailových adres:** Bude přidána validace e-mailových adres při psaní zpráv, aby se předešlo odesílání na nesprávné nebo neexistující adresy. Tím bude sníženo riziko neúspěšného doručení zpráv a zvýšena celková spolehlivost aplikace.
- **Úrovně ochrany (Protection Level):** Aplikace bude podporovat více úrovní ochrany, které budou upravovat chování rozhraní i dostupné funkce. Například ve vyšších úrovních nebude možné odesílat zprávy neověřeným kontaktům, případně bude automaticky informován opatrovník při pokusu o odeslání citlivých dat. Tím bude možné aplikaci přizpůsobit míře digitální gramotnosti uživatele a poskytnout tak větší kontrolu a bezpečnost různým skupinám seniorů.
- **Integrace s modulem `dataProvider`:** Přístup ke konfiguračním údajům bude realizován skrze jednotný datový modul. Tím bude zvýšena modularita, bezpečnost i přehlednost při práci s konfigurací. Nastavení aplikace tak bude možné měnit bez nutnosti zásahů do zdrojového kódu.
- **Sjednocení pojmů:** V rámci práce bude zavedena jednotná terminologie napříč celým projektem. Tento krok zajistí konzistenci ve všech funkcích a rozhraních aplikace, což usnadní orientaci jak uživatelům, tak i vývojářům. Dále bude provedeno zjednodušení aplikace, které bude mimo jiné zahrnovat odstranění hlasové asistence ve prospěch čistého a vizuálně sjednoceného prostředí optimalizovaného pro snadnou obsluhu.

2 Princip e-mailové komunikace

Elektronická pošta, známá také jako e-mail, představuje formu elektronického zasílání zpráv, která je dnes běžnou součástí každodenní komunikace. Tyto zprávy mohou být přenášeny prostřednictvím přímého spojení mezi dvěma počítači, přes lokální síť (LAN), nebo přes rozlehlé sítě, jako je Internet. E-mail se běžně využívá k doručování zpráv mezi konkrétními e-mailovými schránkami a je založen na dobře definované struktuře a standardech, které umožňují jeho spolehlivý přenos a zpracování[1].

Pro správné doručení a čitelnost e-mailu je klíčová znalost jeho stavebních prvků – hlavičky, těla zprávy a adresy odesílatele i příjemce. Každý z těchto komponentů má svou specifickou roli a bez jejich správného fungování by e-mailová komunikace nebyla možná. Dále hrají roli i různé formáty zpráv, které ovlivňují podobu a chování e-mailu na straně příjemce.

Aby však mohlo ke komunikaci vůbec dojít, je nezbytná přítomnost protokolů, které zajišťují samotný přenos zpráv mezi zařízeními a servery. Tato kapitola proto navazuje přehledem nejdůležitějších e-mailových protokolů, které tvoří páteř moderní elektronické komunikace. Jsou představeny čtyři základní protokoly: Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP), Multipurpose Internet Mail Extensions (MIME) a Post Office Protocol (POP3). Následující části se věnují jejich fungování, specifickým úlohám v rámci přenosu e-mailů a vzájemné spolupráci při správě zpráv napříč různými zařízeními a systémy.

2.1 Základní komponenty e-mailu

E-mailová zpráva se skládá z několika částí, které mají různé funkce a významy. Mezi tyto části patří e-mailová adresa, hlavička a tělo zprávy. Hlavička obsahuje data zprávy, zatímco tělo zahrnuje samotný obsah, který chce odesílatel sdělit příjemci.

E-mailová adresa

- **příjemce** - identifikátor uživatele, např. „sos.smail.person.1“ a tento identifikátor se nachází před zavináčem a může obsahovat povolené znaky podle standardu RFC 6532 [2].
- **zavináč(@)** - symbol oddělující uživatelskou část e-mailové adresy od domény,
- **doména** - název serveru, který zprávu zpracovává a doručuje ji konkrétnímu příjemci. Může být například ve tvaru „gmail.com“ nebo „seznam.cz“.

Hlavička

Hlavička e-mailové zprávy je její velmi důležitou součástí, protože obsahuje základní metadata, která umožňují správné směrování a identifikaci zprávy. Hlavička typicky obsahuje následující informace:

- **date (datum)** - datum a čas kdy zpráva opustila odesílatele,
- **from (od)** - pole obsahující e-mailovou adresu odesílatele,
- **to (komu)** - pole obsahující e-mailovou adresu přijemce,
- **subject (předmět)** - krátký popis obsahu zprávy,
- **CC (kopie)** - toto pole umožňuje odeslat kopii zprávy dalším příjemcům
- **BCC (skrytá kopie)** - pole pro skrytou kopii, které zajistí, že další příjemci dostanou kopii zprávy, aniž by jejich e-mailová adresa byla viditelná pro ostatní adresáty [3].

Tělo

Tělo e-mailu obsahuje samotný obsah zprávy, který odesílatel předává příjemci. Obsah těla může být od jednoduchého textu po více formátovaný obsah, který zahrnuje:

- **textová zpráva** - obsahuje samotný obsah zprávy,
- **přílohy** - přídavné dokumenty, obrázky nebo jiné soubory připojeny ke zprávě,
- **hypertextové odkazy** - odkazy, které mohou příjemce navigovat na určité webové stránky.

2.2 Formáty zpráv

Existují tři hlavní formáty, ve kterých jsou e-mailové zprávy odesílány, a každý z nich se hodí pro jiné potřeby. HTML formát poskytuje největší možnosti, pokud jde o stylování a přizpůsobení zprávy, což zahrnuje obrázky a odkazy. Rich Text formát umožňuje jednoduché úpravy, jako je tučný text nebo kurzíva, což je užitečné pro zvýraznění klíčových informací. Pokud je potřeba sdělit pouze čistý obsah bez formátování, používá se Textový formát, který je nejjednodušší a univerzální [4].

HTML formát

HTML (Hypertext Markup Language) je velmi běžně využívaný formát pro e-mailové zprávy, zejména pokud jde o profesionální e-mailové podpisy. Hypertext Markup Language, zkráceně HTML, je základní značkovací jazyk, který se používá pro kódování většiny webových stránek a slouží také k formátování e-mailových zpráv ve většině e-mailových klientů, jako jsou Outlook, Apple Mail či Gmail.

Je však důležité zmínit, že různí e-mailoví klienti mohou HTML interpretovat různými způsoby. Outlook využívá Word Rendering Engine, Apple Mail používá Webkit a Mailbird používá Chromium. To znamená, že e-mail vytvořený v HTML formátu může být na každé platformě vykreslen s mírnými odchylkami.

Výhody tohoto formátu:

- podpora hypertextových odkazů,
- možnost použití vložených nebo hostovaných obrázků,
- flexibilita při použití různých stylů a barev písma,
- podpora HTML tabulek,
- možnost využití většiny HTML atributů [4].

Rich text formát (RTF)

Je formát e-mailových zpráv, který je využíván tam, kde není nutná plná flexibilita HTML, ale přesto je potřeba určitá možnost úpravy textu. Tento formát umožňuje jednoduché textové formátování, jako je například použití tučného písma nebo kurzívy, a zároveň umožňuje přidání obrázků. Nicméně na rozdíl od HTML neumožňuje Rich Text použití HTML atributů.

Je zajímavé, že například Apple Mail, označují HTML formát jako Rich Text. Pokud tedy uživatel v Apple Mail vybere Rich Text, bude zpráva ve skutečnosti odeslána ve formátu HTML [5].

Výhody použití tohoto formátu:

- podpora textového formátování jako *kurzíva*, **tučné písmo**, <u>podtržení</u>,
- možnost přidání obrázků do textu,
- vkládání odrážek do e-mailu,
- některé možnosti zarovnání textu [4].

Textový formát

Tento formát umožňuje pouze použití prostého textu bez jakýchkoliv dalších formátovacích možností. V tomto formátu nejsou povoleny žádné úpravy, jako je tučné písmo, kurzíva nebo změna velikosti fontu. Také není možné přidávat obrázky. Z těchto důvodů není textový formát vhodný pro vytváření vizuálně atraktivních e-mailových podpisů.

Výhody použití tohoto formátu:

- podporován všemi e-mailovými klienty,
- nižší datová náročnost, což znamená menší velikost e-mailu,
- menší pravděpodobnost, že bude e-mail označen jako spam, jelikož zde nelze vkládat škodlivý HTML kód [4].

2.3 Přenos e-mailové zprávy

Přenos e-mailových zpráv probíhá prostřednictvím několika klíčových komponent, které se starají o správné doručení e-mailu od odesílatele k příjemci. Tento proces zahrnuje více etap, kde každá etapa má svou specifickou funkci a roli. Zpráva prochází přes několik uzlů, podobně jako tradiční poštovní dopis, než je doručena do schránky příjemce.

E-mailová zpráva se nejprve odešle pomocí aplikace nazývané Mail User Agent (MUA), která se připojuje k poštovnímu serveru prostřednictvím Mail Submission Agent (MSA). MSA přeposílá zprávu dále pomocí Mail Transfer Agent (MTA), který se stará o přenos e-mailu mezi různými servery. Pokud je příjemce na stejném serveru jako odesílatel, zpráva je doručena prostřednictvím Mail Delivery Agent (MDA) přímo do schránky. V opačném případě je zpráva přesměrována přes další servery. Po doručení je zpráva uložena v Message Store (MS), kde čeká, až ji příjemce prostřednictvím svého MUA otevře nebo jinak zpracuje. Průběh komunikace je znázorněn na obrázku níže [3, 6].

Poštovní klient (Mail User Agent - MUA)

Aplikace, která umožňuje koncovému uživateli komunikovat s e-mailovým systémem. Slouží k vytváření, odesílání a přijímání e-mailových zpráv. MUA je první bod kontaktu uživatele s e-mailovým systémem, přičemž umožňuje uživateli snadný přístup ke zprávám a jejich správu. Příklady MUA zahrnují Microsoft Outlook, Mozilla Thunderbird, Gmail a Email.cz [6].

Agent pro předání pošty k odeslání (Mail Submission Agent - MSA)

Komponenta, která přijímá zprávy od MUA a zajišťuje jejich předání do dalšího procesu, konkrétně k Mail Transfer Agentovi (MTA). MSA kontroluje, zda zprávy splňují požadované formáty, a předává je k dalšímu zpracování, což zajišťuje bezproblémový přenos zpráv směrem k příjemci [6].

Poštovní přenosový agent (Mail Transfer Agent - MTA)

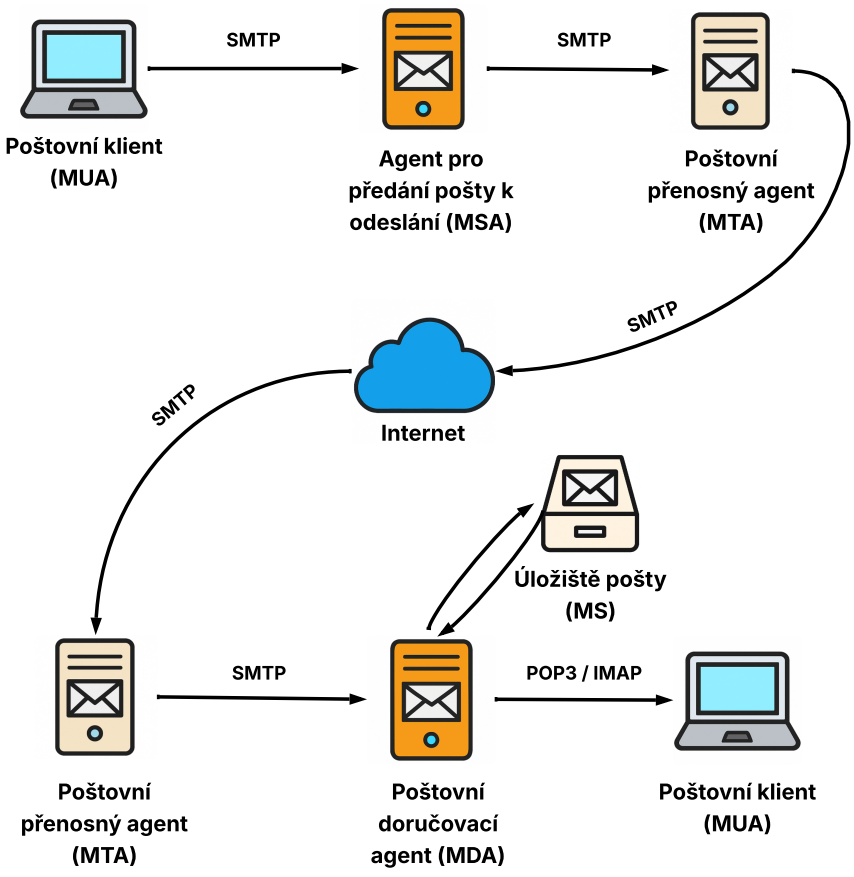
Klíčový prvkek přenosové části e-mailové infrastruktury. Přijímá zprávy od MUA nebo jiných MTA a zajišťuje jejich doručení do příslušného uzlu. Pokud schránka příjemce není na stejném serveru jako odesílatel, agent předává zprávu dalšímu serveru. V případě, že jsou obě schránky na stejném serveru, agent zajistí předání zprávy k Mail Delivery Agentovi (MDA). Typickými příklady MTA jsou Microsoft Exchange a sendmail [6].

Poštovní doručovací agent (Mail Delivery Agent - MDA)

Stará se o doručení zprávy do schránky příjemce. V případě, že jsou schránky odesílatele a příjemce na stejném serveru, MDA zprávu přímo uloží do schránky příjemce. Před doručením může MDA provádět různé operace, jako například filtrování zpráv podle nastavených pravidel, označení jako spam nebo třídění zpráv [6].

Úložiště pošty (Message Store - MS)

Úložiště, ve kterém jsou zprávy uloženy po jejich doručení. MS zajišťuje, že zprávy jsou k dispozici pro příjemce, dokud k nim nepřistoupí a nerozhodne se je přečíst, archivovat nebo smazat. MS také organizuje zprávy do složek, jako jsou doručená pošta, odeslané zprávy nebo koncepty, aby měl uživatel snadný přístup k různým typům zpráv [6].



Obr. 2.1: Znázornění průběhu e-mailové komunikace [6].

Systém doménových jmen (Domain Name System - DNS)

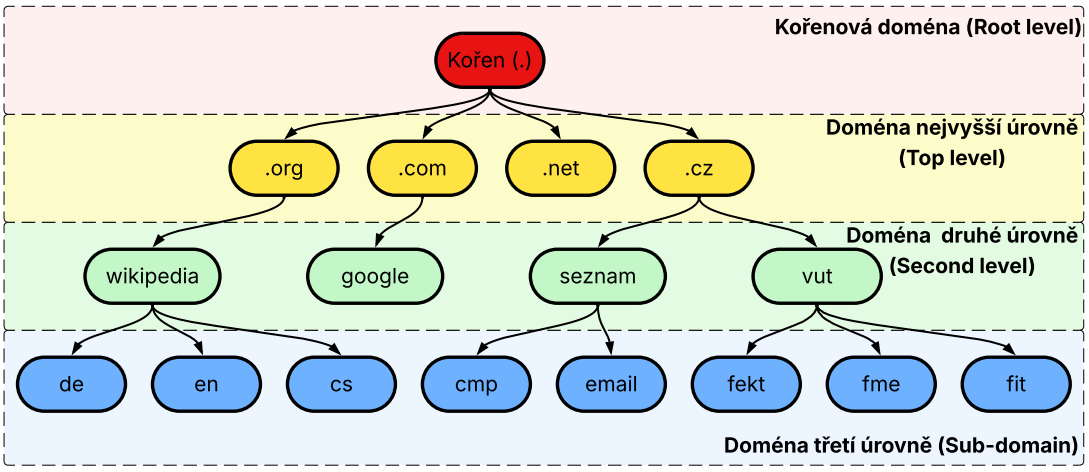
Hlavním úkolem DNS je zajistit spolehlivý a jednotný jmenný prostor, který může být použit k identifikaci různých zdrojů. Aby se předešlo problémům s nejednotným kódováním, DNS se vyhýbá zahrnování síťových identifikátorů nebo tras přímo do názvů.

Vzhledem k velikosti DNS databáze a častým aktualizacím bylo rozhodnuto, že systém musí být distribuovaný a využívat lokální cache, aby se zlepšila jeho výkonnost. Centralizace všech dat by byla nákladná a složitá, a proto se klade důraz na distribuci odpovědnosti. Rozhodování mezi náklady, rychlostí aktualizací a přesností dat závisí na konkrétním zdroji dat, což DNS činí velmi flexibilním.

DNS je navržen tak, aby sloužil širokému spektru aplikací. Jména mohou být použita k vyhledávání různých informací, jako jsou IP adresy nebo informace o poštovních schránkách. Každý záznam je označen jak třídou, tak typem, což umožňuje použití DNS v různých typech sítí. Transakce v DNS jsou nezávislé na konkrétním komunikačním systému. To znamená, že DNS může využívat jak jednoduché datagramy pro běžné dotazy, tak i spolehlivější virtuální okruhy pro složitější úkoly.

DNS je tvořen třemi hlavními částmi:

- **Prostor doménových jmen a zdrojové záznamy:** Jmenný prostor DNS má podobu stromové struktury, kde každý uzel a list obsahuje informace spojené s konkrétní doménou. Tyto informace mohou být různého druhu, jako například IP adresy hostitelů nebo informace o e-mailových schránkách. Dotazy v DNS směřují k získání specifických typů informací z určité části jmenného prostoru. Tento komponent je základem celé struktury DNS, protože každá informace v DNS je uložena právě zde.



Obr. 2.2: Struktura domén v systému DNS.

- **Name servery:** Name servery jsou servery, které uchovávají a poskytují informace o různých částech doménového stromu. Každý name server má úplné informace pouze o části jmenného prostoru, za kterou nese odpovědnost – říkáme, že je „autorita“ pro tuto část. Tyto autoritativní informace jsou organizovány do tzv. zón, které mohou být automaticky distribuovány mezi více name serverů, aby byla zajištěna spolehlivost a dostupnost informací.
- **Resolvery:** Resolvery jsou programy, které na základě požadavků od uživatelů získávají informace z name serverů. Resolver má přístup minimálně k jednomu name serveru, a pokud nemá informace přímo, může vyhledávat odpovědi i u dalších serverů. Z uživatelského pohledu je resolver jednoduchým nástrojem, který komunikuje přímo s name servery a poskytuje odpovědi na dotazy, aniž by uživatel musel mít hlubší znalost o fungování DNS [7].

2.4 Přehled e-mailových protokolů

V této podkapitole je podán komplexní přehled klíčových e-mailových protokolů, které se využívají v moderní elektronické komunikaci. Podkapitola je rozdělena do čtyř hlavních částí, přičemž každá se zaměřuje na konkrétní protokol. Nejprve je uveden přehled protokolu SMTP, který se používá pro přenos e-mailů mezi servery. Dále je popisován protokol IMAP, jenž umožňuje přístup ke zprávám přímo na serveru a synchronizuje jejich správu mezi různými zařízeními. Poté se zaměřuje na MIME, které rozšiřuje možnosti e-mailové komunikace o multimediální obsah, a nakonec je pojednáno o protokolu POP3, který slouží ke stahování e-mailů na zařízení uživatele.

2.4.1 Simple Mail Transfer Protocol (SMTP)

Základní komunikační protokol používaný k doručování e-mailových zpráv. SMTP poskytuje základní mechanismus pro přenos zpráv mezi klientem a serverem. Je navržen tak, aby umožňoval efektivní a spolehlivou výměnu e-mailů mezi uživateli.

SMTP používá pro komunikaci několik základních příkazů, mezi které patří například HELO, MAIL FROM, RCPT TO, DATA, QUIT a další. Tyto příkazy umožňují definovat informace o odesílateli, příjemci, obsahu zprávy a také zavřít spojení se serverem. Typická SMTP konverzace začíná příkazem HELO, kterým se odesílatel připojí k serveru a uvede doménu. Poté následuje MAIL FROM, kterým se identifikuje odesílatel, a RCPT TO, kterým se určí zamýšlený příjemce zprávy.

Kromě základního protokolu byly v průběhu let zavedeny také rozšíření SMTP, která umožňují rozšířenou funkcionalitu. Například příkaz EHLO (rozšířený hello) umožňuje ověřit, zda server podporuje SMTP rozšíření (známé také jako Extended

SMTP nebo ESMTP). Mezi nejběžnější rozšíření patří podpora pro autentizaci uživatelů (RFC 2554), oznamování stavu doručení (RFC 3461), nebo možnost deklarace velikosti zprávy (RFC 1870) [6].

2.4.2 Internet Message Access Protocol (IMAP)

Protokol navržený k tomu, aby poskytoval přístup k e-mailovým zprávám přímo na serveru, aniž by bylo nutné tyto zprávy stahovat do zařízení. Díky IMAP jsou e-maily uložené na jednom centrálním místě, což umožňuje uživatelům přistupovat k nim z různých zařízení. Všechny provedené změny, jako například čtení, mazání nebo třídění zpráv, se synchronizují napříč zařízeními, což udržuje e-maily ve stále aktuálním stavu.

IMAP4rev2 je aktuální verze protokolu, který umožňuje přístup k e-mailovým zprávám na serveru a jejich správu přímo ze zařízení uživatele. Umožňuje uživateli manipulovat se složkami na serveru podobně, jako kdyby pracoval s lokálními složkami na svém počítači. Tato funkcionalita je rozšířena o možnost synchronizace pro offline klienty, což umožňuje, aby změny provedené offline byly později synchronizovány s centrálním serverem.

IMAP4rev2 podporuje různé operace, jako je například vytváření, přejmenování a mazání složek, nebo trvalé mazání zpráv. Mimo jiné lze také selektivně přistupovat k určitým částem zpráv, což přispívá k větší flexibilitě práce s e-maily. Pro přístup k jednotlivým zprávám používá IMAP4rev2 buď sekvenční čísla zpráv, nebo unikátní identifikátory.Protokol IMAP4rev2 spoléhá na spolehlivý datový tok, jakým je například TCP, a komunikace standardně probíhá na portech 143 nebo 993.

Po navázání spojení mezi klientem a serverem se připojení protokolu IMAP4rev2 nachází v jednom ze čtyř možných stavů. Počáteční stav se identifikuje při uvítacím pozdravu serveru. Většina příkazů je platná pouze v určitých stavech, a pokus o vykonání příkazu v nevhodném stavu představuje protokolovou chybu, na kterou server odpoví chybovou zprávou BAD nebo NO (v závislosti na implementaci serveru) [6, 8].

Not Authenticated State (Neautentizovaný stav)

V tomto stavu musí klient poskytnout autentizační údaje, než mu bude dovoleno vykonat většinu příkazů. Tento stav je nastaven při každém novém spuštění spojení, pokud není spojení předem autentizováno. Klient v tomto stavu vykoná přihlášení pomocí LOGIN nebo AUTHENTICATE příkazu.

Authenticated State (Autentizovaný stav)

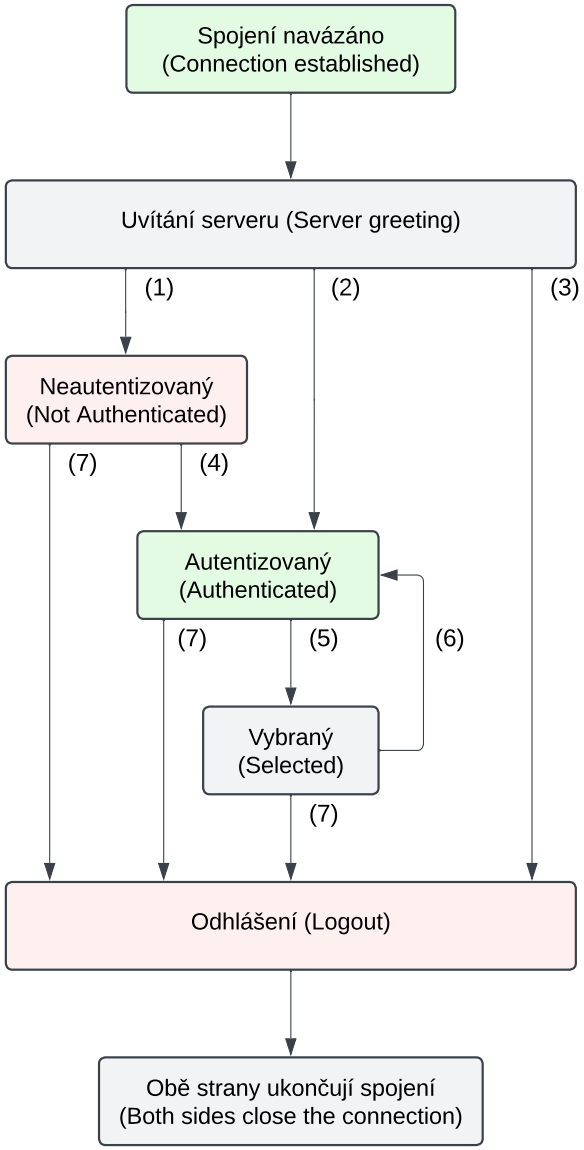
Po úspěšné autentizaci je klient v tomto stavu, ale musí vybrat konkrétní schránku, aby mohl vykonávat příkazy, které ovlivňují zprávy. Tento stav je aktivován buď na začátku spojení s předautentizovaným připojením, po úspěšném přihlášení, nebo po ukončení přístupu k dříve vybrané schránce. Příkazy jako SELECT nebo EXAMINE se používají k výběru schránky.

Selected State (Vybraný stav)

Ve vybraném stavu má klient přístup ke konkrétní schránce a může pracovat s jejími zprávami. Tento stav je dosažen po úspěšném výběru schránky pomocí příkazu SELECT nebo EXAMINE.

Logout State (Odhlášený stav)

Znamená, že se spojení mezi klientem a serverem ukončuje. Tento stav může být iniciován buď příkazem klienta LOGOUT, nebo jednostrannou akcí serveru. Při tomto odhlášení server zasílá odpověď BYE a potvrzuje ukončení spojení, na což klient následně odpoví ukončením svého spojení [8].



Obr. 2.3: Vývojový diagram znázorňující proces připojení a odpojení serveru [8].

Legenda pro diagram:

1. Připojení bez předchozí autentizace (OK greeting)
2. Předem autentizované připojení (PREAUTH greeting)
3. Spojení odmítnuto (BYE greeting)
4. Úspěšné přihlášení nebo autentizace (successful LOGIN or AUTHENTICATE)
5. Úspěšný výběr schránky (successful SELECT or EXAMINE command)
6. Uzavření nebo odznačení schránky (CLOSE or UNSELECT command)
7. Příkaz k odhlášení, vypnutí serveru nebo uzavření připojení (LOGOUT command, server shutdown, or connection closed) [8].

2.4.3 Multipurpose Internet Mail Extensions (MIME)

Technologie, která zajišťuje rozšíření možností e-mailové komunikace tak, aby zahrnovala nejen text, ale i různé multimediální prvky. MIME umožňuje odesílání a přijímání zpráv s obrázky, zvukovými záznamy či videi, což značně rozšiřuje původní schopnosti e-mailů.

Hlavní součásti MIME zahrnují:

- **MIME-Version:** Pole, které určuje verzi MIME standardu, například MIME-Version: 1.0.
- **Content-Type:** Toto pole uvádí typ a podtyp obsahu ve zprávě, např. Content-Type: text/html pro HTML obsah nebo Content-Type application/pdf pro PDF přílohy.
- **Content-Transfer-Encoding:** Udává způsob kódování dat, například Content-Transfer-Encoding: quoted-printable nebo base64, aby byla zajištěna kompatibilita během přenosu [9].

2.4.4 Post Office Protocol (POP3)

Představuje standardní způsob přístupu k e-mailovým zprávám uloženým na serveru. POP3 umožňuje uživatelům stahovat své e-maily na jejich osobní zařízení a je často využíván na menších zařízeních, kde není možné provozovat trvalé e-mailové služby. Typicky e-maily stáhnete a následně smažete z poštovního serveru, což je ideální pro použití s omezenými zdroji nebo připojením.

POP3 operace probíhá ve třech základních stavech:

- **Autorizace (Authorization):** Po navázání spojení se klient musí identifikovat pomocí uživatelského jména a hesla. Pokud je přihlášení úspěšné, server připraví schránku k přístupu.
- **Transakce (Transaction):** Během tohoto stavu může klient odesílat různé příkazy serveru. Mezi často používané příkazy patří LIST (zobrazení seznamu zpráv), RETR (stažení konkrétní zprávy), a DELE (označení zprávy k odstranění).
- **Aktualizace (Update):** Jakmile klient zadá příkaz QUIT, server přejde do stavu aktualizace, kde odstraní zprávy označené ke smazání [10].

3 Hrozby v e-mailové komunikaci

E-mail je dnes jedním z hlavních prostředků digitální komunikace, ovšem jeho popularita je často spojena s bezpečnostními zranitelnostmi. Tyto slabiny z něj dělají častý cíl pro kybernetické útoky, které se zaměřují na lidi, kteří nejsou dostatečně seznámeni s potenciálními riziky, nebo na ty, kteří jsou více zranitelní vůči útokům [3].

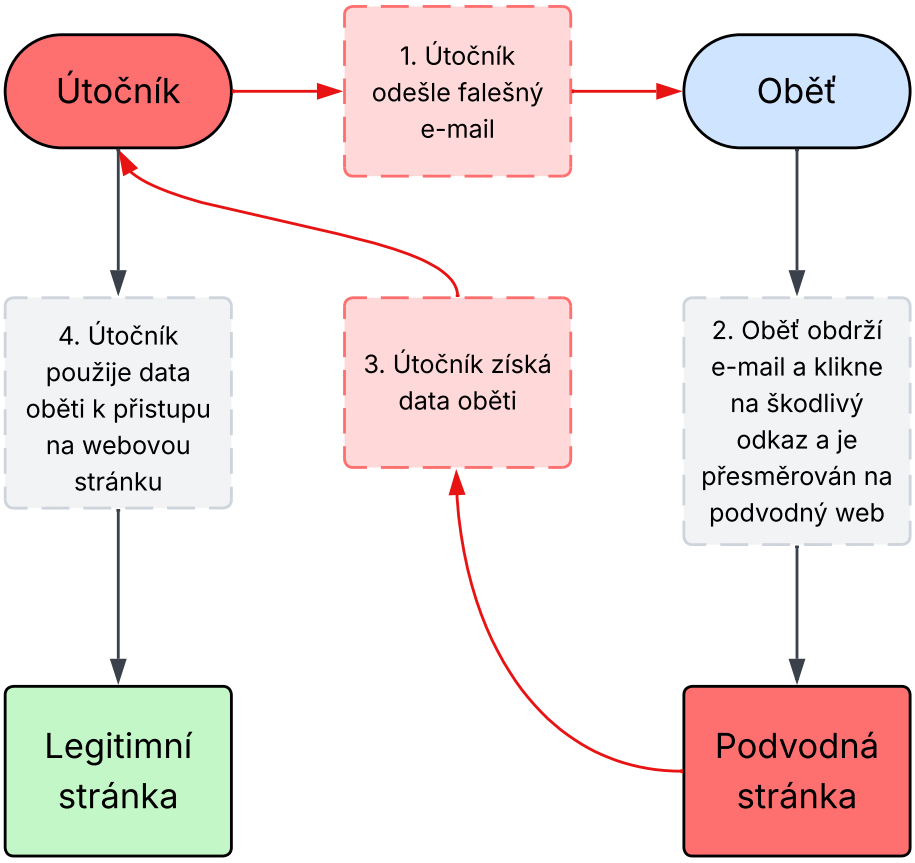
Bezpečnost e-mailové komunikace je klíčovým tématem, zejména v dnešní době, kdy e-mail zůstává jedním z nejčastěji zneužívaných kanálů pro kybernetické útoky. Tato kapitola se zaměřuje na různé bezpečnostní hrozby spojené s e-mailovou komunikací, techniky sociálního inženýrství a způsoby, jak těmto hrozbám čelit.

V oblasti kybernetické bezpečnosti představuje člověk často nejslabší článek celého obranného řetězce. Právě z tohoto důvodu útočníci hojně využívají tzv. techniky sociálního inženýrství – metody manipulace, jejichž cílem je oklamat uživatele a přimět ho, aby nevědomky předal citlivé informace nebo umožnil přístup do systému. Mezi používané přístupy patří například skrývání identity, využití důvěryhodného vzhledu, psychologický nátlak, nebo zneužití autority a postavení v organizaci. Tyto metody útočníkům umožňují proniknout do infrastruktury a dále se v ní šířit. Jednou z nejčastějších forem sociálního inženýrství je phishing, který je rozebrán v následující podkapitole [36].

3.1 Podvodné e-maily

Až 90% všech úspěšných kybernetických útoků začíná phishingovým e-mailem. Tento způsob útoku je pro útočníky stále velmi výhodný. Jejich metody se navíc neustále zdokonalují. Úplná eliminace phishingových pokusů zatím není reálná. Klíčem k prevenci úspěšných útoků je ale důkladné pochopení měnících se trendů v této oblasti a schopnost identifikovat rizika. Útočníci často zneužívají důvěryhodnost „známých“ odesílatelů, což činí phishing ještě nebezpečnějším [33, 34].

Phishing patří mezi nejrozšířenější a nejnebezpečnější formy útoků využívajících sociální inženýrství. Jeho cílem je získání citlivých údajů, jako jsou přihlašovací jména, hesla, čísla platebních karet nebo přístupové údaje k bankovním účtům. Útočník se přitom často vydává za důvěryhodnou autoritu, například banku, zaměstnavatele nebo státní instituci a formuluje žádost, která má vyvolat naléhavost a přimět oběť k reakci. Princip je podobný jako při rybaření: návnada v podobě důvěryhodně vypadající zprávy má „chytit“ nepozorného uživatele [35].



Obr. 3.1: Vizualizace průběhu phishingového útoku [35].

Podvod s platbou předem (advanced-fee scam)

Jedním z klasických příkladů phishingového útoku je tzv. podvod s předem zaplaceným poplatkem. V nejznámější variantě, známé jako „Nigerijský princ“, útočník slibuje vysokou finanční odměnu výměnou za malý počáteční poplatek. Po jeho zaplacení ale žádné peníze samozřejmě nikdy nedorazí. Tento typ útoku má kořeny už v 19. století, kdy byl znám jako „španělský vězeň“. V tehdejší verzi útočník, často vystupující jako bohatý cizinec nebo zástupce šlechty tvrdil, že pomáhá údajně uvězněné osobě s přístupem k velkému jmění, a žádal oběť o peníze na úplatky či soudní výlohy s příslibem pozdější odměny. Tento scénář pracuje s podobnými psychologickými mechanismy jako moderní phishing, vyvolává důvěru, apeluje na chamtivost a zároveň vytváří dojem naléhavosti. Přestože forma se změnila, princip zůstává stejný, což ukazuje, že lidé jsou stále náchylní k manipulaci, zejména pokud se útok maskuje za příležitost k rychlému zisku.

Útok přes hrozbu deaktivace účtu (account deactivation scam)

Útočník vyvolá paniku tvrzením, že pokud oběť ihned nezareaguje, bude její účet (například bankovní) zrušen. Oběť je pak navedena na podvodnou stránku, kde zadá své přihlašovací údaje, které útočník ihned získá. Někdy je po zadání údajů uživatel přesměrován na skutečné webové stránky dané instituce, aby nevzbudil podezření.

Falešná webová stránka (website forgery scam)

V tomto scénáři útočník vytvoří přesnou kopii důvěryhodného webu (např. banky) a přiměje oběť, aby zadala údaje právě zde. Podvodné stránky často přichází jako odkaz v e-mailu nebo jsou dokonce vyhledatelné přes internetové vyhledávače. Dnes jsou tyto falešné weby natolik sofistikované, že bývá obtížné je odlišit. Neobvyklý URL nebo chybějící HTTPS jsou jasným varováním.

Spear phishing

Zatímco běžný phishing cílí na široké publikum, spear phishing je cílený útok na konkrétní osobu nebo organizaci. Útočník si o oběti zjistí detailní informace, často i z veřejně dostupných zdrojů (např. sociální sítě), a na základě nich připraví velmi věrohodný útok. Tento typ phishingu je obzvláště nebezpečný a tvoří přes 90 % všech cílených útoků.

Clone phishing

Tento útok spočívá v úpravě legitimního e-mailu, který oběť již dříve obdržela. Útočník jej zkopíruje, přidá infikovanou přílohu nebo odkaz a e-mail zašle z podvržené adresy. Oběť tak získá pocit, že se jedná o pokračování předešlé komunikace, a s větší pravděpodobností otevře škodlivý obsah.

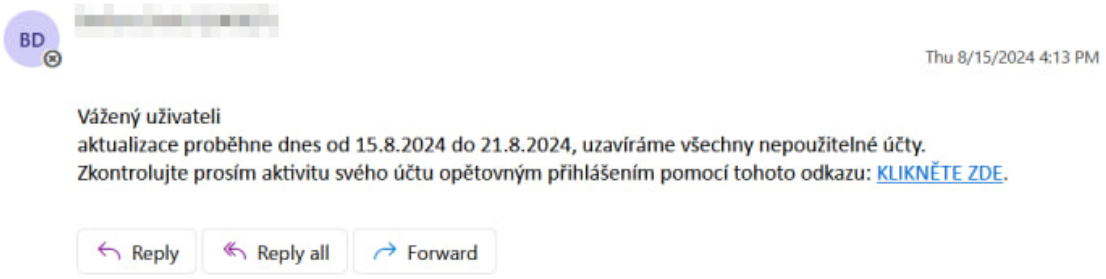
Whaling

Whaling, neboli „lov velryb“, je cílený phishing zaměřený na vrcholový management a vysoce postavené zaměstnance. Útočníci využívají dokumenty s naléhavým a důvěrným obsahem, například fiktivní soudní obsílky nebo požadavky na finanční převody. Častým trikem je e-mail, který údajně pochází od ředitele firmy a žádá například účetní o urychlený převod finančních prostředků [35].

3.2 Reálné jednoduché útoky

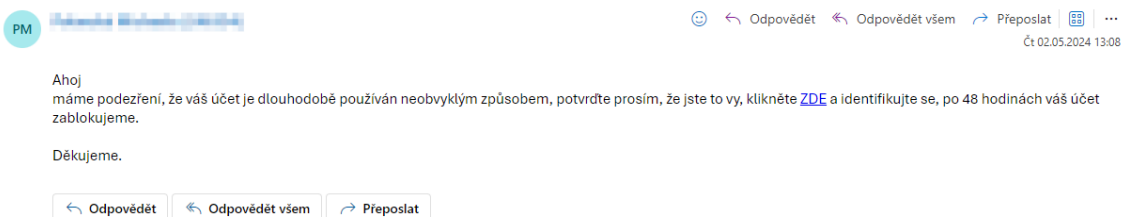
Tato podkapitola představuje sbírku autentických phishingových e-mailů, které byly v nedávné době zachyceny v rámci univerzitního prostředí Vysokého učení technického v Brně. Ukázky slouží jako názorná ilustrace běžně používaných technik sociálního inženýrství, které se mohou snadno stát nebezpečnými především pro méně technicky zdatné uživatele, jako jsou například senioři.

Útočníci v těchto zprávách obvykle vytvářejí falešný pocit naléhavosti, ohrožení nebo důležitosti. Cílem je přimět uživatele, aby klikl na podvodný odkaz a zadal své přihlašovací údaje, často bez podezření, že se jedná o podvrženou stránku. Právě senioři, kteří mohou mít menší zkušenosti s digitální bezpečností, bývají na podobné zprávy náchylnější, protože často spoléhají na důvěru k autoritám a berou text e-mailu jako závazný pokyn k akci.



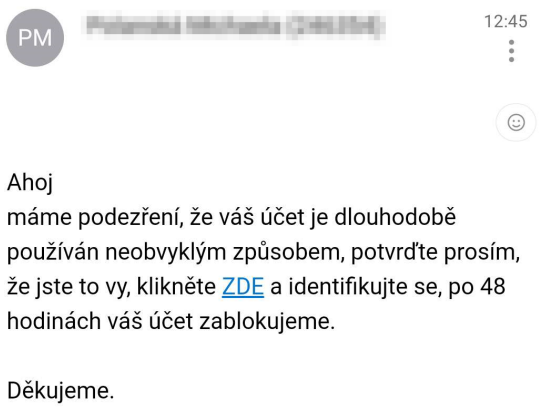
Obr. 3.2: Ukázka podvodného e-mailu s výzvou k reaktivaci účtu.

Tento e-mail simuluje výzvu k ověření aktivity účtu a vyhrožuje jeho uzavřením, pokud uživatel nereaguje. Útočník využívá časový nátlak a připojuje odkaz, kterým se snaží přimět oběť k zadání přihlašovacích údajů.



Obr. 3.3: Ukázka podvodného e-mailu s výzvou k ověření identity.

Vytváří dojem naléhavosti a strachu. Naznačuje, že byl účet dlouhodobě používán podezřelým způsobem a bude zablokován, pokud uživatel nepotvrdí svou identitu kliknutím na přiložený odkaz. Útočník se tak snaží získat přihlašovací údaje nebo další citlivé informace.



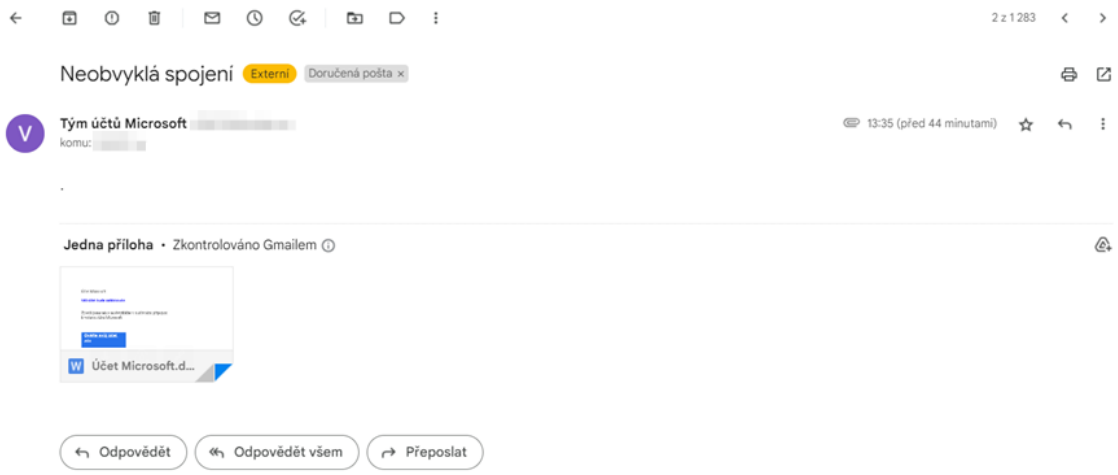
Obr. 3.4: Ukázka podvodného e-mailu vybízející k ověření aktivity účtu.

Navazuje na předchozí formu útoku, ale v mobilní podobě. Opět hrozí zablokováním účtu kvůli údajnému podezřelému používání a naléhá na uživatele, aby se co nejdříve identifikoval. Zpráva obsahuje odkaz „ZDE“, který je typickým phishingovým nástrojem pro odcizení přihlašovacích údajů.

3.3 Reálné sofistikované útoky

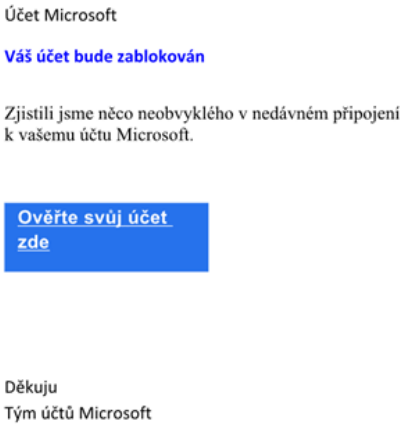
Na rozdíl od jednodušších podvodných zpráv, které spoléhaly především na časový nátlak a základní textové výzvy, představují následující příklady sofistikovanější typy útoků. Tyto útoky často využívají kombinaci formálně znějícího jazyka, graficky upravených příloh nebo přihlašovacích stránek, které věrně napodobují oficiální vzhled známých institucí (např. VUT).

Zatímco jednodušší phishingové útoky bývají často zaměřeny na širokou veřejnost nebo uživatele s omezenou orientací v digitálním prostředí, níže uvedené příklady cílí i na zaměstnance VUT, kteří by měli mít vyšší míru informovanosti. Právě proto útočníci volí sofistikovanější přístupy – například imitace univerzitního přihlašovacího portálu, e-maily předstírající komunikaci od IT správců nebo použití důvěryhodných domén v odkazu. Tyto zprávy často využívají autoritativní tón a grafiku připomínající oficiální komunikaci, což zvyšuje pravděpodobnost úspěchu útoku.



Obr. 3.5: Podvodný e-mail s přiloženým souborem, který má vyvolat důvěru a přimět uživatele k otevření nebezpečné přílohy.

Působí na první pohled věrohodně. Útočník se vydává za oficiální tým Microsoftu a upozorňuje na „neobvyklé přihlášení“. Přiložený dokument má uživatele přimět k akci bez zbytečného zdržování. Použití přílohy je záměrné a očekává se, že příjemce bude mít menší podezření než při proklikávání podezřelých odkazů. Uživatelé často věří formálně znějícím oznámením, zvláště pokud je doprovází naléhavý tón.

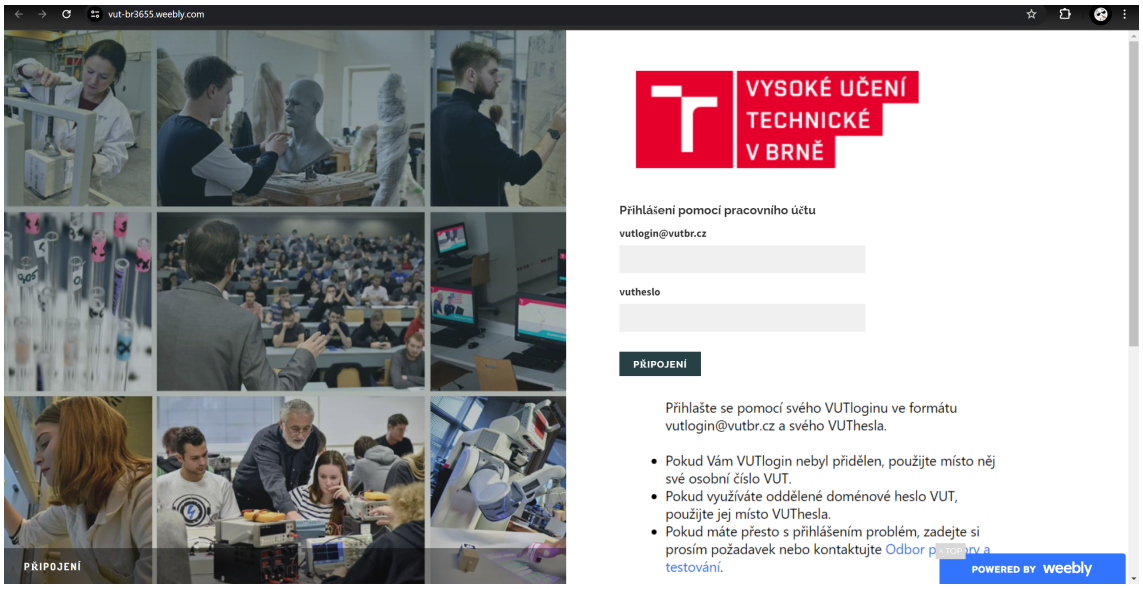


Obr. 3.6: Obsah přílohy předstírá zprávu od Microsoftu a využívá výhružného tónu k přinucení oběti k ověření účtu.

Obsah přílohy navazuje na text e-mailu a dále stupňuje tlak na příjemce. Útočník zde pracuje s psychologickým nátlakem („Váš účet bude zablokován“) a přidává výrazné vizuální prvky, jako je modré tlačítko s výzvou k ověření účtu. Tento styl komunikace cílí na vyvolání okamžité reakce oběti bez hlubšího přemýšlení. Takové přílohy často obsahují škodlivý skript nebo přesměrování na podvodnou stránku.

Obr. 3.7: Podvodný e-mail vydávající se za správce IT služby FSI VUT [38].

Zpráva se snaží působit důvěryhodně pomocí odkazu na web obsahující název univerzity a předstírá, že byl odeslán z oficiální adresy správce IT. Obsah zprávy varuje před „zastaralou schránkou“ a vyhrožuje mazáním účtů, čímž vytváří nátlak na uživatele. Odkaz odkazuje na stránku hostovanou mimo doménu školy (na službě Weebly), což je jednoznačný varovný signál. Cílem útočníka je přimět oběť k zadání citlivých údajů.



Obr. 3.8: Další varianta podvodné přihlašovací stránky [38].

Tato stránka se snaží imitovat přihlašovací formulář univerzity s cílem získat přihlašovací údaje uživatele. Nicméně webová adresa (doména třetí strany weebly.com) odhaluje, že nejde o oficiální web VUT. Podvržený vzhled a doména jsou typickými znaky phishingového útoku. Cílem je opět vylákat od uživatele přihlašovací údaje.

4 Použité technologie a nástroje

V této kapitole jsou představeny technologie a nástroje, které byly použity při vývoji aplikace SMAIL (E-mailový klient pro seniory). Klíčovým prvkem celého vývoje byl programovací jazyk **Python**, který byl zvolen pro svou čitelnost, jednoduchost a širokou dostupnost knihoven. Python je interpretovaný a univerzální jazyk zaměřený na srozumitelný zápis kódu, což umožňuje vývojářům rychlou a efektivní práci. Díky podpoře více programovacích paradigmat (objektově orientované, funkcionální i procedurální programování) a rozsáhlé standardní knihovně se jedná o jeden z nejvhodnějších jazyků pro vývoj desktopových aplikací, jako je SMAIL.

Python se vyznačuje přenositelností mezi různými operačními systémy a podporuje integraci s externími knihovnami v jazycích C nebo C++. Velkou výhodou je také aktivní komunita a rozsáhlý ekosystém nástrojů dostupných přes PyPI, které výrazně urychlují vývoj a testování softwaru.

V rámci této kapitoly jsou dále podrobně popsány knihovny a moduly, které byly využity k implementaci konkrétních funkcionalit aplikace. Knihovny jako `os`, `sys`, `datetime`, `ssl` a další byly použity pro práci se souborovým systémem, manipulaci s daty, zpracování času či šifrovanou komunikaci. Knihovna **PyQt5** byla klíčová pro tvorbu grafického uživatelského rozhraní a zajištění přehledného zobrazení jednotlivých částí aplikace. Pro práci s obrázky byl použit modul **Pillow**, zatímco knihovna **chardet** sloužila k detekci znakové sady při zpracování e-mailových zpráv.

V textu jsou popsány konkrétní funkce každé knihovny, důvody jejich nasazení a způsob, jakým přispěly k vytvoření stabilní, bezpečné a uživatelsky přívětivé aplikace.

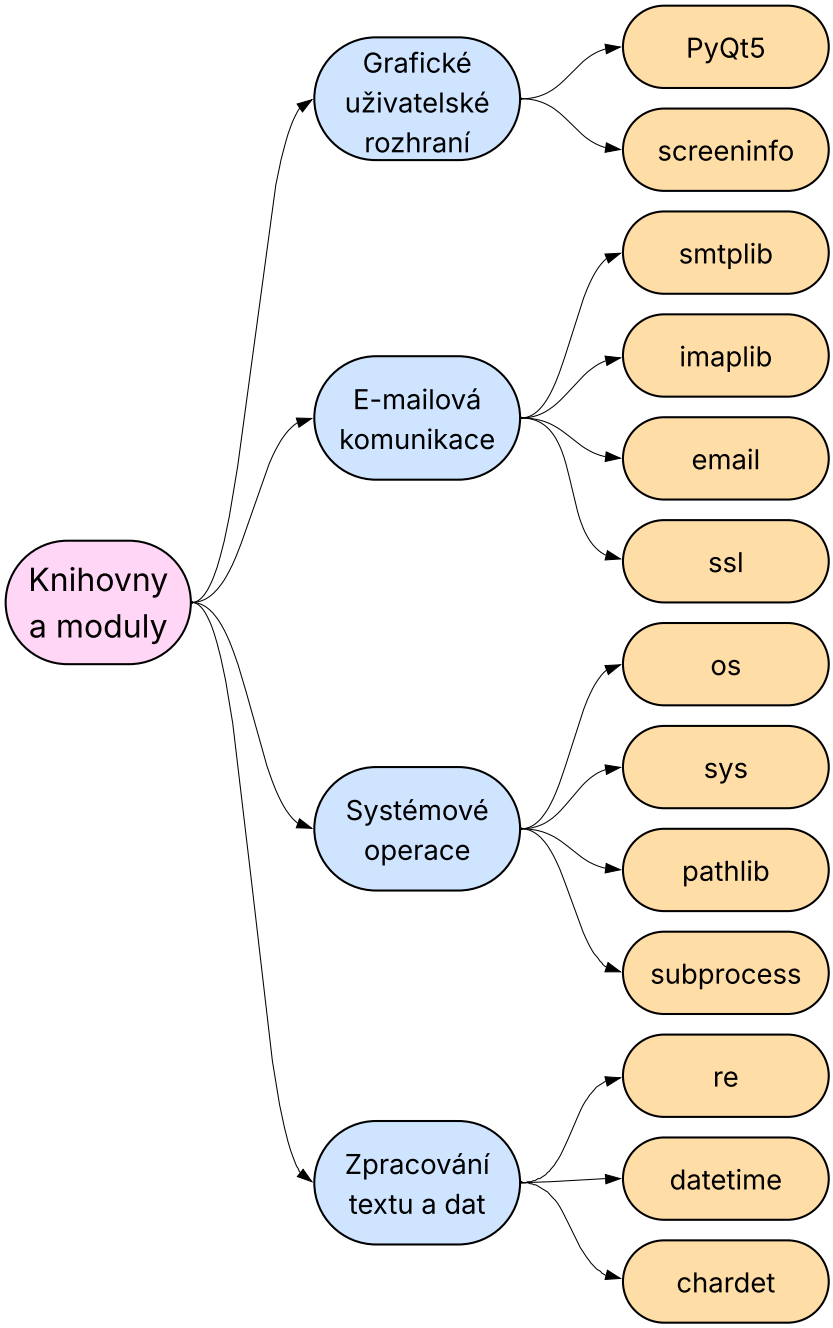
Dále je popsán také externí nástroj **Poetry**, který byl využit pro správu závislostí, sestavování a balíčkování projektu. Tento nástroj poskytuje opakovatelné a konzistentní vývojové prostředí, zjednodušuje instalaci balíčků a jejich aktualizaci a umožňuje jednodušší sdílení projektu s dalšími vývojáři. Využití Poetry přispělo ke zvýšení udržitelnosti a přehlednosti celého projektu.

4.1 Moduly a knihovny

V této sekci jsou popsány knihovny a moduly, které byly využity pro implementaci aplikace SMAIL. Knihovny Pythonu byly použity k řešení různorodých úkolů – od práce se souborovým systémem, manipulace s daty, zpracování času, až po implementaci síťových protokolů a bezpečnostních prvků.

Každý z modulů a knihoven plní specifickou roli při tvorbě aplikace. Knihovna **os** poskytuje prostředky pro přístup k funkcím závislým na operačním systému, zatímco **sys** umožňuje interakci s interpreterem Pythonu a přístup k argumentům

příkazové řádky. Modul **datetime** byl použit pro práci s datem a časem, a modul **ssl** pro implementaci šifrované komunikace. Všechny knihovny byly vybrány tak, aby efektivně podporovaly vývoj aplikace s důrazem na bezpečnost, stabilitu a uživatelskou přívětivost.



Obr. 4.1: Přehled knihoven a modulů použitých v aplikaci SMAIL, rozdělených podle účelu.

Knihovna **PyQt5** byla použita pro tvorbu grafického uživatelského rozhraní, které je přehledné a snadno použitelné, přičemž knihovna **Pillow** zajišťuje práci s obrázky. Dále byl použit modul **subprocess** pro spouštění nových procesů, a modul **threading** pro podporu paralelního zpracování úloh. Modul **re** umožňuje práci s regulárními výrazy, což bylo užitečné při zpracování a validaci textových vstupů.

V této sekci je tedy představeno, jak každá z uvedených knihoven přispěla k vývoji aplikace, jaké hlavní funkcionality byly implementovány pomocí těchto nástrojů a jakým způsobem bylo zajištěno efektivní a bezpečné řešení každodenních úloh aplikace.

Modul pro práci s funkcemi operačního systému - os

Poskytuje přenosný způsob použití funkcí závislých na operačním systému. Je navržena tak, aby zajišťovala konzistentní rozhraní pro všechny vestavěné funkce operačního systému, které jsou v Pythonu dostupné. To znamená, že bez ohledu na konkrétní operační systém poskytuje stejnou funkcionalitu stejným způsobem. Například funkce os.stat(path) vrací informace o souboru nebo adresáři ve stejném formátu napříč různými platformami.

Tato knihovna umožňuje práci se souborovým systémem a cestami, jako je získání seznamu souborů v adresáři, práce s proměnnými prostředí, nebo kontrola přístupových práv k souborům. Funkce pro čtení nebo zápis do souborů jsou dostupné přes funkci open(). Pro práci s cestami lze využít modul os.path. Pokud potřebujeme vytvářet dočasné soubory nebo adresáře, je vhodné využít modul tempfile. Pro pokročilejší manipulaci se soubory a adresáři lze využít modul shutil.

Je také třeba poznamenat, že knihovna os nabízí rozšíření specifická pro jednotlivé operační systémy, což může omezit přenositelnost kódu mezi různými platformami. Například na některých platformách, jako je WebAssembly, Android nebo iOS, nejsou dostupné některé funkce související s procesy, jako jsou fork() nebo execve(), nebo s prostředky jako nice(). V těchto případech se některé funkce chovají odlišně nebo jsou nahrazeny emulacemi či stuby. Funkce v knihovně os přijímají jak bajtové, tak řetězcové objekty, což umožňuje flexibilnější práci se soubory a cestami [12].

Modul pro interakci s Python interpretem a proměnnými příkazové řádky - sys

Poskytuje přístup k různým proměnným a funkcím, které jsou používány nebo spravovány Python interpreterem, a které umožňují úzkou interakci s ním. Tento modul je vždy k dispozici a umožňuje například manipulovat s cestou, kde Python hledá moduly, nebo pracovat s argumenty příkazového řádku.

Jednou z klíčových funkcí modulu sys je proměnná sys.path, která obsahuje seznam cest, kde interpreter hledá moduly při importování. Tento seznam je inicializován z proměnné prostředí PYTHONPATH a výchozí cesty závislé na instalaci. Cesty v sys.path lze modifikovat i v průběhu programu, což umožňuje flexibilitu při dynamickém načítání modulů z různých míst.

Modul sys také poskytuje možnost ovlivnit chování Pythonu při spuštění skriptu. Například při spuštění skriptu pomocí python script.py se k proměnné sys.path přidá cesta k adresáři skriptu. Podobně při použití python -m module se přidá aktuální pracovní adresář. Aby se zabránilo přidání potenciálně nebezpečných cest, lze použít příkazovou volbu -P nebo nastavit proměnnou prostředí PYTHONSAFEPATH.

Kromě sys.path obsahuje modul sys i další proměnné a funkce pro práci s argumenty příkazového řádku (sys.argv), spravování verze Pythonu (sys.version), nebo pro ovládání výstupů (sys.stdout a sys.stderr) [13].

Modul pro práci s datem a časem - datetime

Poskytuje třídy pro práci s daty a časy, včetně jejich vytváření, formátování, aritmetiky a extrakce jednotlivých atributů. Tato knihovna je užitečná například při práci s časovými razítky, kdy potřebujeme získat nebo zpracovat konkrétní datum nebo čas.

Modul datetime umožňuje vytvářet tzv. „naivní“ a „uvědomělé“ objekty. Naivní objekty neobsahují informace o časovém pásmu a jejich výklad je zcela na konkrétním programu. Naopak uvědomělé objekty zahrnují informace o časovém pásmu a letním čase, díky čemuž dokážou přesně určit svůj časový vztah k ostatním objektům. Pro aplikace vyžadující uvědomělé objekty je možné nastavit atribut tzinfo — ten zachycuje informace o časovém posunu od UTC, názvu časového pásma a případném letním čase.

Třída timezone je jedinou konkrétní implementací tzinfo poskytovanou modulem datetime. Pomocí této třídy lze reprezentovat jednoduchá časová pásma s pevnými posuny od UTC, například samotné UTC nebo časová pásma v Severní Americe jako EST a EDT.

Modul datetime rovněž poskytuje několik konstant, jako například MINYEAR a MAXYEAR, které určují minimální a maximální hodnoty let pro objekty date a datetime (od roku 1 až po rok 9999). Další důležitou konstantou je UTC, která představuje časové pásmo UTC [14].

Modul pro implementaci šifrování a bezpečné komunikace - ssl

Poskytuje přístup k šifrování a autentizaci pomocí protokolů Transport Layer Security (TLS) a Secure Sockets Layer (SSL) pro síťové sokety, a to jak na straně klienta,

tak serveru. Pro šifrování využívá knihovnu OpenSSL, což znamená, že je dostupná na většině moderních systémů, jako jsou Unix, Windows, macOS, a další platformy, kde je OpenSSL nainstalován.

Modul ssl zahrnuje třídu ssl.SSLSocket, která rozšiřuje typ socket.socket a poskytuje zašifrované sokety, jejichž data jsou šifrována a dešifrována pomocí SSL. Umožňuje také dodatečné metody jako getpeercert(), která získává certifikát druhé strany připojení, nebo cipher(), která vrací informace o šifře používané pro zabezpečené spojení.

Pro pokročilejší aplikace knihovna obsahuje třídu ssl.SSLContext, která slouží k řízení nastavení šifrování a správě certifikátů. Konkrétní SSL kontext lze vytvořit pomocí metody `create_default_context()`, která poskytuje bezpečné výchozí nastavení pro klientské i serverové použití. Toto nastavení zahrnuje omezení na bezpečnější šifry a vylučuje zastaralé verze SSL (SSLv2 a SSLv3), stejně jako slabší šifrovací sady.

Knihovna ssl je klíčová při implementaci zabezpečené komunikace mezi klienty a servery, čímž zajišťuje důvěrnost, integritu dat a autentizaci [15].

Modul pro spouštění nových procesů a interakci s nimi - subprocess

Umožňuje vytvářet nové procesy, připojit se k jejich vstupním, výstupním a chybovým proudům a získávat jejich návratové kódy. Tento modul nahrazuje několik starších modulů a funkcí, jako je os.system a os.spawn*. Pro základní spouštění procesů se doporučuje používat funkci run(), která zajišťuje jednoduché spuštění a čekání na dokončení procesu.

Pro pokročilejší scénáře použití je možné přímo pracovat s rozhraním Popen, které poskytuje větší kontrolu nad vytvářením a komunikací s podřízenými procesy. Pomocí subprocess.run() můžeme zadat různé argumenty jako vstup, výstup, časový limit, nebo zda má být zachycena chyba. Když je nastaven argument capture_output=True, zachycují se výstupy standardního výstupu (stdout) a chybového výstupu (stderr).

Modul subprocess nabízí také možnost specifikovat proměnné prostředí pro nový proces, což může být užitečné v případech, kdy potřebujeme odlišné proměnné od těch, které jsou dostupné aktuálnímu procesu. Modul je tak ideální pro práci s externími programy a skripty, protože poskytuje vysokou míru flexibility při jejich spouštění a správě [16].

Modul pro práci s vlákny - threading

Používán k poskytování vyšší úrovně rozhraní pro práci s vlákny nad nízkoúrovňovým modulem `_thread`. V implementaci CPython je kvůli Globálnímu zámku in-

terpretátoru (Global Interpreter Lock, GIL) umožněno, aby Python kód vykonávalo pouze jedno vlákno najednou. Přestože některé knihovny zaměřené na výkon mohou tuto limitaci obejít, doporučuje se použít místo toho modul multiprocessing nebo concurrent.futures.ProcessPoolExecutor, pokud se má zajistit co nejefektivnější využití výpočetních zdrojů vícejádrových strojů. Modul threading je však vhodný pro případy, kdy je potřeba provádět více úloh zaměřených na I/O (vstupně-výstupní operace) paralelně.

Knihovna threading umožňuje vytvářet a spravovat více vláken v rámci aplikace. Nabízí několik užitečných funkcí, například:

- `threading.active_count()`: Vrací počet momentálně aktivních vláken.
- `threading.current_thread()`: Vrací objekt aktuálního vlákna, které právě provádí kód volající funkce.
- `threading.excepthook(args)`: Umožňuje manipulovat s výjimkami, které nebyly zachyceny během vykonávání vlákna.

I když modul threading neposkytuje výhody pro CPU-bound (úlohy zaměřené na intenzivní využití procesoru) úlohy kvůli GIL, je ideální pro I/O-bound úlohy, jako je čtení nebo zápis do souboru, komunikace po síti a podobně. Threading je tak vhodný nástroj, pokud chceme zajistit responzivitu aplikace během vykonávání dlouhých I/O operací [17].

Modul pro práci s regulárními výrazy - re

Je využíván k provádění operací pro shodu s regulárními výrazy, podobně jako je tomu u jazyka Perl. Vzory i řetězce, které mají být prohledávány, mohou být ve formě Unicode řetězců (str) nebo 8bitových řetězců (bytes). Kombinování Unicode řetězců a 8bitových řetězců není povoleno, tzn. nelze například použít vzor zapsaný jako bytes pro vyhledávání v Unicode řetězci.

V regulárních výrazech se zpětné lomítko ('\') používá k vyznačení speciálních forem nebo k úniku speciálního významu znaků. Toto se dostává do kolize s Pythonovým používáním stejného znaku ve stringových literálech, takže pro použití doslovného zpětného lomítka je nutné použít čtyři zpětná lomítka ('\\\\'). Pro snazší práci s regulárními výrazy se často doporučuje používat tzv. surové (raw) řetězce, které se označují předponou 'r', což eliminuje potřebu používat více zpětných lomítek.

Většina operací s regulárními výrazy je poskytována jak ve formě funkcí na úrovni modulu, tak jako metody na kompilovaných regulárních výrazech. Modulové funkce slouží jako zkratky, které nevyžadují nejprve kompilaci objektu regex, ale tyto funkce postrádají některé možnosti jemného doladění parametrů [18].

Knihovna pro práci s e-mailovými zprávami - emaiL

Knihovna email je určena k práci s e-mailovými zprávami, jako je jejich tvorba, čtení a manipulace. Tato knihovna slouží jako nástroj pro správu a práci s e-mailovými zprávami.

Hlavní struktura knihovny email se skládá ze tří komponent:

- **Objektový model zpráv** - představuje e-mailové zprávy jako stromovou strukturu objektů. Aplikace může využívat rozhraní této komponenty k tvorbě, dotazování a manipulaci s e-maily, což zahrnuje i přidávání či odstraňování dílčích částí e-mailu.
- **Parser a generátor** - parser slouží k převodu serializované verze e-mailu (streamu bytů) na strom objektů typu EmailMessage. Generátor naopak převede objekt EmailMessage zpět na serializovaný proud bajtů.
- **Policy (politika)** - kontroluje chování zpráv, parserů a generátorů. Tato politika se definuje při vytvoření e-mailové zprávy nebo může být změněna při serializaci zprávy.

Knihovna se snaží zjednodušit práci s e-maily tím, že skrývá složité detaily různých RFC standardů, jako jsou RFC 5322, RFC 6532 nebo MIME související RFC. Tímto způsobem se může pracovat s e-maily jako s hierarchií textových a binárních příloh, aniž by se zabývalo konkrétním způsobem, jak jsou tyto zprávy serializovány [19, 20].

Knihovna pro komunikaci s IMAP servery - imaplib

Poskytuje tři třídy pro komunikaci s IMAP servery: IMAP4, IMAP4_SSL a IMAP4-_stream. Tyto třídy umožňují připojení k IMAP4 serveru a implementují většinu funkcionalit IMAP4rev1 klientského protokolu, definovaného v RFC 2060. Knihovna je zpětně kompatibilní se servery podporujícími IMAP4 (RFC 1730), ale nepodporuje příkaz STATUS, který byl součástí starší specifikace.

- **IMAP4:** Tato třída implementuje základní IMAP4 protokol. Umožňuje připojení k serveru IMAP na specifikovaném hostiteli a portu. Standardně se používá port 143 pro IMAP4. Třída podporuje použití s with statementem, kdy je příkaz LOGOUT automaticky odeslán po opuštění bloku.
- `IMAP4-SSL`: Tato třída je odvozena od IMAP4 a poskytuje zabezpečené připojení pomocí SSL. Umožňuje vytvořit šifrované připojení na portu 993, který je standardně používán pro IMAP4-over-SSL. Volitelný argument ssl_context umožňuje specifikovat SSL konfiguraci, včetně certifikátů a soukromých klíčů, což zajišťuje bezpečnější připojení.

Knihovna imaplib definuje také několik výjimek:

- **IMAP4.error:** Vzniká při chybách během IMAP komunikace.

- **IMAP4.abort:** Tato výjimka signalizuje chybu na straně IMAP serveru. Obvykle lze chybu vyřešit opětovným uzavřením instance a jejím znovuotevřením.
- **IMAP4.readonly:** Tato výjimka nastane, když dojde ke změně stavu poštovní schránky na serveru (například pokud jiný klient získá oprávnění k zápisu) [21].

Knihovna pro odesílání e-mailů pomocí SMTP protokolu - smtplib

Poskytuje rozhraní pro klientskou SMTP relaci, které umožňuje odesílat e-maily na libovolný stroj připojený k internetu, který má spuštěn SMTP nebo ESMTP server. Používá se pro implementaci protokolu SMTP (Simple Mail Transfer Protocol) a je užitečný pro automatické odesílání e-mailů z aplikací.

Objekt SMTP v tomto modulu zabalí připojení k SMTP serveru a poskytuje metody podporující plnou sadu operací SMTP a ESMTP, jako jsou inicializace připojení, odesílání e-mailů nebo ukončení relace. Modul také zahrnuje třídu SMTP_SSL pro šifrované připojení přes SSL od začátku relace. Tento modul se používá pro jednoduchou komunikaci mezi klientem a SMTP serverem, což zahrnuje například metody sendmail() pro odesílání e-mailů a quit() pro uzavření relace [22].

Modul pro spuštění Python modulů bez importu - runpy

Umožňuje spouštět Python moduly bez toho, aby byly nejprve importovány. Používá se hlavně pro implementaci přepínače příkazového řádku -m, který umožňuje najít a spustit skripty pomocí názvového prostoru Python modulů namísto přímé práce se souborovým systémem.

Funkce poskytované modulem runpy umožňují lokalizovat a vykonat kód zadaného modulu nebo souboru. Kód je vykonán v samostatném modulovém jmenném prostoru, což zajišťuje jeho nezávislost na aktuálním prostředí. Ovšem kód není vykonán v sandboxu, což znamená, že všechny vedlejší efekty, například cachování importů, zůstanou po provedení funkce v aktuálním procesu. Pro použití, kde by tato omezení nebyla vhodná, může být lepší volbou modul importlib [23].

Knihovna pro automatickou detekci kódování textu - chardet

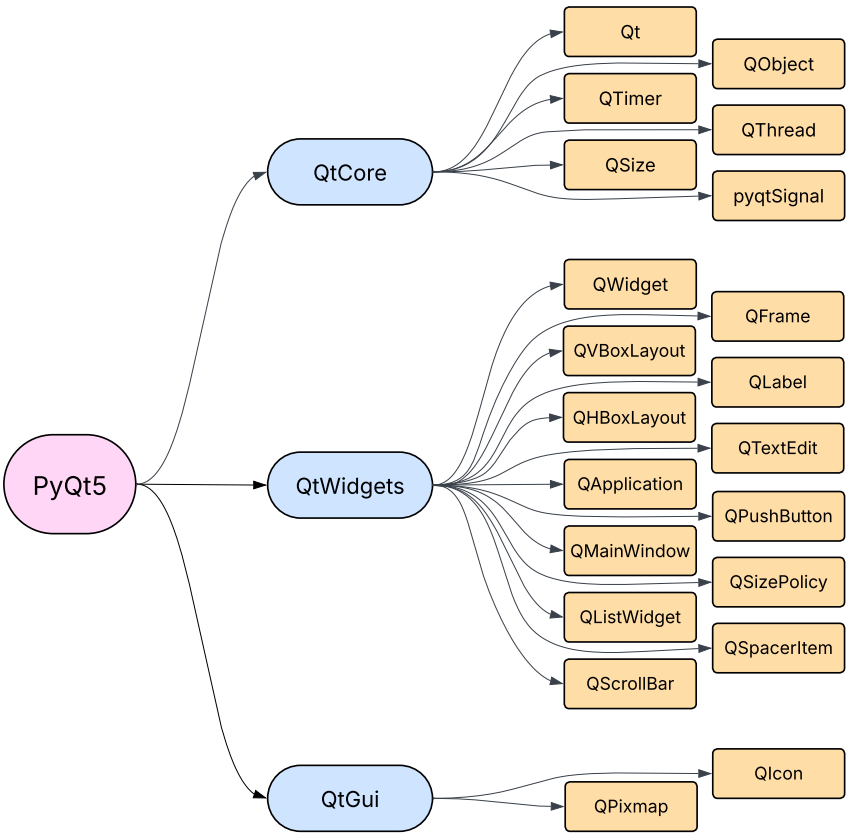
Používá se k automatické detekci kódování textu. Každý text, který vidíme na obrazovce, je uložen v určitém kódování znaků. Kódování znaků poskytuje mapování mezi znaky, které vidíme, a tím, co počítač skutečně ukládá v paměti nebo na disk. Automatická detekce kódování znamená převzít sekvenci bajtů v neznámém kódování a pokusit se určit kódování, aby bylo možné text správně zobrazit nebo zpracovat.

I když automatická detekce kódování není vždy stoprocentně spolehlivá a bývá pomalejší než používání standardních metod specifikace kódování, může být užitečná v případech, kdy kódování textu není explicitně určeno nebo je nesprávné [24].

Knihovna pro tvorbu grafického uživatelského rozhraní - PyQt5

Knihovna `PyQt5` je sada Pythonových vazeb pro Qt aplikační framework, který umožňuje vytvářet platformově nezávislé grafické uživatelské rozhraní (GUI) pomocí Pythonu. Tento framework zahrnuje rozhraní pro uživatelská rozhraní, práci s databázemi, podporu SVG, OpenGL, XML a mnoho dalších funkcionalit, včetně komunikace (NFC a Bluetooth), prohlížení webu, animace a 3D vizualizace. PyQt5 implementuje přes 1000 těchto tříd jako Python moduly.

PyQt5 podporuje Windows, Linux, UNIX, Android, macOS a iOS platformy a umožňuje vývoj aplikací rychleji než v C++, aniž by došlo ke snížení výkonu. Používání PyQt5 je výhodné zejména pro vývojáře, kteří preferují Python kvůli jeho jednoduchosti, ale chtějí využít sílu Qt napsaného v C++ [25].



Obr. 4.2: Strukturovaný přehled importovaných tříd a funkcí z PyQt5.

Knihovna pro práci s obrázky - pillow

Přidává možnosti zpracování obrázků do Pythonu. Poskytuje širokou podporu různých formátů souborů, efektivní vnitřní reprezentaci a silné nástroje pro úpravu obrázků. Pillow je jádrová knihovna navržená pro rychlý přístup k datům uloženým v základních formátech pixelů, což ji činí ideální pro tvorbu nástrojů pro obecné zpracování obrázků.

Knihovna je vhodná pro aplikace zaměřené na archivaci a dávkové zpracování obrázků, umožňuje tvorbu miniatur, převod mezi formáty a tisk obrázků. Obsahuje také funkce pro zobrazení obrázků a základní zpracování obrázků, jako jsou operace na pixelech, filtrace s využitím konvolučních jader a převody mezi barevnými prostory. Podporuje i změny velikosti obrázků, rotaci a afinní transformace [26].

Knihovna pro získávání informací o obrazovkách - screeninfo

Umožňuje získávat informace o fyzických obrazovkách, jako je jejich umístění a velikost. Podporuje různé prostředí včetně MS Windows, GNU/Linux (prostřednictvím X11 nebo experimentálně DRM) a macOS (přes AppKit). Tato knihovna může být užitečná například pro aplikace, které vyžadují informace o rozlišení nebo fyzickém rozměru obrazovek, což je důležité při vytváření uživatelských rozhraní nebo grafických výstupů [27].

Knihovna pro (de)serializaci datových tříd - dataclass_wizard

Poskytuje flexibilní možnosti pro (de)serializaci datových tříd, což zahrnuje převod do/z formátů JSON, TOML, YAML nebo běžných Python slovníků (dict). Nabízí také generování datových tříd z JSON vstupu, což usnadňuje práci se strukturou dat získanou z externích zdrojů.

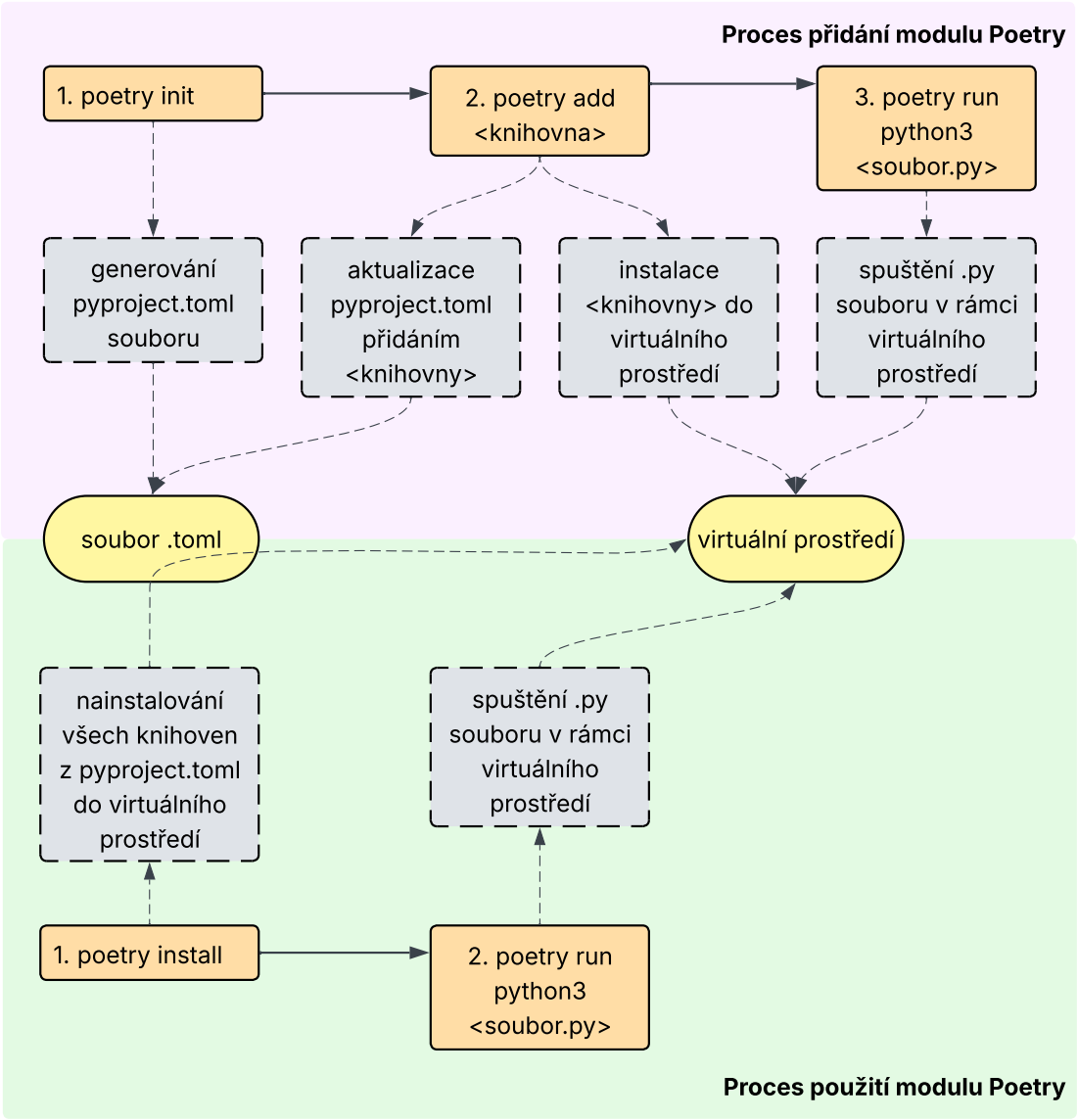
`Dataclass_wizard` poskytuje užitečné mixiny, které zjednodušují práci s datovými třídami. Například `JSONPyWizard` pomáhá zachovat klíče v JSON bez změny (bez převodu do camelCase) a `JSONFileWizard` umožňuje snadnější konverzi instancí datových tříd z/do souborů uložených na lokálním disku. Kromě toho knihovna podporuje typy jako `list`, `dict`, `set`, `Enum`, a různé generické typy z modulu `typing`, což činí práci s datovými třídami více flexibilní.

Dataclass_wizard nabízí snadné rozhraní pro serializaci a deserializaci datových tříd do/z JSON. Pomocí tříd a mixinů jako je `JSONWizard` nebo `property_wizard`, knihovna rozšiřuje možnosti standardních Python datových tříd o funkce pro serializaci a deserializaci, vlastnosti s výchozími hodnotami a další užitečné funkce. Vývojáři mohou pomocí těchto mixinů efektivně a jednoduše převádět své datové třídy mezi formáty a zjednodušit tak práci s datovými modely [29].

4.2 Externí nástroj Poetry

`Poetry` je nástroj pro správu závislostí a balíčkování Python projektů. Umožňuje snadné přidávání knihoven, uzamyká jejich verze v `lockfile` a zajišťuje tak opakovatelné prostředí napříč vývojářským týmem. Automaticky vytváří soubor `pyproject.toml` s deklarací projektu a jeho závislostí.

Poetry pracuje s izolovaným virtuálním prostředím a podporuje jednoduché přidávání závislostí pomocí příkazu `poetry add`. Lze jej využít jak pro vývoj aplikací, tak knihoven. Nabízí režim pro správu závislostí i režim pro distribuci balíčků [30].



Obr. 4.3: Schéma použití nástroje Poetry při generování `pyproject.toml` souboru a spuštění aplikace ve virtuálním prostředí [32].

4.3 Regulární výrazy pro detekci citlivých dat

Regulární výrazy představují mocný nástroj pro vyhledávání a zpracování textu podle specifických vzorů. Využívají se k identifikaci opakujících se struktur v řetězcích, jako jsou e-mailové adresy, telefonní čísla nebo hesla. V kontextu této aplikace slouží jako mechanismus pro automatickou detekci citlivých údajů, které by neměly být odesílány bez patřičného zabezpečení nebo upozornění.

Pomocí knihovny `re` v jazyce Python lze definovat vzory a následně testovat, zda daný text odpovídá požadovaným podmínkám. Takto lze spolehlivě odhalit například čísla platebních karet, rodná čísla nebo adresy elektronické pošty. V rámci aplikace SMAIL je tato technika využita jako bezpečnostní opatření, které upozorní uživatele na možné riziko, pokud e-mail obsahuje potenciálně citlivý obsah.

Zápis regulárních výrazů

Pro správné pochopení fungování regulárních výrazů je důležité rozumět základním konstrukcím, jako jsou kvantifikátory a zástupné znaky. Kvantifikátory určují, kolikrát se má určitý znak nebo skupina znaků v textu opakovat. Je jich několik typů, například:

- `{X}` – kde `X` udává přesný počet opakování daného znaku nebo skupiny.
- `{X,Y}` – kde `X` je minimální a `Y` maximální počet opakování.
- Předdefinované kvantifikátory jako `*`, `+` a `?`, které zjednodušují zápis.

`?` je zkrácený zápis pro rozsah `{0,1}`, `*` odpovídá rozsahu `{0,`∞`}`, a `+` pak `{1,`∞`}`. Tyto předdefinované tvary umožňují kompaktnější a přehlednější zápis regulárních výrazů.

Znak	Význam
`.`	jeden libovolný znak
`*`	žádný nebo více výskytů znaku
`+`	jeden nebo více výskytů znaku
`?`	žádný nebo jeden výskyt znaku (volitelný znak)
`{X}`	přesně `X` opakování znaku (např. `[0-9]{4}` pro čtyři číslice)
`{X,}`	`X` a více opakování znaku
`{X,Y}`	mezi `X` a `Y` opakování znaku

Tab. 4.1: Přehled kvantifikátorů regulárních výrazů [39].

Zástupné znaky zjednodušují zápis výrazů tím, že nahrazují skupiny znaků nebo specifické konstrukce. Například znak `\d` označuje libovolnou číslici v rozsahu 0–9,

a tedy je ekvivalentní výrazu `[0-9]`. Naopak výraz `\D` označuje jakýkoli znak, který číslicí není (tedy `[^0-9]`). V následující tabulce jsou uvedeny nejčastější zástupné znaky a jejich význam:

Znak	**Význam**
`\t`	tabulátor
`\n`	nový řádek
`\b`	začátek nebo konec slova
`\B`	pozice, která není na začátku ani na konci slova
`\d`	číslice (ekvivalentní `[0-9]`)
`\D`	znak, který není číslicí (ekvivalentní `[^0-9]`)
`\w`	písmena, číslice a podtržítko (ekvivalentní `[a-zA-Z0-9_]`)
`\W`	znak, který není písmeno, číslice ani podtržítko
`\s`	bílý znak (mezera, tabulátor, nový řádek apod.)
`\S`	znak, který není bílý znak
`\\`	zpětné lomítko

Tab. 4.2: Přehled základních zástupných znaků používaných v regulárních výrazech [39].

V následující tabulce jsou uvedeny pokročilejší konstrukce používané v regulárních výrazech, které umožňují přesněji definovat, jaké znaky se mají v textu vyhledávat nebo vyloučit:

Znak	**Význam**
`abc`	řetězec `abc`
`[abc]`	jeden ze znaků `a`, `b`, `c`
`[^abc]`	jeden znak kromě `a`, `b`, `c` (negace)
`[a-z]`	malá písmena
`[A-Z]`	velká písmena
`[^A-Za-z0-9]`	symbol (vše kromě písmen a číslic)
`^abc`	výraz `abc` na začátku řetězce
`abc$`	výraz `abc` na konci řetězce
`^abc$`	celý řetězec musí být přesně `abc`

Tab. 4.3: Přehled dalších konstrukcí v regulárních výrazech [39].

Znak `^` (stříška) označuje začátek řetězce, zatímco `$` označuje jeho konec. Pokud je výraz uveden v hranatých závorkách, např. `[a-z]`, označuje výběr z daného rozsahu

znaků. Pokud před znak vložíme `^`, jako ve výrazu `[^abc]`, vyhledávají se všechny znaky kromě uvedených. Rozsahy vycházejí z ASCII tabulky, a tedy například znak „č“ nebude zahrnut do rozsahu `a-z` [39].

Dále je ukázán příklad regulárního výrazu určeného k detekci e-mailových adres:

```
(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)
```

- `^` – začátek řetězce (speciální znak)
- `[a-zA-Z0-9_.+-]` – povolené znaky před zavináčem (znaková třída)
- `+` – jeden nebo více výskytů (kvantifikátor)
- `@` – zavináč jako oddělovač (speciální znak)
- `[a-zA-Z0-9-]` – doménová část (znaková třída)
- `+` – jeden nebo více výskytů (kvantifikátor)
- `\.` – tečka mezi doménou a příponou (uniklý speciální znak)
- `[a-zA-Z0-9-.]` – přípona domény (znaková třída)
- `+` – jeden nebo více výskytů (kvantifikátor)
- `$` – konec řetězce (speciální znak).

Příklad validní e-mailové adresy je poté `xfiala59@vutbr.cz`

Typ dat	**Regulární výraz**	**Příklad**
Číslo kreditní karty (VISA)	`\b4[0-9]{3}([ -]?[0-9]{4}){2}[0-9]{1,4}\b`	4111 1111 1111 1111
Číslo kreditní karty (Mastercard)	`\b5[1-5][0-9]{2}([ -]?[0-9]{4}){3}\b`	5127 0000 0000 0004
Rodné číslo	`\b{6}/{3,4}\b`	981101/1234
Telefonní číslo	`\b{3}[ -]?{3}[ -]?{3}\b`	722 456 789
Poštovní směrovací číslo (PSČ)	`\b{3} ?{2}\b`	592 45

Tab. 4.4: Ukázka regulárních výrazů použitých k detekci citlivých údajů [39].

5 Vytvoření e-mailového klienta

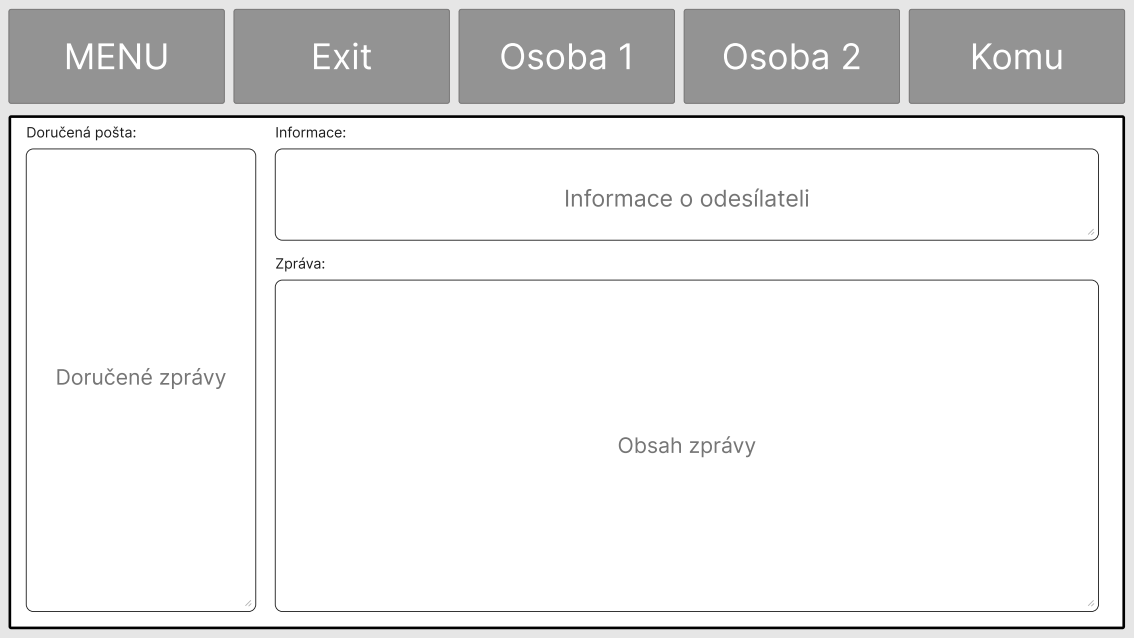
Tato kapitola se zaměřuje na proces vývoje aplikace SMAIL (Emailový klient pro seniory), který byl navržen tak, aby vyhovoval potřebám seniorů a uživatelů s kognitivními omezeními. Aplikace byla vyvíjena v jazyce Python a prostředí pro vývoj bylo zajištěno pomocí editoru PyCharm.

Funkční požadavky na e-mailového klienta zahrnovaly především snadné ovládání, minimalizaci počtu funkcí na ty nejnutnější, jednoduchou navigaci pomocí velkých a srozumitelných tlačítek. Aplikace byla navržena tak, aby uživatelé mohli bezpečně a pohodlně komunikovat se svými blízkými bez nutnosti složitého učení se nových technologií.

5.1 Uživatelské rozhraní

Uživatelské rozhraní aplikace bylo vyvinuto s využitím knihovny `PyQt5`, která je postavena na frameworku Qt. `PyQt5` poskytuje široké možnosti pro tvorbu moderních a přístupných GUI, splňujících požadavky na jednoduché a přehledné ovládání. Aplikace SMAIL byla navržena s důrazem na jednoduchost a maximální srozumitelnost pro cílovou skupinu uživatelů.

Celé rozhraní je rozděleno na několik základních částí:



Obr. 5.1: Základní rozložení aplikace

- **Navigační tlačítka:** Velká a srozumitelná tlačítka pro rychlý přístup k hlavním funkcím, jako je přepínání menu, ukončení aplikace, výběr osoby a univerzální tlačítko pro zadání příjemce e-mailu.
- **Doručené zprávy:** Levá část rozhraní, zobrazující seznam přijatých e-mailů s informacemi o odesílateli a předmětu.
- **Informace o odesílateli:** Horní část pravého panelu, zobrazující detaily odesílatele vybrané zprávy.
- **Obsah zprávy:** Hlavní část pravého panelu, kde se zobrazí tělo vybraného e-mailu.

Struktura rozvržení (layout)

Rozložení uživatelského rozhraní je definováno hierarchií `layoutů`, které určují vzájemné rozmístění jednotlivých prvků. Hlavní rozložení je vertikální (`QVBoxLayout`), které obsahuje horní pás tlačítek a spodní horizontální sekci.

Spodní část je řízena pomocí `QHBoxLayout`, kde je umístěn levý panel se seznamem zpráv a pravý panel s detailem vybrané zprávy. Každý z těchto panelů je dále tvořen samostatnými widgety a dílčími rozvrženími, což umožňuje snadnou údržbu a rozšiřitelnost rozhraní.

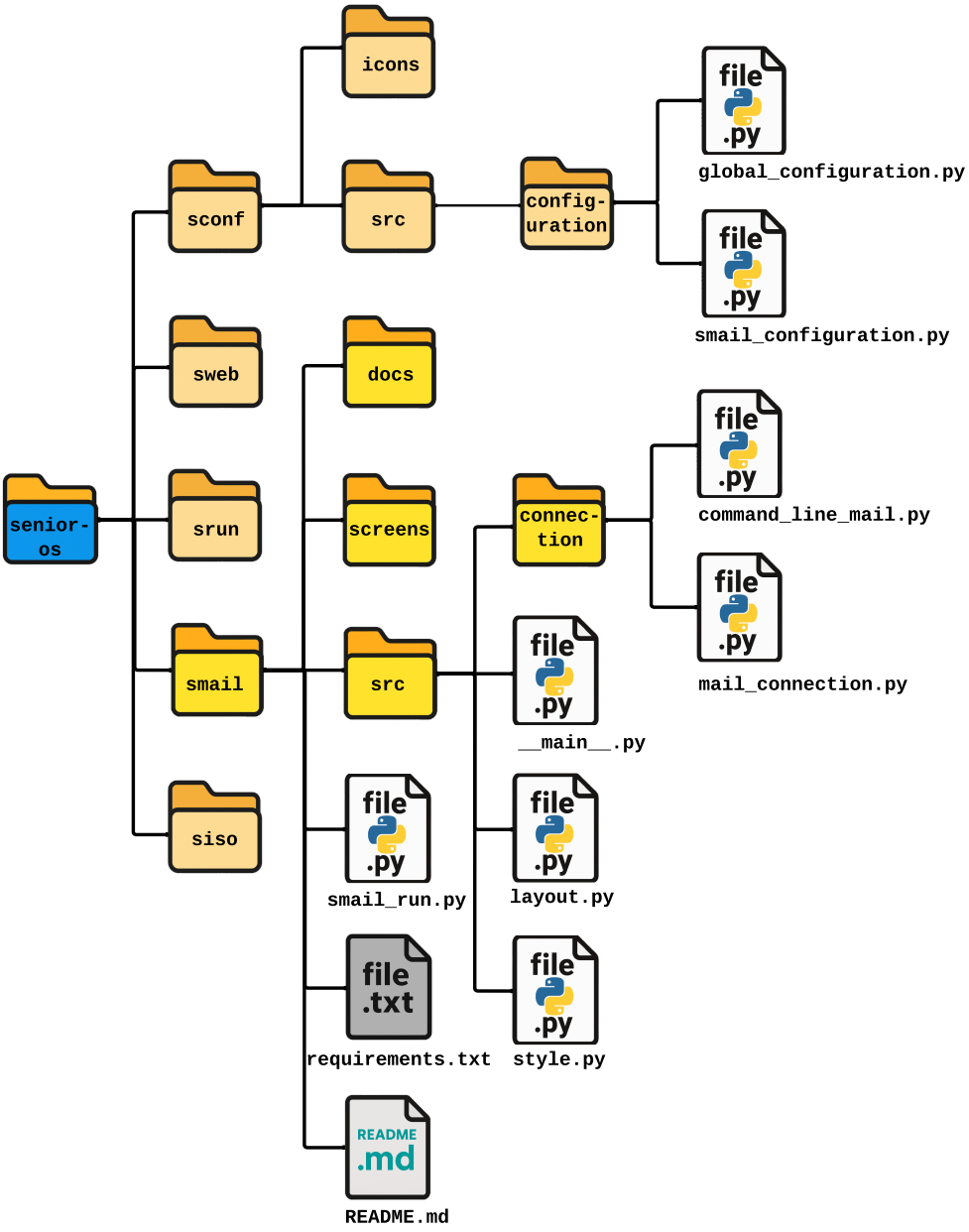
Logika sestavení layoutu je shrnuta v následujícím výpisu:

```
main_layout = QVBoxLayout()  # 0. Vytvoření hlavního vertikálního layoutu
...
self.button_frame_setup() # 1. Horní pás s tlačítky
...
self.bottom_layout = QHBoxLayout() # 2. Spodní část (vodorovné rozložení)
...
self.left_panel_setup() # 3. Levý panel (seznam e-mailů)
self.bottom_layout.addWidget(self.left_panel)  # 4. Přidání levého panelu do spodního layoutu
self.right_panel_setup() # 5. Pravý panel (obsah zprávy)
self.bottom_layout.addWidget(self.right_panel, stretch=1) # 6. Přidání pravého panelu do spodního layoutu
main_layout.addLayout(self.bottom_layout) # 7. Přidání spodního layoutu do hlavního
self.setLayout(main_layout) # 8. Nastavení hlavního layoutu pro okno
```

Výpis 5.1: Sestavení hlavního rozvržení uživatelského rozhraní

5.2 Vnitřní struktura aplikace

Abychom lépe porozuměli organizaci projektu SMAIL, je nejprve vhodné představit základní strukturu jeho adresářů a klíčových souborů.



Obr. 5.2: Struktura adresáře projektu SMAIL

Hlavní vstupní skripty aplikace smail_run.py a ___main___.py

Soubory `smail_run.py` a `__main__.py` společně představují hlavní vstupní bod aplikace SMAIL, který se stará o její inicializaci a spuštění. Níže je popsán jejich

účel a způsob fungování.

Skript `smail_run.py` poskytuje flexibilitu pro spuštění aplikace pomocí modulu `runpy`. Obsahuje funkci `main()`, která volá metodu `run_module()` pro spuštění celého modulu `smail`. Soubor ___main___.py je hlavním vstupním bodem aplikace a obsahuje klíčovou funkci `main()`, která se stará o:

V rámci funkce `main()` se nejprve načítá konfigurace aplikace z konfiguračního souboru `config.json`, pomocí modulu ConfigurationProvider z `sconf.configuration.-configuration_provider`. Tento krok je zásadní pro získání potřebných parametrů, jako jsou přihlašovací údaje a další nastavení aplikace.

Po načtení konfigurace je vytvořeno hlavní uživatelské rozhraní aplikace pomocí knihovny `PyQt5`. Hlavní okno je vytvořeno jako instance třídy `QMainWindow` a nastavuje se bez rámce (`Qt.FramelessWindowHint`), což zajišťuje vizuální čistotu a jednoduchost rozhraní pro cílovou skupinu uživatelů.

Rozvržení hlavního okna je definováno funkcí `first_frame()` ze souboru `layout.py`, která zajišťuje umístění a inicializaci všech důležitých prvků GUI, jako jsou tlačítka, seznam zpráv a textová pole.

Po dokončení nastavení rozhraní se hlavní okno aplikace zobrazí voláním metody `show()`. Aplikace poté vstoupí do hlavního cyklu událostí, dokud není ukončena. Celý proces je obalen do bloku `try...except`, aby se případné chyby (např. problémy s konfigurací) zachytily a informovaly uživatele nebo vývojáře o příčině selhání.

Blok `if __name__ == '__main__':` zajišťuje, že funkce `main()` se spustí pouze tehdy, když je tento soubor spuštěn jako hlavní skript, což umožňuje využití modulárního kódu v jiných částech projektu bez opakovaného spouštění hlavní logiky.

Tvorba hlavního grafického prostředí pomocí layout.py

Soubor `layout.py` obsahuje kompletní definici rozvržení uživatelského rozhraní aplikace SMAIL. Tento soubor hraje klíčovou roli v tvorbě interaktivního prostředí, ve kterém uživatelé aplikace komunikují s e-maily. Celkové rozložení a funkce aplikace byly navrženy s ohledem na jednoduchost ovládání a přehlednost.

Hlavní částí souboru je třída `first_frame`, která představuje základní komponentu uživatelského rozhraní aplikace. Tato třída je zodpovědná za vytvoření a řízení všech hlavních prvků GUI, a to včetně nastavení barevného schématu, rozložení tlačítek, načítání e-mailů, a zobrazování jejich obsahu.

Třída `first_frame` rovněž zajišťuje interakci uživatele s e-maily. E-maily se načítají pomocí funkce `load_emails()` a jsou pravidelně aktualizovány, aby uživatel měl k dispozici aktuální obsah doručené pošty. Při výběru e-mailu ze seznamu na

levém panelu je obsah zprávy zobrazen na pravém panelu, což umožňuje přehlednou práci s doručenými zprávami.

Každé tlačítko v horním panelu má specifickou funkci, která je definována metodou `buttons_setup()`. Tlačítka umožňují například psaní nových e-mailů, volbu příjemců, ukončení aplikace a přepínání mezi menu. Při práci s e-mailem mohou uživatelé zadat adresu příjemce, předmět a obsah zprávy, a následně e-mail odeslat. Tato funkcionalita je navržena tak, aby byla co nejvíce intuitivní a odpovídala potřebám seniorů.

Součástí funkcionality třídy jsou i mechanismy pro potvrzení akcí a upozornění na citlivý obsah. Například při zjištění, že obsah e-mailu obsahuje citlivá data (např. heslo), je uživatel upozorněn, což snižuje riziko náhodného sdílení citlivých informací. Tyto upozornění a ověřovací kroky zajišťují, že uživatel je chráněn před možnými bezpečnostními problémy.

Nastavení vzhledu uživatelského rozhraní pomocí style.py

Soubor `style.py` slouží jako klíčový modul pro definování vzhledu a vizuální prezentace aplikace SMAIL. Obsahuje všechny potřebné metody a konfigurace, které zajišťují jednotný styl aplikace, včetně barevných schémat, stylů tlačítek a textových polí, což přispívá k přehlednosti a snadnému ovládání uživatelského rozhraní.

Barevné schéma zahrnuje základní barvy pro různé akce, jako jsou pozitivní akce (zelená), upozornění (červená), a neutrální prvky (odstíny šedé a bílá). Tyto barvy zajišťují konzistentní vizuální dojem a snadné rozpoznávání funkcionalit pro uživatele.

Soubor také poskytuje metody pro definici vzhledu hlavních částí uživatelského rozhraní:

- `get_left_panel_style()`, `get_right_panel_style()`: Tyto metody definují vzhled a rozvržení hlavních panelů v aplikaci, včetně levého panelu pro seznam zpráv a pravého panelu pro zobrazení obsahu e-mailů.
- `get_button_frame_style()`: Definuje vzhled horního rámce aplikace, kde jsou umístěna tlačítka pro různé funkce.

Soubor `style.py` poskytuje metodu `images()`, která načítá cesty k obrázkům a ikonám používaným v aplikaci. Tyto obrázky jsou používány jako ikony na tlačítkách (např. pro výběr osoby nebo pro ukončení aplikace), což přispívá k intuitivnímu vizuálnímu rozhraní.

Kromě vizuálních prvků zahrnuje `style.py` metody, které umožňují aplikaci získávat konfiguraci zadanou uživatelem. Například:

- `load_credentials(data_provider)`: Načítá přihlašovací údaje a informace potřebné pro připojení k e-mailovému serveru (jako jsou přihlašovací jméno,

heslo, SMTP a IMAP servery).

- `get_language(data_provider)`: Poskytuje aktuální nastavení jazyka aplikace, což zajišťuje podporu lokalizace.
- `search_mail(id, data_provider)`: Umožňuje načítání e-mailových adres kontaktů na základě ID, což zjednodušuje výběr příjemce při psaní e-mailu.

Komunikace se serverem v mail_connection.py

Soubor se stará o všechny operace související s komunikací mezi aplikací a e-mailovými servery. Zajišťuje odesílání a načítání e-mailů, připojení k serverům pomocí protokolů SMTP a IMAP, a správu e-mailových zpráv. Zároveň je zde obsažen mechanismus pro přeposílání e-mailů, který umožňuje sdílení informací s pečovatelem. Níže jsou popsány hlavní funkce obsažené v tomto souboru.

Funkce `send_email()` zajišťuje odesílání e-mailů prostřednictvím protokolu SMTP. Tato operace vyžaduje přihlášení k SMTP serveru pomocí přihlašovacích údajů získaných z konfiguračního souboru. Obsahuje mechanismy pro bezpečné připojení k SMTP serveru pomocí protokolu TLS a podporuje autentizaci pomocí e-mailové adresy a hesla.

Funkce `imap_connection()` umožňuje připojení k IMAP serveru pro načítání e-mailů. Používá protokol SSL pro zabezpečení připojení a zajišťuje bezpečný přístup k poštovní schránce uživatele. Pokud dojde k chybě (například chybné přihlašovací údaje nebo špatný server), funkce vrací chybu a informuje uživatele o problému.

Funkce `read_mail()` slouží pro připojení k IMAP serveru a načítání doručené pošty z `INBOX`. Díky tomu je možné zobrazit všechny přijaté zprávy uživateli.

- Funkce se připojí k IMAP serveru a prochází všechny zprávy ve složce `INBOX`.
- Pomocí knihovny `email` a `chardet` dekóduje textové části zpráv, včetně předmětu, odesílatele, datumu a obsahu.
- Načtené zprávy jsou následně zobrazeny uživateli, aby mohli snadno číst své doručené e-maily.

Součástí aplikace je také mechanismus pro odeslání kopie e-mailů pečovateli, což je zajištěno funkcí `send_email_with_guardian- _copy()`. Tato funkce je užitečná zejména v případě, kdy je nutné zajistit, aby komunikace uživatele byla pod dohledem, například pro zajištění jejich bezpečnosti nebo sledování podezřelé aktivity.

Odesílání e-mailů z příkazové řádky (command_line_mail.py)

Tento soubor umožňuje odesílání e-mailů pomocí příkazového řádku. Tento skript je navržen pro situace, kdy je potřeba rychle odeslat zprávu, aniž by bylo nutné spustit grafické uživatelské rozhraní aplikace. Níže je popis hlavních funkcionalit skriptu.

Stejně jako ostatní části aplikace, tento skript využívá konfigurační soubor `config.json` k načtení potřebných údajů o e-mailovém serveru a přihlašovacích údajích. Konfigurace je načítána pomocí modulu `ConfigurationProvider`, který zajišťuje správné načtení informací jako je e-mailová adresa, heslo, SMTP server a port.

Tato funkce slouží k odesílání e-mailů na zadanou adresu příjemce. Odesílání je realizováno pomocí protokolu SMTP a zahrnuje tyto kroky:

- Načtení přihlašovacích údajů a informací o SMTP serveru z konfiguračního souboru.
- Vytvoření zprávy pomocí `MIMEText`, včetně vyplnění předmětu, odesílatele a příjemce.
- Připojení k SMTP serveru přes zabezpečené připojení pomocí TLS, přihlášení k serveru a odeslání zprávy.
- V případě úspěšného odeslání se vypíše hlášení o úspěšném odeslání zprávy.

Skript je navržen tak, aby byl spouštěn přímo z příkazové řádky, přičemž e-mailová adresa příjemce a obsah e-mailu jsou zadávány uživatelem. Na začátku souboru se kontroluje, zda byly zadány správné argumenty (adresa příjemce a obsah zprávy), a pokud tomu tak není, uživateli je zobrazen návod, jak skript správně použít:

```
Usage: python command_line_mail.py <recipient_email> „<email_content>"
```

Blok `if __name__ == „__main__"` zajišťuje, že skript je spuštěn pouze v případě, že je vykonáván přímo jako hlavní program. To znamená, že uživatel může zadat e-mailovou adresu příjemce a obsah zprávy přímo při spuštění skriptu z příkazového řádku. Pokud nebyly zadány správné argumenty, skript zobrazí zprávu s nápovědou a ukončí běh.

5.3 Funkce aplikace

Architektura aplikace SMAIL se skládá z několika klíčových částí, které jsou oddělené do samostatných modulů, což podporuje přehlednost kódu a snadnou údržbu. V aplikaci jsou následující klíčové soubory:

- **___main___.py:** Obsahuje logiku pro spuštění aplikace a inicializaci hlavního okna.
- **layout.py:** Obsahuje veškerý funkční kód, který zajišťuje kompletní rozvržení uživatelského rozhraní aplikace.
- **style.py:** Obsahuje definici stylů aplikace, jako jsou barvy, fonty a velikosti prvků uživatelského rozhraní.
- **mail_connection.py:** Slouží k implementaci připojení k e-mailovýmu serveru.
- **command_line_mail.py:** Umožňuje komunikaci prostřednictvím příkazové řádky.
- **config.json:** Obsahuje nastavení parametrů aplikace.

5.3.1 Jazyková lokalizace rozhraní

Aplikace podporuje vícejazyčné uživatelské rozhraní, které umožňuje přepínání mezi češtinou, angličtinou a němčinou. Volba jazyka se provádí v rámci konfigurační části aplikace `SCONF`, kde uživatel nastaví preferovaný jazyk. Na základě tohoto nastavení jsou načteny odpovídající jazykové texty pro veškeré popisky a hlášky v grafickém rozhraní.

Všechny jazykové řetězce jsou uloženy ve třídě `LanguageSet`, která se nachází v konfiguračním souboru `smail_configuration.py`. Přístup k těmto údajům je realizován pomocí instance třídy `data_provider`. Díky tomu je možné snadno měnit nebo rozšiřovat jazykovou podporu bez zásahu do logiky aplikace.

Klíč	**Čeština (Cz)**	**Angličtina (En)**	**Němčina (De)**
`smailXxSendToButton`	Komu	Send To	Senden An
`smailXxLoadingInbox`	Načítám...	Loading...	Laden...
`smailXxRecipientLabel`	Příjemce:	To:	An:
`smailXxMessageLabel`	Zpráva:	Message:	Nachricht:
`smailXxFrom`	Informace:	Information:	Information:

Tab. 5.1: Přehled jazykových variant uživatelského rozhraní

5.3.2 Navigační panel a chování tlačítek

Navigační panel v horní části aplikace slouží k výběru adresáta, přepínání mezi dvěma hlavními menu, odeslání zprávy a vypnutí aplikace. Tlačítka jsou zobrazená graficky pomocí avatarů či fotek seniora a textových polí, přičemž jejich jazyk i ikony lze přizpůsobit v konfigurační části aplikace. Tlačítka reagují na různé stavy aplikace, mění svůj vzhled a chování v závislosti na zvoleném adresátovi nebo na aktivní úrovni ochrany.

Základní podoba navigačního menu

Obr. 5.3: Výchozí vzhled prvního menu

Obr. 5.4: Výchozí vzhled druhého menu

Každé menu se skládá z pěti tlačítek:

- První tlačítko `MENU 1/ MENU 2` slouží k přepínání mezi dvěma sadami kontaktů.
- Druhé tlačítko ukončuje aplikaci.
- Třetí až páté tlačítko reprezentuje adresáty. Při prvním kliknutí se vyplní pole příjemce, při druhém dojde ke spuštění odesílacího procesu.

```
def buttons_setup(self):
    button_9 = getattr(self.text_configuration, f"smail{self.
        language.capitalize()}SendToButton")  # Lokalizovaný text
        tlačítka

    button_menu1 = ["MENU 1", "EXIT", "Button 2", "Button 3", "
        Button 4"]  # Texty tlačítek pro první menu
    button_menu2 = ["MENU 2", "Button 6", "Button 7", "Button 8",
        button_9]  # Druhá sada tlačítek

    spacer_left = QSpacerItem(20, 40, QSizePolicy.Expanding,
        QSizePolicy.Minimum)  # Vycentrování tlačítek
    ...
    for index, text in enumerate(current_menu):
        button = QPushButton(text, self.button_frame)  # Vytvoření
            tlačítka s daným textem
        button.setStyleSheet(style.get_button_style(self.
            data_provider))  # Stylování tlačítka
        self.button_layout.addWidget(button, alignment=Qt.
            AlignCenter)  # Umístění tlačítka na panel

        if self.menu1:
            if index == 0:
                button.clicked.connect(lambda _, btn=button, nb=
                    None: self.decide_action_for_button(btn, nb))  #
                     Zpětné přepnutí
            elif index == 1:
                button.setText("")
                button.setIconSize(QSize(72, 72))  # Nastavení
                    velikosti ikony
                button.setIcon(self.exit_image)  # Ikona pro ukonč
                    ení aplikace
                button.clicked.connect(self.exit_app)
            ...
        else:
            # Obdobné nastavení tlačítek pro druhé menu
            ...
```

Výpis 5.2: Funkce pro nastavení tlačítek v horním panelu

Funkce `buttons_setup` má na starosti kompletní vytvoření a rozmístění tlačítek v horním panelu aplikace. Dle aktivního menu (první nebo druhé) generuje odpovídající sadu tlačítek, nastavuje jejich styl, text nebo ikonu a přiřazuje jim konkrétní chování při kliknutí.

Na začátku funkce je načten lokalizovaný text pro jedno z tlačítek (Komu), který se mění v závislosti na nastaveném jazyce uživatele. Dále jsou definovány dvě varianty menu – první menu (MENU 1) obsahuje základní funkce jako „EXIT“ a adresáti 1–3, druhé menu (MENU 2) pak adresáty 4–6 a univerzální tlačítko pro odeslání zprávy libovolnému příjemci.

Před přidáním samotných tlačítek jsou vloženy tzv. `spacery`, které zajišťují vycentrování tlačítek v rámci panelu. Následně probíhá iterace přes seznam aktuálního menu, kde je pro každý prvek vytvořeno nové tlačítko `QPushButton`.

V případě prvního menu je jednotlivým tlačítkům přiřazeno chování:

- první tlačítko přepíná mezi menu,
- druhé tlačítko ukončuje aplikaci a zobrazuje ikonu „exit“,
- další tři tlačítka reprezentují různé adresáty a zobrazují jejich avatar.

Druhé menu má obdobné chování, s tím rozdílem, že poslední tlačítko je při vyšším stupni ochrany (např. Úroveň ochrany 2) deaktivováno. V takovém případě se místo textu „Komu“ zobrazí text „Uzamčeno“ a tlačítko není klikatelné. V opačném případě je i tomuto tlačítku přiřazeno chování odesílání zprávy.

Na závěr funkce se nastaví mezery mezi tlačítky (`setSpacing`), přidá se pravý `spacer` a určí se vnější odsazení tlačítkového panelu (`setContentsMargins`).

Funkce tímto způsobem zajišťuje konzistentní, lokalizované a přehledné ovládání aplikace podle zvoleného režimu menu, aktivního jazyka i bezpečnostních pravidel.

Jazykové přizpůsobení a změna ikon

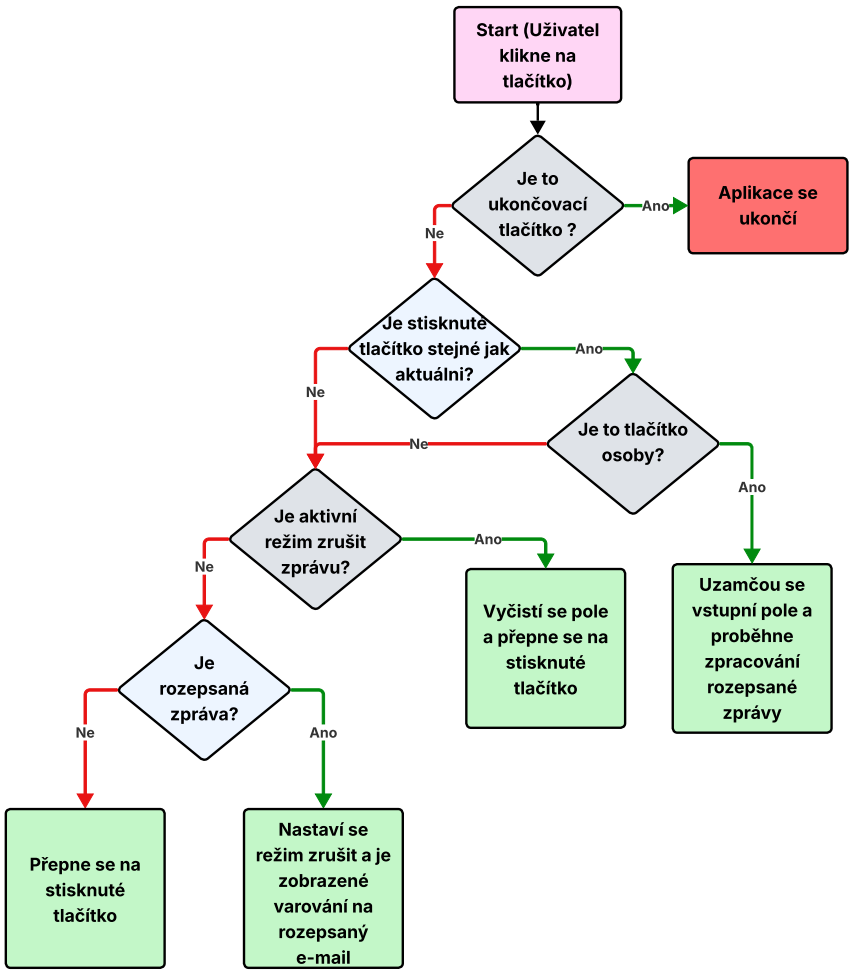
Texty tlačítek se přizpůsobují zvolenému jazyku dle nastavení v konfigurační části aplikace. Stejně tak lze zvolit různé sady ikon, např. kreslené postavy nebo reálné fotografie.

Obr. 5.5: Menu 2 v angličtině s reálnými fotografiemi ikon

Obr. 5.6: Menu 2 s kreslenými ikonami v češtině

Logika chování tlačítek

Tlačítka mají různé reakce podle stavu aplikace, např. při výběru kontaktu, zrušení rozepsané zprávy nebo při potvrzení odeslání. Rozhodovací logika je shrnuta v následujícím schématu:



Obr. 5.7: Rozhodovací logika tlačítek

Změna vzhledu při výběru adresáta

Po kliknutí na některého z dostupných adresátů v navigačním menu se příslušné tlačítko vizuálně zvýrazní změnou pozadí. Toto zvýraznění slouží k jasnému odlišení aktuálně vybraného kontaktu a zároveň informuje uživatele o tom, ke komu bude případná zpráva směřována.

Obr. 5.8: Zvýraznění vybraného kontaktu

```python
def get_button_style(data_provider=None, normal=True, highlight=False):
    # Získání barevného schématu z konfigurační části
    colors = get_color_scheme(data_provider)
    # Výběr barvy pozadí dle stavu tlačítka
    if highlight:
        background_color = colors["highlight_color"]  # např. zelená pro potvrzení
    elif normal:
        background_color = colors["dark_grey_color"]  # neutrální výchozí šedá
    else:
        background_color = colors["alert_color"]  # červená pro výstrahu
    return f"""
        QPushButton {{
            background-color: {background_color};  /* Výchozí barva tlačítka */
        }}
        QPushButton:pressed {{
            background-color: {colors["highlight_color"]};  # Barva při stisknutí
        }}
        QPushButton:hover {{
            background-color: {colors["highlight_color"]};  # Barva při přejetí myší (hover)
        }}
        QPushButton::icon {{
            alignment: {button_position};  # Zarovnání ikony (např. na střed)
        }}
    """
```

Výpis 5.3: Získání stylu pro zvýrazněné tlačítko

Funkce pro změnu stylu tlačítka využívá definici `get_button_style`, která vrací specifický CSS-like styl v závislosti na stavu tlačítka. Parametr `highlight=True` zajistí přiřazení zvýrazňující barvy (např. zelené), která je definována jako `highlight_color`

v aktivním barevném schématu. Tlačítko tak vizuálně vystupuje z ostatních a indikuje výběr. Zvýrazněná barva se použije nejen pro výchozí zobrazení, ale i při najetí kurzorem nebo stisknutí tlačítka.
Barvy, které se používají při zvýraznění tlačítek, nejsou pevně zakódované – lze je upravit v konfigurační části aplikace. Výchozí nastavení obsahuje:

- `highlight_color` – zvýrazňovací barva (standardně zelená),
- `alert_color` – varovná barva (standardně červená),
- `dark_grey_color` – základní neutrální barva tlačítek.

Tyto barvy jsou definovány ve funkci `get_color_scheme`, která je volána při generování stylu v `get_button_style()`. Uživatel tak může barvy snadno přizpůsobit dle vizuálních preferencí v konfigurační části aplikace SCONF.

Výstražný režim

Při varování (například kvůli detekci citlivých dat, neúspěšnému odeslání zprávy nebo pokusu opustit neuložený koncept) se celé navigační menu vizuálně přepne do výstražného režimu. V tomto stavu jsou všechna tlačítka obarvena do červené (výstražné) barvy a ikony do černobílé barvy, což slouží jako důležitý vizuální signál pro uživatele, že je třeba zvýšené pozornosti.

Obr. 5.9: Výstražný režim tlačítek – barva signalizuje chybu nebo varování

Tento režim je řízen funkcí `alert_buttons`, která nastavuje nebo vrací původní styl tlačítek. Pokud je parametr `alert=True`, tlačítka přepne do výstražného vzhledu; jinak obnoví jejich původní styl. Využívá k tomu i vnitřní paměť `original_button_styles`, aby se po odeznění varování vrátily tlačítka do své původní podoby.

```
def alert_buttons(self, alert=True):
    """
    Mění vzhled tlačítek na výstražný (červený) režim.
    Pokud je varování zrušeno, obnoví původní vzhled tlačítek.
    """
    if not hasattr(self, "original_button_styles"):
        # Inicializace úložiště původních stylů
        self.original_button_styles = {}

    # Prochází všechna tlačítka v layoutu
    for i in range(self.button_layout.count()):
        widget = self.button_layout.itemAt(i).widget()

        if isinstance(widget, QPushButton):
            if alert:
                # Uloží původní styl tlačítka (pokud ještě není ulo
                    žen)
                if widget not in self.original_button_styles:
                    self.original_button_styles[widget] = widget.
                        styleSheet()

                # Aplikuje výstražný styl (např. červený)
                widget.setStyleSheet(style.get_button_style(self.
                    data_provider, normal=False))
            else:
                # Obnoví původní vzhled tlačítka
                if widget in self.original_button_styles:
                    widget.setStyleSheet(self.
                        original_button_styles[widget])
```

Výpis 5.4: Funkce pro aktivaci a deaktivaci výstražného režimu

5.3.3 Načítání e-mailů na pozadí

Pro zvýšení plynulosti uživatelského rozhraní je načítání e-mailů implementováno pomocí samostatného vlákna s využitím `QThread`. Tento přístup zajišťuje, že během načítání e-mailů nedochází k blokování hlavního GUI vlákna.

Proces probíhá následovně:

- Při spuštění aplikace je každých 10 sekund volána metoda `insert_emails()`, která spouští načítání e-mailů na pozadí.
- V metodě `insert_emails()` je vytvořeno nové vlákno typu `QThread` a pracovní objekt `EmailLoaderWorker`.
- Pracovní objekt se přesune do vlákna pomocí `moveToThread()`.
- Signály jsou připojeny tak, aby bylo spuštěno načítání, předány načtené e-maily a zachyceny případné chyby.
- Po dokončení načítání se vlákno ukončí a uvolní.

```
# Spuštění první načítání e-mailů při startu
self.insert_emails()

# Nastavení časovače pro pravidelné načítání e-mailů
self.email_timer = QTimer(self)

# Po vypršení intervalu se znovu zavolá metoda insert_emails
self.email_timer.timeout.connect(self.insert_emails)

# Spuštění časovače s intervalem 10 sekund (10 000 ms)
self.email_timer.start(10000)

# Povolení zobrazení e-mailů v seznamu po jejich načtení
self.allow_show_email = True
```

Výpis 5.5: Spuštění periodického načítání e-mailů pomocí časovače

Třída `EmailLoaderWorker`

Tato třída dědí od `QObject` a definuje dva signály:

- `emails_loaded` – signál vysílaný po úspěšném načtení e-mailů (předává seznam e-mailů a předmětů).
- `error_occurred` – signál vysílaný v případě chyby při načítání e-mailů (předává text chyby).

Metoda `run()` obsahuje vlastní logiku pro:

- načtení přihlašovacích údajů,
- získání e-mailů pomocí funkce `read_mail()`,
- ošetření chyb.

```python
class EmailLoaderWorker(QObject): # Definice signálu, který je emitován při úspěšném načtení e-mailů
    emails_loaded = pyqtSignal(list, list) # Definice signálu, který je emitován při chybě při načítání e-mailů
    error_occurred = pyqtSignal(str)

    def __init__(self, data_provider, text_configuration):
        super().__init__()
        self.data_provider = data_provider  # Zdroj dat, např. konfigurace
        self.text_configuration = text_configuration  # Textová konfigurace podle jazyka uživatele

    def run(self):
        try:
            login, password, smtp_server, smtp_port, imap_server, imap_port = style.load_credentials(self.data_provider) # Načte přihlašovací údaje a servery
            language, text = style.get_language(self.data_provider) # Zjistí jazykové nastavení aplikace

            emails, subjects = read_mail(login, password, imap_server, imap_port, language, text, self.data_provider) # Načte e-maily a jejich předměty z IMAP serveru

            if emails is None or subjects is None:   # Pokud se e-maily nepodaří načíst, oznámí chybu
                self.error_occurred.emit("Failed to load emails.")
                return

            self.emails_loaded.emit(emails, subjects)  # Pokud bylo vše úspěšné, vyšle signál s načtenými e-maily

        except Exception as e:
            self.error_occurred.emit(str(e)) # Při jakékoliv výjimce vyšle signál s chybovým hlášením
```

Výpis 5.6: Třída EmailLoaderWorker pro načítání e-mailů na pozadí

Získávání e-mailů pomocí IMAP

Stahování e-mailových zpráv probíhá v modulu `mail_connection.py` prostřednictvím funkce `read_mail`. Nejprve se naváže šifrované spojení s IMAP serverem pomocí funkce `imap_connection`, kde je využita knihovna `imaplib` a SSL kontext pro zabezpečení.

```
def imap_connection(login, password, imap_server, imap_port):
    try:
        # Navázání šifrovaného spojení pomocí IMAP protokolu přes SSL
        mail = imaplib.IMAP4_SSL(
            imap_server, imap_port, ssl_context=ssl.create_default_context()
        )
        mail.login(login, password)  # Přihlášení pomocí zadaných údajů
        return mail  # Vrací aktivní spojení
    except imaplib.IMAP4.error:
        # Chyba na úrovni IMAP protokolu (např. špatné přihlašovací údaje)
        return 0
    except ConnectionError:
        # Problém s připojením k serveru (např. nedostupná síť)
        return -1
    except Exception as error:
        # Obecná neočekávaná chyba
        return -2
```

Výpis 5.7: Navázání šifrovaného spojení s IMAP serverem

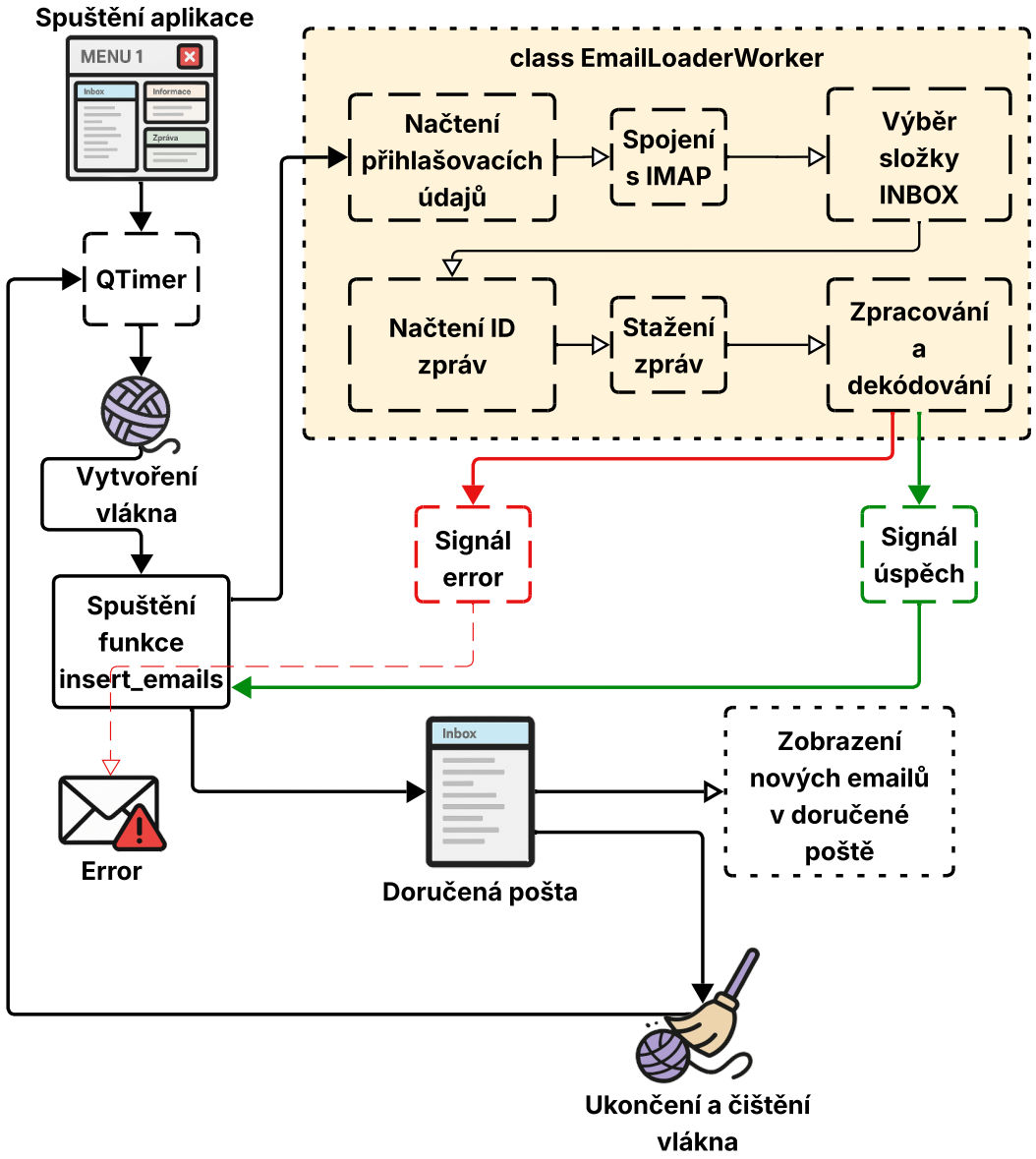
Po úspěšném přihlášení je vybrána složka `INBOX`, ze které se načtou všechna dostupná ID zpráv. Každá zpráva je poté stažena ve formátu `RFC822` a následně zpracována:

Zpracování výsledků

Po načtení e-mailů jsou zavolány následující metody:

- `on_emails_loaded(emails, subjects)` – přijímá načtené e-maily a předměty, porovnává je s předchozími a aktualizuje obsah seznamu v uživatelském rozhraní.
- `on_email_error(message)` – zobrazí chybovou zprávu v případě selhání načítání.

Díky tomuto přístupu je uživatelské rozhraní stále responzivní a načítání e-mailů probíhá transparentně na pozadí.



Obr. 5.10: Schéma načítání e-mailů na pozadí

5.3.4 Zpracování a zobrazení obsahu e-mailové zprávy

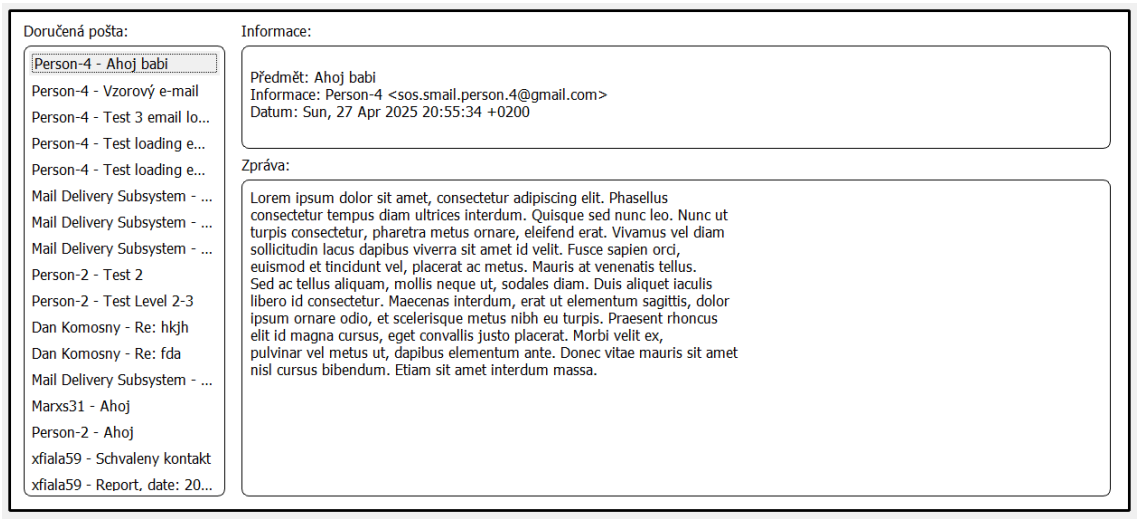
Načítání a zobrazování obsahu e-mailových zpráv je realizováno pomocí několika propojených funkcí, které zajišťují bezpečné stažení, dekódování a zobrazení obsahu v uživatelském rozhraní aplikace.

Zobrazení zprávy v rozhraní

Po kliknutí na položku v seznamu doručených e-mailů je vyvolána funkce `show_email()`, která provádí načtení vybrané zprávy a její zobrazení v pravém panelu:

```
selected_index = self.inbox_list.currentRow()
selected_email = self.all_emails[selected_index]
email_content = selected_email[0]
email_lines = email_content.split("\n")

# Extrakce základních informací
email_subject_line = email_lines[0]
email_sender_line = email_lines[1]
email_date_line = email_lines[2]
email_message_content = "\n".join(email_lines[4:])

# Nastavení obsahu pravého panelu
self.configure_message_area(email_message_content,
    email_subject_line, email_sender_line, email_date_line)
```

Výpis 5.8: Zobrazení obsahu vybraného e-mailu v rozhraní



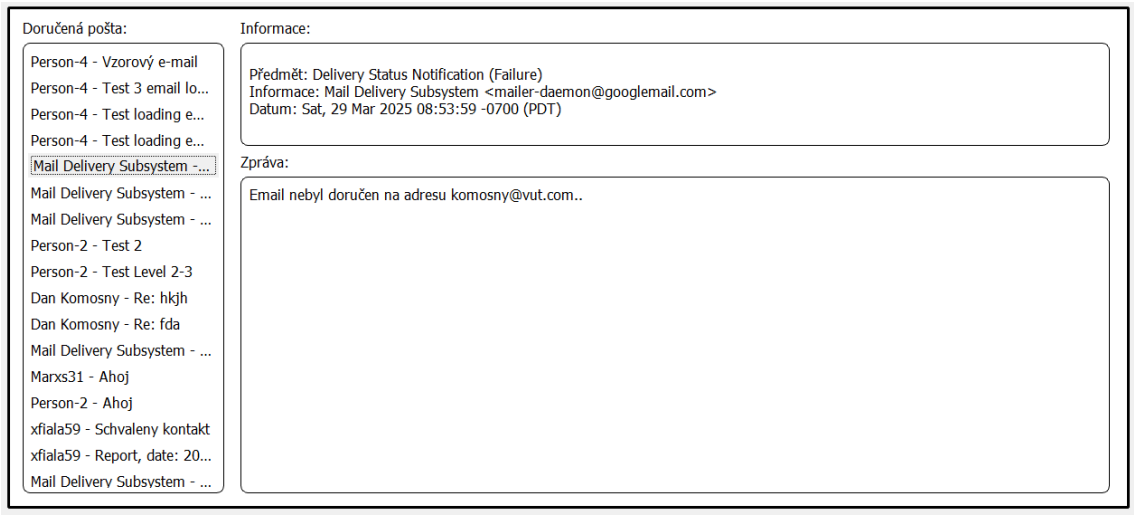
Obr. 5.11: Snímek zobrazeného e-mailu

Úprava zobrazení při nedoručení zprávy

Pokud systém zjistí, že se jedná o zprávu o nedoručení (například pokud je v odesílateli uvedeno `Mail Delivery Subsystem` nebo text obsahuje `Message not delivered`), obsah se nahradí varovnou zprávou:

```
simplified_content = email_content
if "Message not delivered" in email_content or "Mail Delivery
    Subsystem" in email_sender:
     # Vyhledání e-mailu příjemce, kterému nebylo doručeno
     recipient_email_match = re.search(r"to\s+([\w\.-]+@[\w\.-]+)",
         email_content)
     if recipient_email_match:
         recipient_email = recipient_email_match.group(1)
         # Přepsání zprávy lokalizovaným varováním
         simplified_content = getattr(self.text_configuration,
                                      f"smail{self.language.
                                          capitalize()}
                                          UndeliveredEmail").format(
             recipient_email=recipient_email)
```

Výpis 5.9: Detekce a nahrazení zprávy o nedoručení

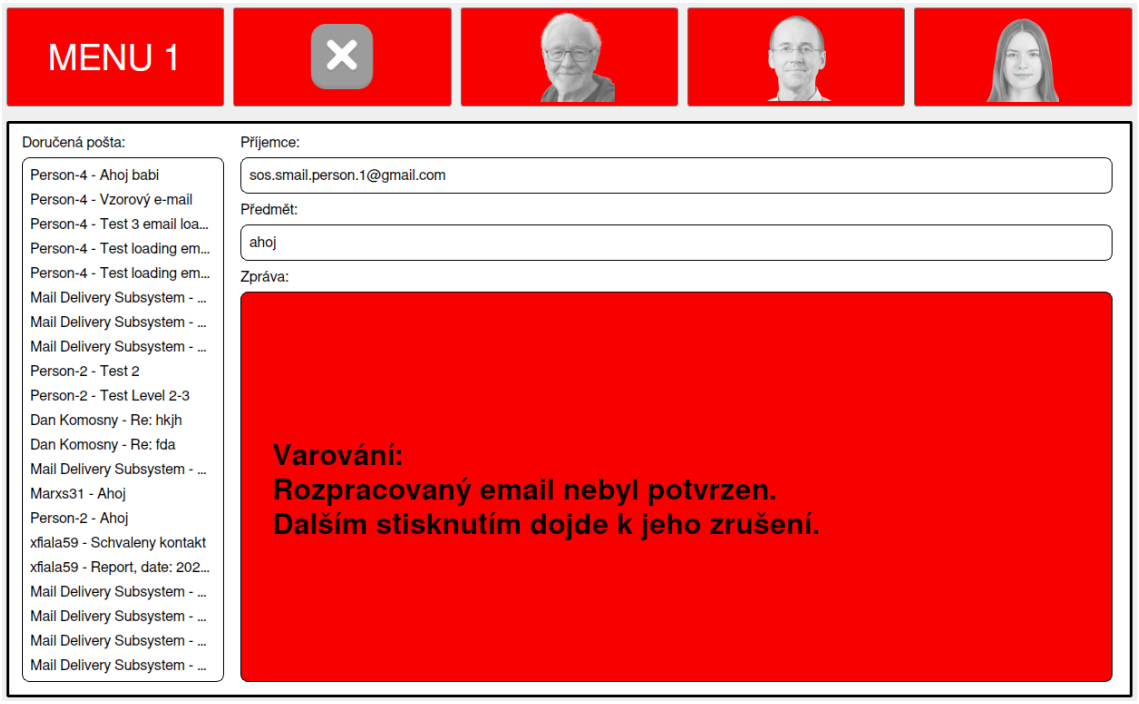


Obr. 5.12: Snímek zprávy oznamující nedourčený e-mail

Výsledné zobrazení je tím pádem jednodušší a uživatelsky přívětivější, bez potřeby číst technické detaily odesílací chyby. Tento mechanismus načítání a zobrazování e-mailů umožňuje seniorům snadno se orientovat ve své poště bez nutnosti řešit složitosti spojené s dekódováním, různými formáty znakových sad či chybovými zprávami.

5.3.5 Nepotvrzený rozepsaný e-mail

Aplikace chrání uživatele před neúmyslným ztracením rozepsané zprávy. Pokud se uživatel pokusí přepnout do jiného režimu nebo opustit rozepsaný e-mail bez odeslání, je zobrazeno výrazné upozornění přímo v editačním poli zprávy. Celé rozhraní přejde do výstražného režimu a po několika sekundách se buď obnoví původní obsah, nebo čeká na další akci uživatele.



Obr. 5.13: Upozornění na nepotvrzený rozepsaný e-mail

Po aktivaci výstražného režimu se:

- deaktivují všechna tlačítka,
- zobrazí varování přímo ve zprávě na červeném pozadí,
- automaticky po uplynutí časového limitu (4 sekundy) obnoví původní text e-mailu a aktivují tlačítka.

Tento mechanismus výrazně snižuje riziko ztráty důležitých rozepsaných zpráv a zvyšuje bezpečnost práce v aplikaci SMAIL.Tato logika je implementována ve funkci `alert_unconfirmed_email()`:

```
def alert_unconfirmed_email(self):
    # Nastavení barev výstražného režimu
    colors = self.color_scheme
    alert_color = colors["alert_color"]
    default_color = colors["default_color"]

    # Uložení původního obsahu e-mailu
    original_content = self.email_content_label_2.toPlainText()

    # Aktivace varovného režimu tlačítek
    self.alert_buttons(alert=True)

    # Příprava oblasti zprávy na vložení upozornění
    self.email_content_label_2.setReadOnly(False)
    self.email_content_label_2.clear()
    self.email_content_label_2.setStyleSheet(
    ...

    # Výpočet pozice pro vystředění varování
    height = self.email_content_label_2.height()
    total_lines = max(1, height // 32)
    padding = "\n" * (total_lines // 2 - 2)

    # Vložení výstražného textu
    self.email_content_label_2.insertPlainText(padding)
    self.email_content_label_2.insertPlainText(
        getattr(self.text_configuration, f"smail{self.language.
            capitalize()}UnconfirmedEmailWarning")
    )
    self.email_content_label_2.setAlignment(Qt.AlignLeft)

    # Formátování vzhledu varování
    self.email_content_label_2.setStyleSheet(
    ...

    # Naplánování obnovení původního obsahu a tlačítek
    QTimer.singleShot(4000, lambda: self.restore_original_content(
        original_content, default_color))
    QTimer.singleShot(4000, lambda: self.alert_buttons(alert=False)
        )
    QTimer.singleShot(4000, lambda: self.set_buttons_enabled(True))

    # Zakázání tlačítek po dobu výstrahy
    self.sensitive_content_warning_displayed = False
    self.set_buttons_enabled(False)
```

Výpis 5.10: Kód výstrahy při pokusu opustit rozepsaný e-mail

5.3.6 Posílání e-mailu

Odesílání zpráv je klíčovou funkcionalitou aplikace. Tento proces zahrnuje validaci vstupních polí, kontrolu ochranné úrovně, případné upozornění na citlivý obsah a samotné odeslání zprávy pomocí protokolu SMTP. V této sekci je popsán proces zpracování e-mailu od vyplnění údajů až po jeho odeslání prostřednictvím SMTP protokolu.

Zpracování odesílací akce

Po kliknutí na tlačítko s příjemcem se aktivuje funkce `decide_action_for_button`, která rozhoduje, zda je možné přejít k odeslání e-mailu. Pokud uživatel vyplnil zprávu a zvolil příjemce, volá se metoda `send_email_status`. Tato metoda provádí kontrolu zadaných údajů a rozhoduje, zda je možné e-mail odeslat. Průběh lze shrnout následovně:

- Načtení přihlašovacích údajů a nastavení serveru pomocí funkce `load_credentials`.
- Získání textu z jednotlivých vstupních polí: příjemce, předmět a obsah zprávy.
- Pokud je aktivní vyšší úroveň ochrany (protection level ≥ 2), ověřuje se, zda je příjemce ve whitelistu (seznamu schválených kontaktů). Pokud se adresa ve whitelistu nenachází, pole příjemce je zvýrazněno výstražnou barvou a proces odeslání se zastaví. Uživatel by se do této situace neměl dostat, protože ve vyšší úrovni je zakázáno tlačítko „Komu", které slouží k posílání zpráv libovolnému uživateli a jsou předepsány jen osoby ze schválených kontaktů. V ochranné úrovni 1 se tento krok neprovádí.
- Ověření, zda pole nejsou prázdná a zda je e-mailová adresa ve správném formátu (pomocí regexu). Pokud některé z těchto podmínek selže, příslušné pole se zvýrazní výstražnou barvou a odeslání je zablokováno.
- Pokud všechna pole splňují podmínky a neobsahují citlivý obsah, je e-mail odeslán. V závislosti na detekci citlivých dat se volá buď `send_email()`, nebo `send_email_with_guardian_copy()`.
- Na základě návratové hodnoty z těchto funkcí je vyvolána buď metoda `send_email_success()` nebo `send_email_fail()`, které zobrazí příslušnou zprávu o úspěchu nebo selhání.

Validace vstupních polí

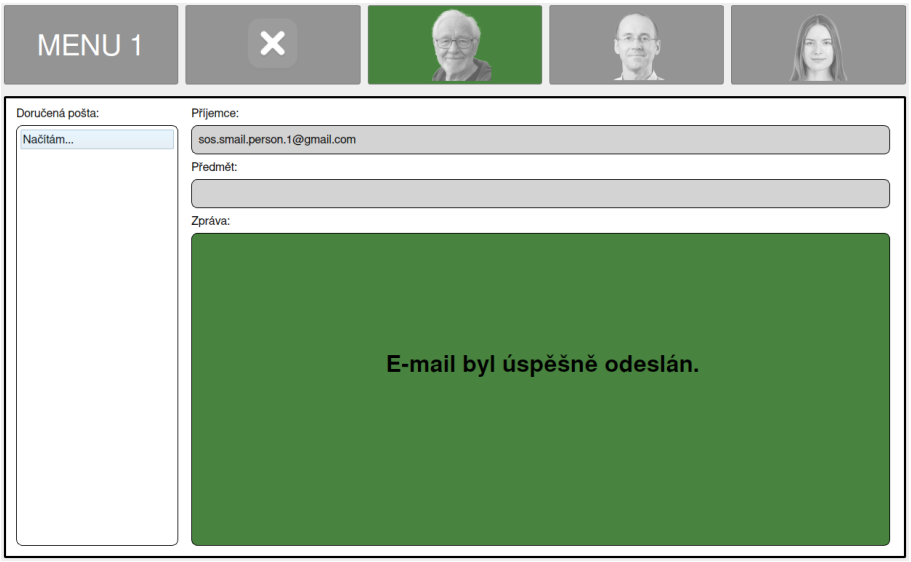
Aplikace ověřuje, zda jsou všechna pole správně vyplněna. V případě, že chybí příjemce nebo text zprávy, je dané pole zvýrazněno výstražnou barvou. To je realizováno pomocí metody `alert_missing_text`, která využívá dočasné přebarvení daného GUI prvku. Při upozornění jsou vstupní pole zároveň dočasně uzamčena, aby uživatel omylem nezasahoval do jejich obsahu během automatického zpracování nebo obnovy stavu.

```
def alert_missing_text(self, entry, default_color, select_color):
    # Kontrola: pole pro předmět zprávy
    if entry == self.recipient_info_label_4:
        # Nastaví výstražnou barvu pozadí, pokud pole není vyplněné
        entry.setStyleSheet(style.get_label_style() + f"background-
            color: {select_color};")
        # Po 2 sekundách obnoví výchozí barvu
        QTimer.singleShot(2000, lambda: self.clear_content_entry(
            default_color, 1))

    # Kontrola: pole pro e-mailovou adresu příjemce
    elif entry == self.recipient_info_label_2:
        # Nastaví výstražnou barvu pozadí, pokud pole není vyplněné
        entry.setStyleSheet(style.get_label_style() + f"background-
            color: {select_color};")
        # Po 2 sekundách obnoví výchozí barvu
        QTimer.singleShot(2000, lambda: self.clear_content_entry(
            default_color, 2))

    # Kontrola: pole pro obsah zprávy
    elif entry == self.email_content_label_2:
        # Nastaví výstražnou barvu pozadí, pokud pole není vyplněné
        entry.setStyleSheet(style.get_email_content_label() + f"
            background-color: {select_color};")
        # Po 2 sekundách obnoví výchozí barvu
        QTimer.singleShot(2000, lambda: self.clear_content_entry(
            default_color, 3))
```

Výpis 5.11: Zvýraznění nevyplněných polí

Úspěšné odeslání e-mailu

Pokud je e-mail úspěšně odeslán, dojde k vyčištění polí a zobrazení potvrzující hlášky ve středu okna. Současně je rozhraní dočasně uzamčeno, aby se předešlo duplicitnímu odeslání. Hláška využívá barevné zvýraznění podle aktivního motivu.



Obr. 5.14: Snímek úspěšného poslání e-mailu

Neúspěšné odeslání e-mailu

V případě chyby při odesílání je uživatel informován výstražnou zprávou. Aplikace zprávu neuloží ani neodešle. GUI se vrátí do výchozího stavu a umožní uživateli znovu vyplnit nebo upravit obsah.

Obr. 5.15: Snímek neúspěšného poslání e-mailu

5.3.7 Detekce citlivých dat

Aplikace obsahuje mechanismus, který automaticky kontroluje obsah e-mailu na přítomnost citlivých údajů. Tento mechanismus funguje ve všech úrovních ochrany (*Protection Level*) a má za cíl zabránit nechtěnému odeslání důvěrných informací. V případě detekce je uživatel upozorněn a zpráva je následně v případě potvrzení odeslána i na e-mail vychovatele.

Citlivý obsah je identifikován dvěma způsoby:

- Porovnáním s předdefinovaným seznamem klíčových slov ve třech jazycích (čeština, němčina, angličtina).
- Pomocí regulárních výrazů, které zachycují specifické formáty jako čísla karet, rodná čísla, IBAN, telefonní čísla apod.

Detekce je zajištěna metodou `contains_sensitive_data`, která vrací hodnotu `True`, pokud je nalezen citlivý obsah.

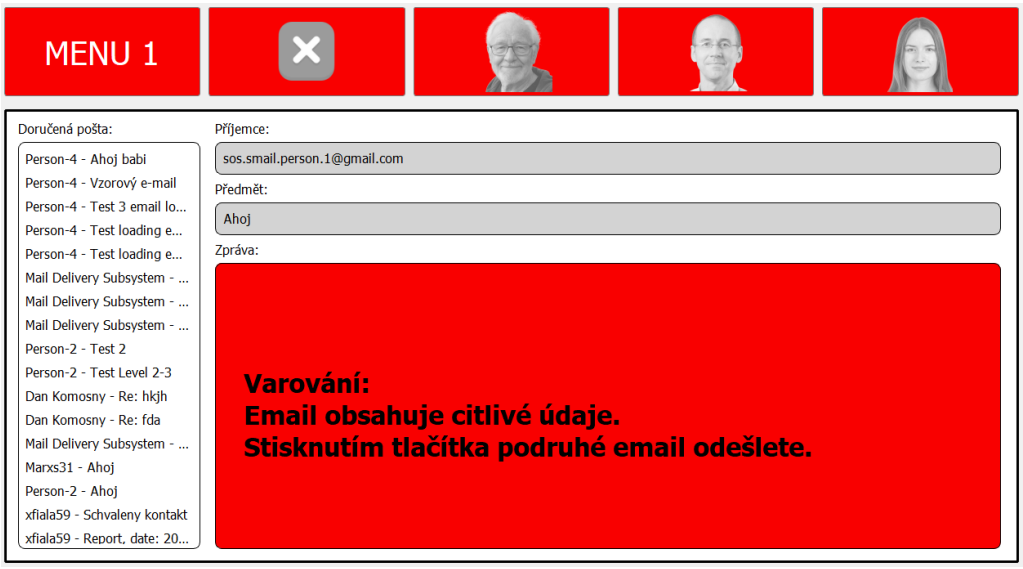
Typ citlivých dat	**Příklady klíčových slov nebo struktur**
Hesla a přihlašovací údaje	heslo, password, login, credentials, geheimes passwort
Bankovní a platební údaje	číslo účtu, card number, IBAN, kreditní karta, routing number
Osobní identifikace	rodné číslo, personal ID, passport, personalausweisnummer
Telefonní a kontaktní údaje	telefonní číslo, email address, adresa bydliště, zip code
Kryptoměny a peněženky	bitcoin, ethereum, crypto wallet, kryptoměna
Daňové identifikační údaje	IČO, DIČ, tax ID, steueridentifikationsnummer
Strukturované údaje (regulární výrazy - regex)	čísla kreditních karet, rodná čísla, IBAN, čísla pasů

Tab. 5.2: Vzorek citlivých údajů detekovaných aplikací

Zobrazení varování

Kontrola citlivého obsahu probíhá při stisku tlačítka pro odeslání zprávy, konkrétně ve funkci `send_email_status`, která je volána z metody `decide_action_for_button`. Tím je zajištěno, že se obsah zprávy zkontroluje vždy těsně před samotným odesláním. Při detekci citlivého obsahu se zobrazí výstražná zpráva přímo v poli s obsahem

e-mailu. Odesílání zprávy je dočasně zablokováno. Uživatel může zprávu odeslat až po opětovném potvrzení akce.



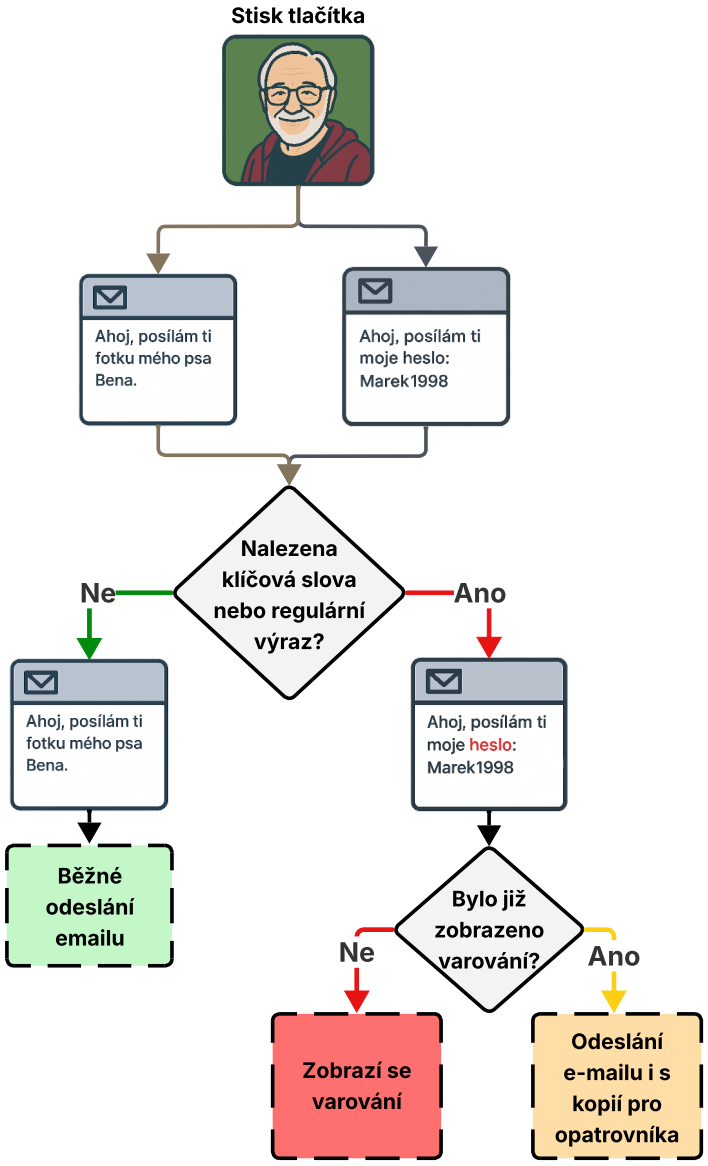
Obr. 5.16: Snímek varující na obsah citlivých dat

Odeslání kopie vychovateli

Pokud je obsah zprávy označen jako citlivý a uživatel i přes zobrazené varování potvrdí jeho odeslání, systém automaticky odešle e-mail pomocí metody `send_email_with_guardian_copy`, která přidá jako skrytou kopii adresu vychovatele. Tím je zajištěn dodatečný dohled nad komunikací.

```
if self.contains_sensitive_data(content):
    if not self.sensitive_content_warning_displayed: # Pokud ještě
        nebylo zobrazeno varování, zobrazí se
        self.alert_sensitive_data()
        self.sensitive_content_warning_displayed = True
        return
    else: # Pokud uživatel varování potvrdil, odešle se e-mail i
        vychovateli
        success = send_email_with_guardian_copy(
            recipient, subject, content, login, password,
                smtp_server, smtp_port, self.guardian_email)
else:
    # Pokud zpráva neobsahuje citlivý obsah, odešle se běžně
    success = send_email(recipient, subject, content, login,
        password, smtp_server, smtp_port)
```

Výpis 5.12: Rozhodovací logika pro odeslání e-mailu



Obr. 5.17: Diagram detekce citlivých dat při odesílání e-mailu

Tento diagram znázorňuje proces kontroly obsahu e-mailu na citlivá data před jeho odesláním. Po stisku tlačítka odeslat je nejprve zkontrolováno, zda zpráva obsahuje klíčová slova nebo vzory odpovídající citlivým údajům (například hesla, čísla platebních karet, rodná čísla apod.).

Pokud není nalezen žádný problém, e-mail je běžně odeslán. Pokud citlivý obsah nalezen je, aplikace ověří, zda již bylo uživateli zobrazeno varování. Pokud ne, zobrazí se výstraha a uživatel musí potvrdit, že si je obsahu vědom. Pokud už bylo varování zobrazeno a potvrzeno, e-mail je odeslán společně s kopií vychovateli.

5.3.8 Úrovně omezení uživatele z důvodu bezpečnosti

V aplikaci je implementovaný víceúrovňový systém ochrany nazývaný *Protection Level* (PL), který zajišťuje zvýšenou bezpečnost při práci s e-maily. Úroveň ochrany je nastavena v konfiguračním souboru a ovlivňuje, jakým způsobem může uživatel pracovat s příchozími i odchozími zprávami.

- **Úroveň ochrany 1 (PL1):** Uživatel má možnost zasílat e-maily na libovolnou adresu. Při detekci citlivých údajů (hesla, bankovní údaje, osobní doklady) je však zobrazena výstraha a při odeslání je kopie zprávy automaticky odeslána i na předdefinovaný e-mail vychovatele.
- **Úroveň ochrany 2 (PL2):** Funkce tlačítka *Komu* je deaktivována, protože pouze schválené kontakty uvedené v konfiguračním souboru mohou přijímat zprávy. Stejně tak jsou blokovány příchozí e-maily od neschválených odesílatelů, aby se minimalizovalo riziko podvodné komunikace.

Obr. 5.18: Uzamčené tlačítko v úrovni ochrany 2

Kontrola příjemce při odesílání e-mailu

Před odesláním zprávy aplikace ověřuje, zda je adresa příjemce uvedena ve schváleném seznamu kontaktů. Pokud není a ochrana je nastavena na PL2 nebo vyšší, aplikace zabrání odeslání zprávy a upozorní uživatele.

```
if self.protection_level >= 2 and recipient.lower() not in
    approved_contacts:
    self.alert_missing_text(self.recipient_info_label_2,
        default_color, alert_color)
```

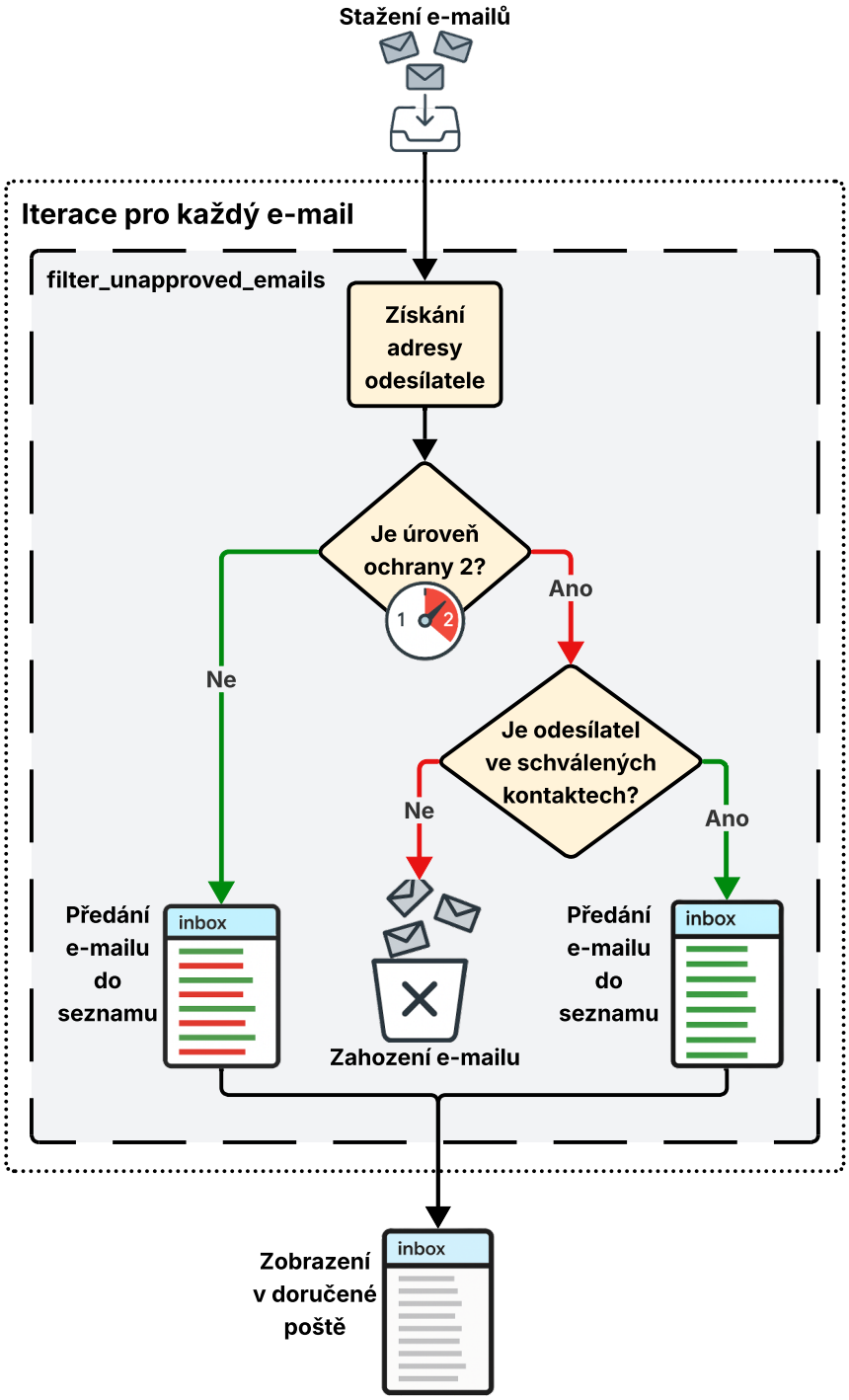
Výpis 5.13: Kontrola příjemce před odesláním e-mailu

Filtrování příchozích e-mailů

Při načítání příchozích zpráv aplikace filtruje e-maily od neschválených kontaktů. Pouze e-maily od známých odesílatelů se zobrazí v seznamu doručených zpráv.

```
if protection_level >= 2 and sender_email not in approved_contacts:
    continue  # Email od neznámého odesílatele je ignorován
```

Výpis 5.14: Filtrování příchozích e-mailů podle ochrany PL2



Obr. 5.19: Diagram zobrazující proces filtrování příchozích e-mailů

Diagram znázorňuje proces filtrování příchozích e-mailů na základě nastavené úrovně ochrany. Pro každý načtený e-mail se zjistí adresa odesílatele. Pokud je aktivní vyšší ochrana (Protection Level 2), aplikace ověří, zda se odesílatel nachází ve schváleném seznamu kontaktů. Pokud ano, e-mail je přijat a zobrazen v doručené poště; pokud ne, je e-mail automaticky zahozen a nezobrazí se uživateli.

6 Návody k použití

Tato kapitola se zaměřuje na uživatelské manuály pro aplikaci SMAIL, které jsou určeny především pro seniory, aby jim usnadnily používání této aplikace. Manuály jsou navrženy tak, aby byly co nejjednodušší a snadno pochopitelné, s důrazem na přehlednost a srozumitelnost. Jsou dostupné ve třech jazycích: češtině, angličtině a němčině. Tyto návody jsou uloženy online na GitHub projektu.

Manuály poskytují detailní informace o všech klíčových funkcích aplikace SMAIL, včetně správy doručených e-mailů, psaní a odesílání nových zpráv. Jsou navrženy tak, aby vedly uživatele krok za krokem a minimalizovaly nutnost samostatného objevování aplikace. Jedná se o podrobný průvodce, který by měl i méně zkušeným uživatelům pomoci zvládnout ovládání aplikace a využívat všechny její hlavní funkce bez složitostí.

Návody byly vytvořeny s ohledem na specifické potřeby seniorů, přičemž důraz je kladen na:

- **Jednoduchý a srozumitelný jazyk** – všechny pokyny jsou prezentovány ve snadno pochopitelném jazyce bez zbytečných technických výrazů.
- **Velká písmena a strukturovaný obsah** – manuály používají větší písma pro lepší čitelnost a strukturovaný formát, který usnadňuje orientaci v dokumentu.
- **Vizualizace kroků** – manuály zahrnují obrázky, které ukazují konkrétní části uživatelského rozhraní a procesy spojené s používáním aplikace.
- **Bezpečnostní tipy** – jednotlivé kroky zahrnují také rady, jak zůstat při práci s aplikací v bezpečí, například upozornění na citlivý obsah, jako jsou hesla nebo údaje o platební kartě.

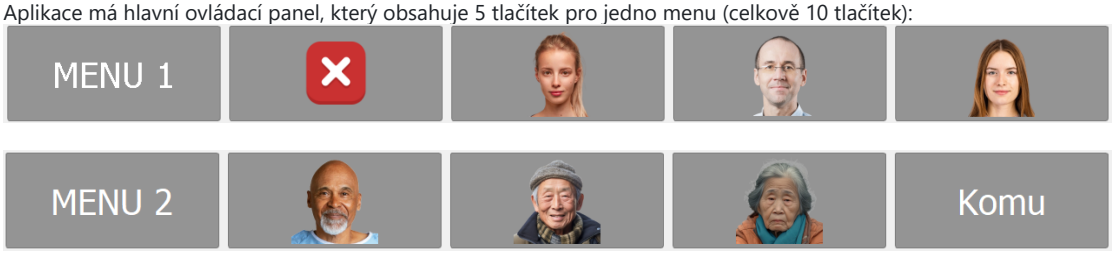
Manuály jsou dostupné ve formátu PDF (Portable Document Format) a Markdown (.md) a lze je snadno stáhnout nebo vytisknout. Pro ty, kteří preferují přístup k návodům online, jsou tyto dokumenty k dispozici na oficiální webové stránce projektu.

V následujících stránkách jsou zobrazeny manuály k použití aplikace SMAIL ve třech jazycích:

Návod na použití aplikace SMAIL

Ovládání aplikace:

Aplikace má hlavní ovládací panel, který obsahuje 5 tlačítek pro jedno menu (celkově 10 tlačítek):



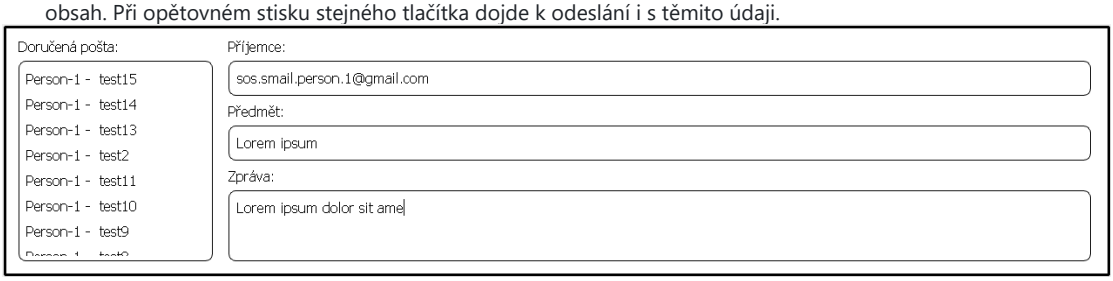
1. **Menu** – umožňuje přepínat mezi dvěma různými částmi aplikace (Menu 1 a Menu 2).
2. **Červené tlačítko s bílým křížkem** – slouží k ukončení aplikace.
3. **Tlačítka s ikonami osob** – slouží k napsání a odeslání zprávy konkrétní osobě. Prvním stisknutím tlačítka otevřete prostředí pro psaní zprávy dané osobě. Opětovným stisknutím tlačítka se zpráva odešle.
4. **Tlačítko "Komu"** – funguje podobně jako tlačítka s osobami, ale je určeno pro zadání libovolné e-mailové adresy, pokud není konkrétní osoba vybrána. Toto tlačítko může být z důvodu bezpečnosti vypnuto.

Zobrazení e-mailů:

Kliknutím na libovolný e-mail v seznamu přijatých zpráv (Doručená pošta) se vybraná zpráva zobrazí v okně **Informace a Zpráva** v pravé části aplikace

Psaní e-mailu:

1. Pro napsání e-mailu vyberte **Tlačítko s osobou**, které chcete napsat, nebo použijte tlačítko **Komu**, kde zadáte e-mailovou adresu příjemce do pole **Příjemce**.
2. Do pole **Předmět** můžete napsat stručné téma zprávy (není povinné, může zůstat prázdné).
3. Do pole **Zpráva** napište text vaší zprávy.
4. K odeslání stačí podruhé stisknout vybrané **tlačítko s osobou** nebo tlačítko **Komu**.
5. Při úspěšném odeslání se zobrazí zpráva **„Email byl úspěšně odeslán"** na zeleném podkladu.
6. Při pokusu opustit rozepsanou zprávu se zobrazí upozornění, že její opuštěním dojde ke ztrátě obsahu, pokud uživatel znovu stiskne jiné tlačítko.
7. Při pokusu odeslání zprávy obsahující citlivý obsah (hesla, číslo karty) se zobrazí zpráva upozurňující na tento obsah. Při opětovném stisku stejného tlačítka dojde k odeslání i s těmito údaji.

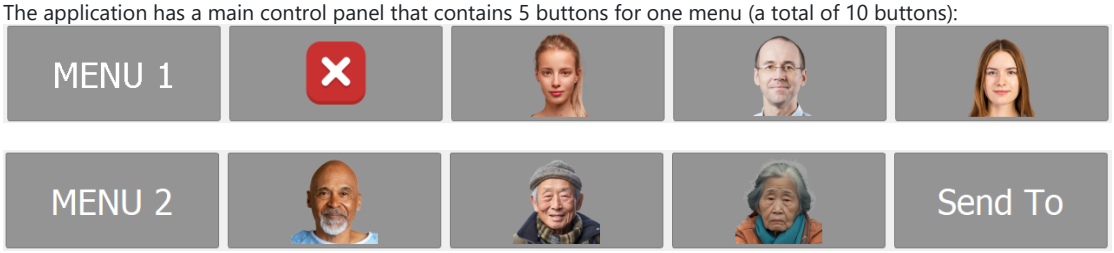


Obr. 6.1: Návod k použití v češtině

User Guide for SMAIL Application

Application Control:

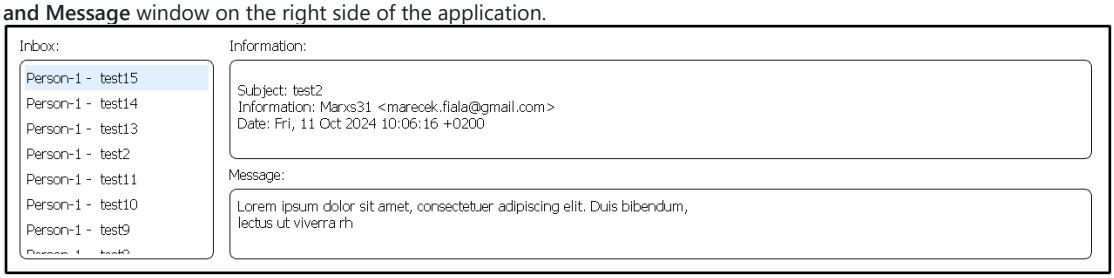
The application has a main control panel that contains 5 buttons for one menu (a total of 10 buttons):



1. **Menu** – allows switching between two different parts of the application (Menu 1 and Menu 2).
2. **Red button with a white cross** – is used to exit the application.
3. **Buttons with person icons** – are used to write and send messages to specific people. Pressing the button once opens the message writing environment for that person. Pressing the button again sends the message.
4. **"To" button** – works similarly to the person buttons but is intended for entering any email address if no specific person is selected. This button may be disabled for security reasons.

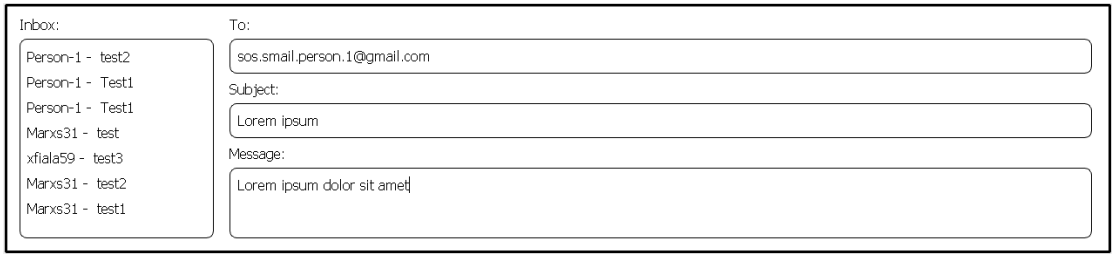
Viewing Emails:

Clicking on any email in the list of received messages (Inbox) will display the selected message in the **Information and Message** window on the right side of the application.



Writing an Email:

1. To write an email, select the **Person Button** you want to write to or use the **To** button, where you can enter the recipient's email address in the **Recipient** field.
2. In the **Subject** field, you can write a brief topic for the message.
3. Write your message in the **Message** field.
4. To send, press the selected **person button** or the **To** button again (marked in green).
5. Upon successful sending, the message **"Email has been sent successfully"** will be displayed on a green background.
6. If you try to leave a message while writing it, a warning will appear stating that leaving the message will result in losing its content if you press another button again.
7. If you attempt to send a message containing sensitive information (e.g., passwords or card numbers), a warning message will be displayed. Pressing the **person button** or the **To** button again will send the message, including the sensitive information.

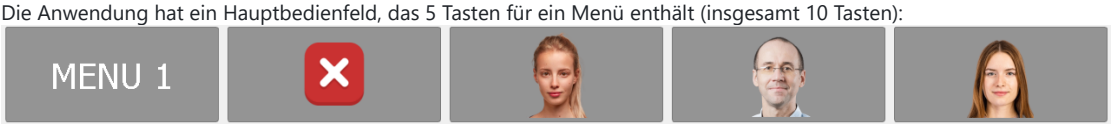


Obr. 6.2: Návod k použití v angličtině

Benutzerhandbuch für die SMAIL-Anwendung

Steuerung der Anwendung:

Die Anwendung hat ein Hauptbedienfeld, das 5 Tasten für ein Menü enthält (insgesamt 10 Tasten):



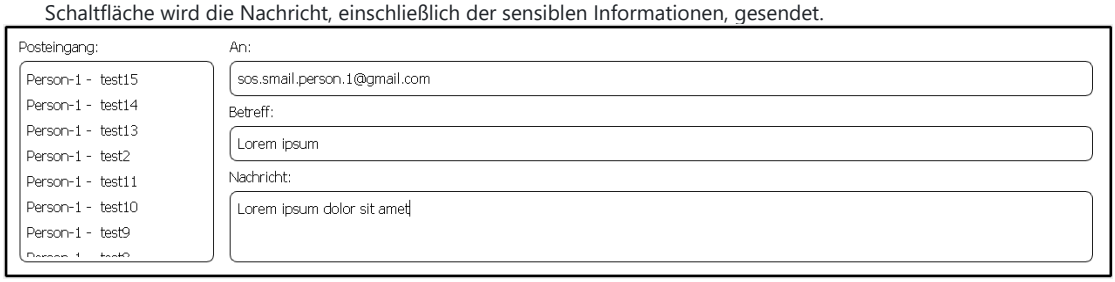
1. **Menü** – ermöglicht das Umschalten zwischen zwei verschiedenen Teilen der Anwendung (Menü 1 und Menü 2).
2. **Rote Schaltfläche mit weißem Kreuz** – dient zum Beenden der Anwendung.
3. **Schaltflächen mit Personen-Icons** – dienen zum Schreiben und Senden von Nachrichten an bestimmte Personen. Durch einmaliges Drücken der Schaltfläche wird die Nachrichtenumgebung für die jeweilige Person geöffnet. Durch erneutes Drücken wird die Nachricht gesendet.
4. **"An"-Schaltfläche** – funktioniert ähnlich wie die Personenschaltflächen, ist jedoch zum Eingeben einer beliebigen E-Mail-Adresse vorgesehen, wenn keine bestimmte Person ausgewählt ist. Diese Schaltfläche kann aus Sicherheitsgründen deaktiviert sein.

Anzeigen von E-Mails:

Durch Klicken auf eine beliebige E-Mail in der Liste der empfangenen Nachrichten (Posteingang) wird die ausgewählte Nachricht im Fenster **Information und Nachricht** auf der rechten Seite der Anwendung angezeigt.

Schreiben einer E-Mail:

1. Um eine E-Mail zu schreiben, wählen Sie die **Personen-Schaltfläche**, an die Sie schreiben möchten, oder verwenden Sie die **An**-Schaltfläche, um die E-Mail-Adresse des Empfängers im **Empfänger**-Feld einzugeben.
2. In das Feld **Betreff** können Sie ein kurzes Thema für die Nachricht eingeben.
3. Schreiben Sie Ihre Nachricht in das **Nachricht**-Feld.
4. Zum Senden drücken Sie erneut die ausgewählte **Personen-Schaltfläche** oder die **An**-Schaltfläche (grün markiert).
5. Nach erfolgreichem Senden wird die Meldung **„E-Mail wurde erfolgreich gesendet"** auf grünem Hintergrund angezeigt.
6. Beim Versuch, eine begonnene Nachricht zu verlassen, erscheint eine Warnmeldung, dass der Inhalt verloren geht, wenn erneut eine andere Schaltfläche gedrückt wird.
7. Wenn Sie versuchen, eine Nachricht mit sensiblen Informationen (z. B. Passwörtern oder Kartennummern) zu senden, wird eine Warnmeldung angezeigt. Durch erneutes Drücken der **Personen-Schaltfläche** oder der **An**-Schaltfläche wird die Nachricht, einschließlich der sensiblen Informationen, gesendet.



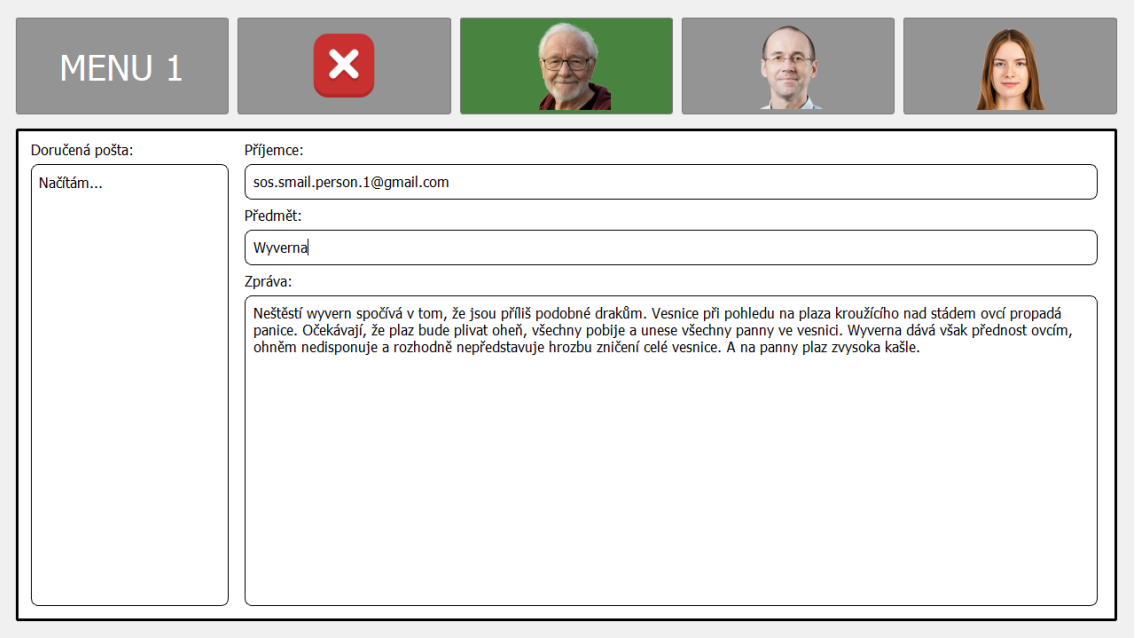
Obr. 6.3: Návod k použití v němčině

7 Výzvy při řešení práce a výsledky

Při vývoji aplikace SMAIL bylo nutné čelit několika technickým a návrhovým výzvám, jejichž překonání významně přispělo k dosažení kvalitního výsledku. V této kapitole jsou shrnuty hlavní problémy, které se během implementace objevily, a způsoby jejich řešení.

Přechod z `Tkinter` na `PyQt5`

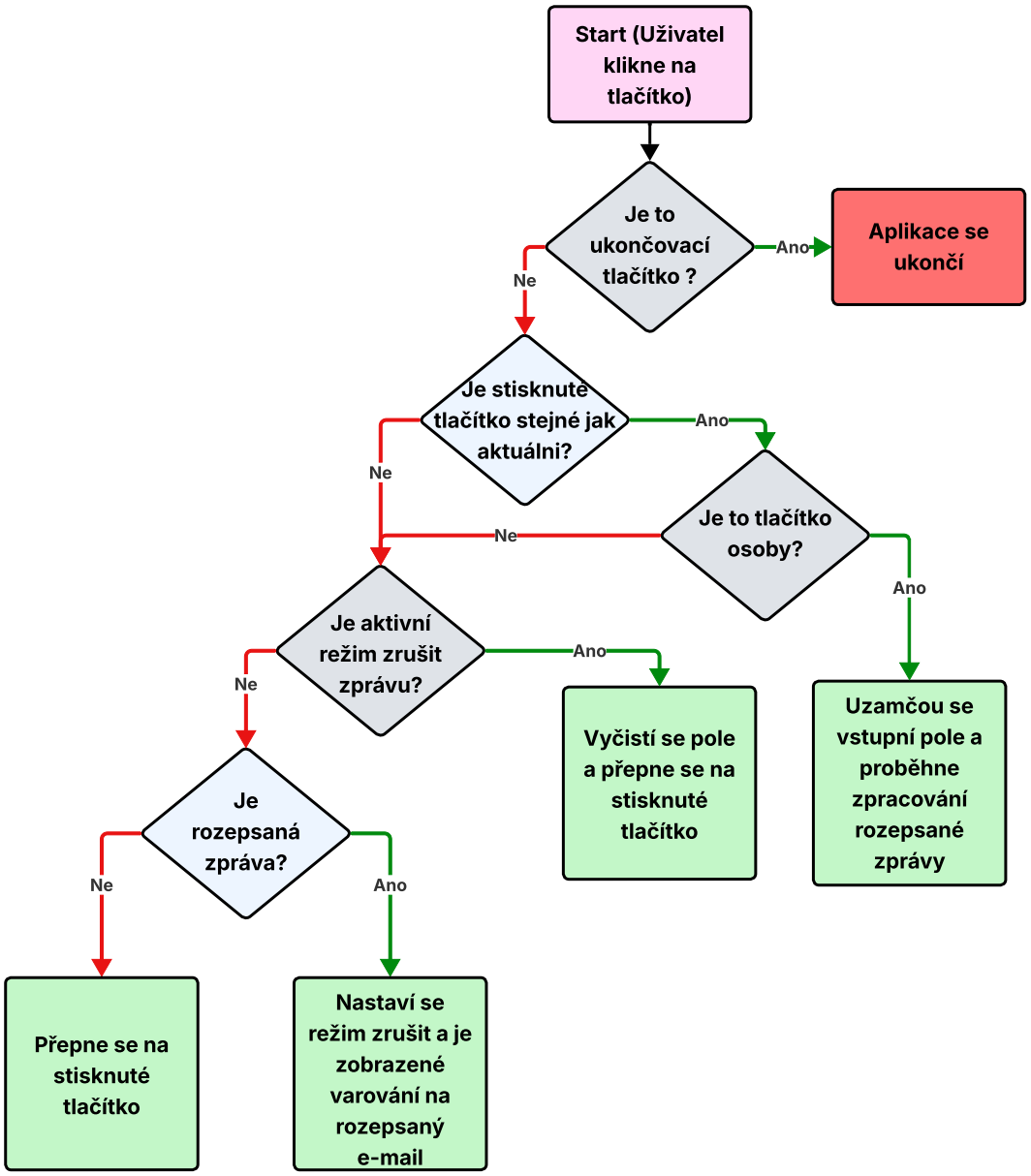
Jednou z klíčových výzev bylo přepracování celé aplikace z původního GUI frameworku `Tkinter` na modernější `PyQt5`. Tento krok si vyžádal kompletní přepsání grafického rozhraní od základů. Všechny původní funkce závislé na `Tkinteru` musely být buď nově implementovány, upraveny, nebo zcela odstraněny. Přechodem se změnil způsob práce se styly, událostmi i strukturou rozhraní, což vedlo ke zvýšení udržovatelnosti kódu a modernizaci vzhledu aplikace.



Obr. 7.1: Ukázka grafického prostředí aplikace

Správná logika tlačítek a uživatelské interakce

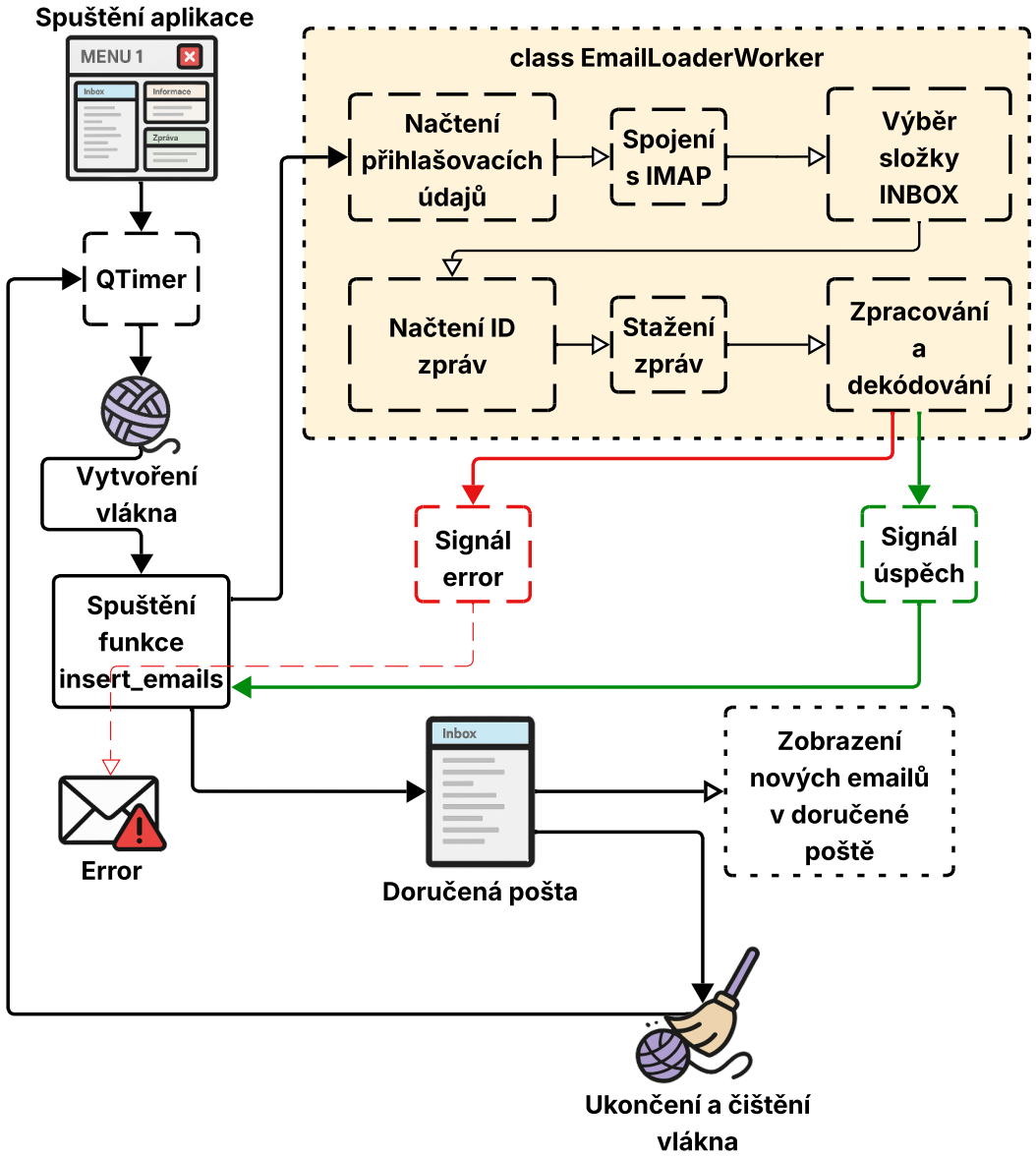
Velkou výzvou bylo zajištění správného chování tlačítek v rámci uživatelského rozhraní. Bylo nutné ošetřit situace, kdy by mohla uživatelská akce neúmyslně narušit právě probíhající operaci (například opuštění rozepsaného e-mailu nebo výběr příjemce). Proto bylo implementováno dynamické deaktivování ostatních tlačítek během kritických operací, čímž se zabránilo nechtěnému přepsání nebo zablokování stavu aplikace.



Obr. 7.2: Ukázka rozhodovací logiky tlačítek

Periodické načítání e-mailů bez blokování GUI

Další technickou komplikací bylo zajištění periodického načítání nových e-mailů bez zamrznutí hlavního grafického rozhraní. Původně bylo načítání realizováno v hlavním vlákně aplikace, což vedlo k výraznému zpomalování a neodpovídání GUI. Tento problém byl vyřešen přesunutím načítací logiky do samostatného vlákna pomocí třídy `QThread`, čímž byla zachována plynulost práce uživatele i při načítání nových zpráv.



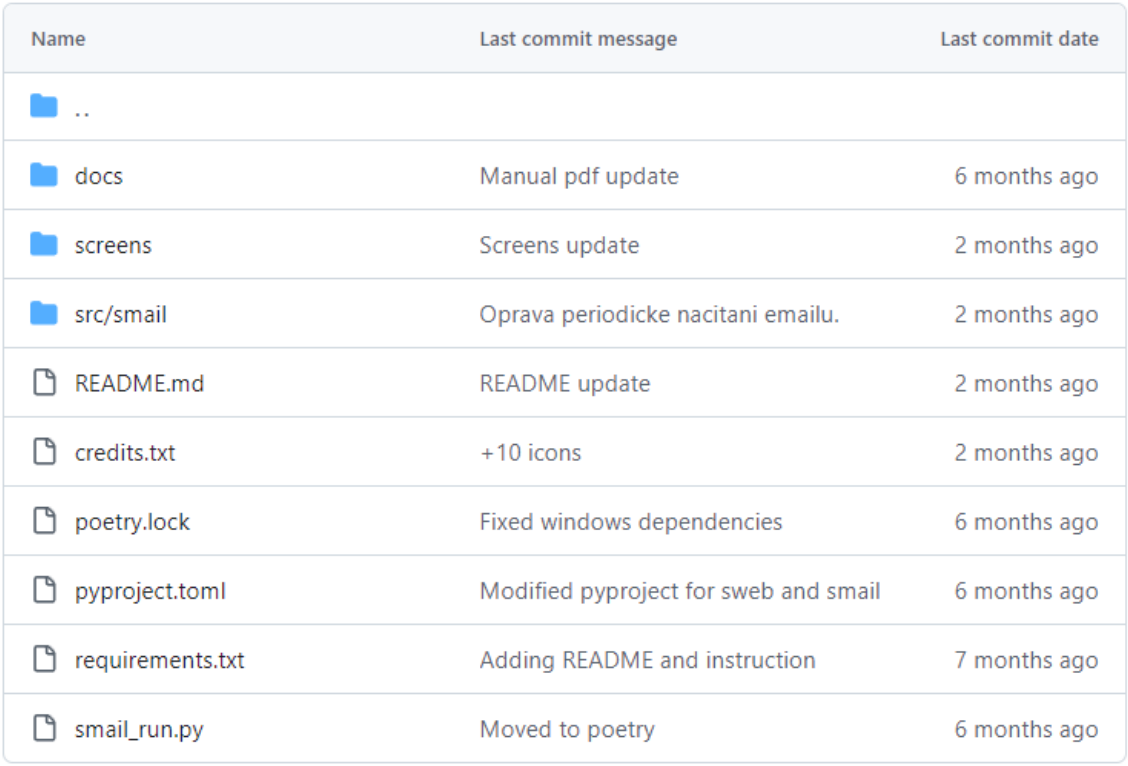
Obr. 7.3: Ukázka načítání zpráv

Odstranění cyklických závislostí v kódu

Při refaktorizaci kódu byly odhaleny skryté cyklické závislosti mezi moduly, zejména v části zajišťující komunikaci se serverem. Tyto závislosti mohly způsobovat neočekávané chyby při běhu aplikace. Problém byl odstraněn restrukturalizací importů a důsledným oddělením odpovědností jednotlivých modulů. Díky odstranění těchto překážek byla vytvořena aplikace, která je nejen funkční, ale také bezpečná, přístupná a přívětivá pro cílovou skupinu uživatelů.

8 Distribuce aplikace

Tato kapitola se zabývá způsobem správy, sdílení a zpřístupnění projektu SMAIL. Aplikace je distribuována prostřednictvím repozitáře na platformě GitHub na adrese: github.com/forsenior/senior-os a obsahuje veškeré potřebné komponenty, které jsou nezbytné pro správné stažení, instalaci a konfiguraci aplikace. Na obrázku 8.1 je znázorněna struktura repozitáře projektu SMAIL.



Name	Last commit message	Last commit date
..		
docs	Manual pdf update	6 months ago
screens	Screens update	2 months ago
src/smail	Oprava periodicke nacitani emailu.	2 months ago
README.md	README update	2 months ago
credits.txt	+10 icons	2 months ago
poetry.lock	Fixed windows dependencies	6 months ago
pyproject.toml	Modified pyproject for sweb and smail	6 months ago
requirements.txt	Adding README and instruction	7 months ago
smail_run.py	Moved to poetry	6 months ago

Obr. 8.1: Struktura repozitáře GitHub projektu SMAIL

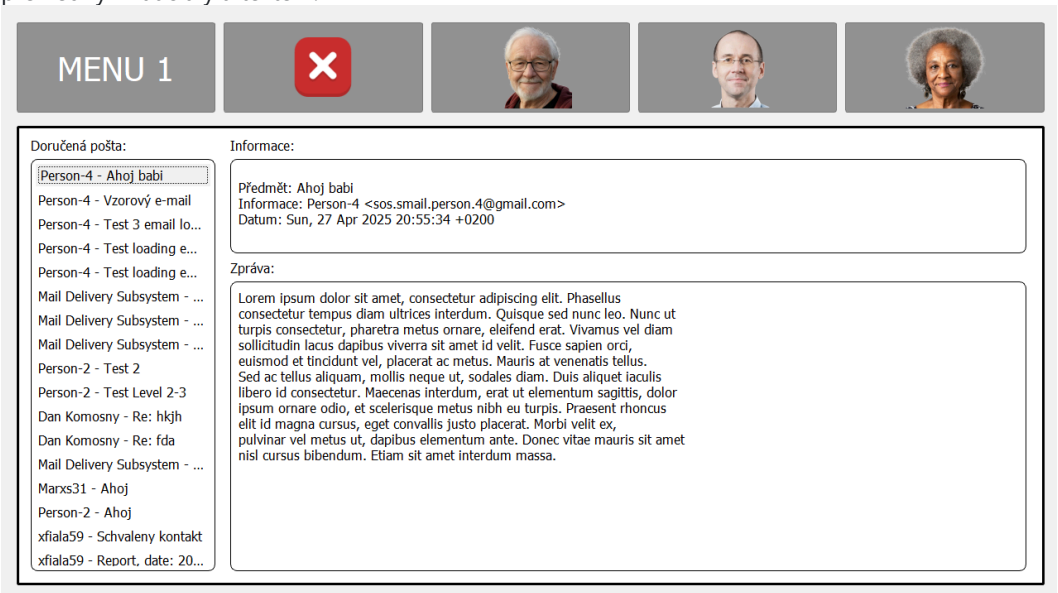
- **docs:** Obsahuje manuály pro použití aplikace v PDF formátu ve třech jazycích: češtině, angličtině a němčině.
- **screens:** Snímky obrazovky jednotlivých částí aplikace. Tyto obrázky slouží pro ilustraci prostředí a chování uživatelského rozhraní.
- **src:** Zdrojové kódy napsané v jazyce Python. Složka obsahuje hlavní logiku aplikace včetně odesílání a přijímání e-mailů, GUI a bezpečnostních kontrol.
- **README.md:** Textový soubor s přehledným shrnutím všech důležitých informací o aplikaci. Obsahuje návod na instalaci, spuštění a konfiguraci aplikace, a je považován za úvodní bod pro nové uživatele i vývojáře. Pro snadnější pochopení a přístupnost byl obsah README souboru přeložen do češtiny a rozdělen na více částí z důvodu rozsahu dokumentu.

SMAIL – E-mailový klient pro seniory

Tento e-mailový klient je přizpůsoben pro seniory ve věkové skupině 90 let a více.
Vyvinutý klient je snadno ovladatelný a obsahuje pouze funkce, které může senior potřebovat.

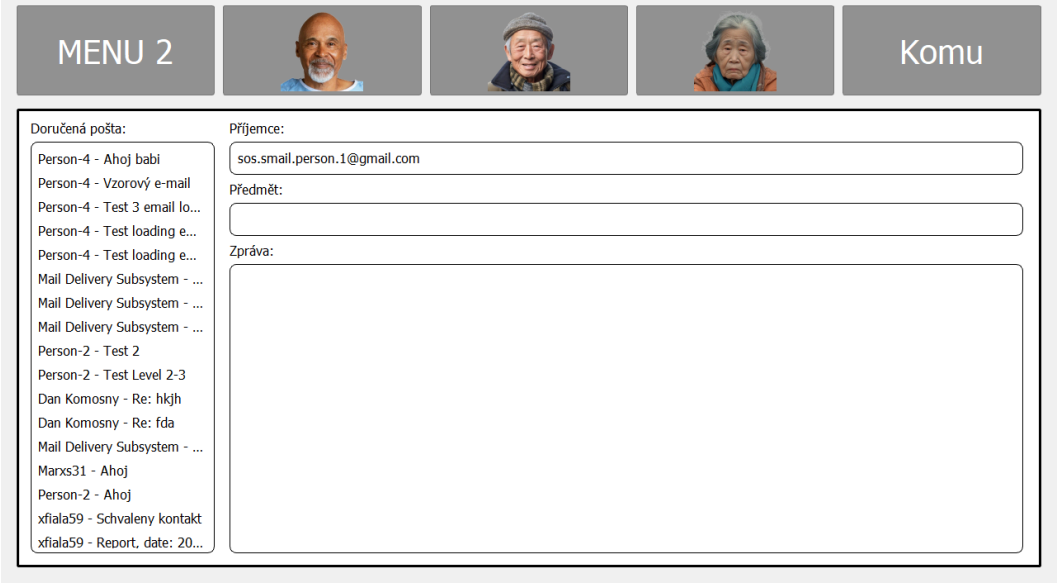
Prostředí pro čtení e-mailových zpráv

SMAIL má prostředí speciálně navržené pro čtení e-mailů přívětivou formou, s velkými a přehlednými tlačítky a textem.



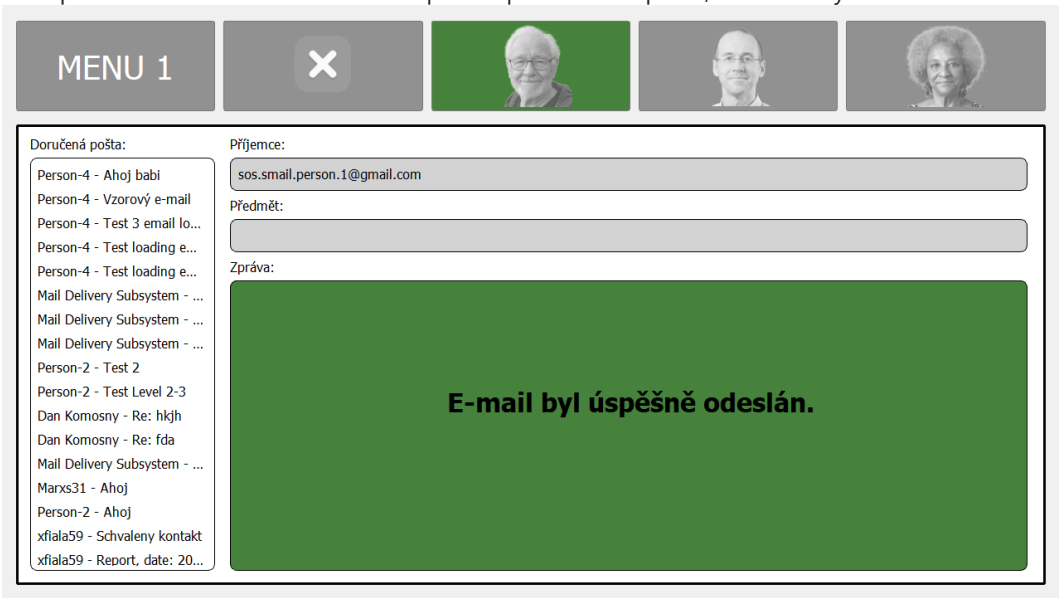
Prostředí pro psaní e-mailových zpráv

Prostředí pro psaní e-mailů je co nejjednodušší, s předdefinovanými kontakty pro snadné odesílání.



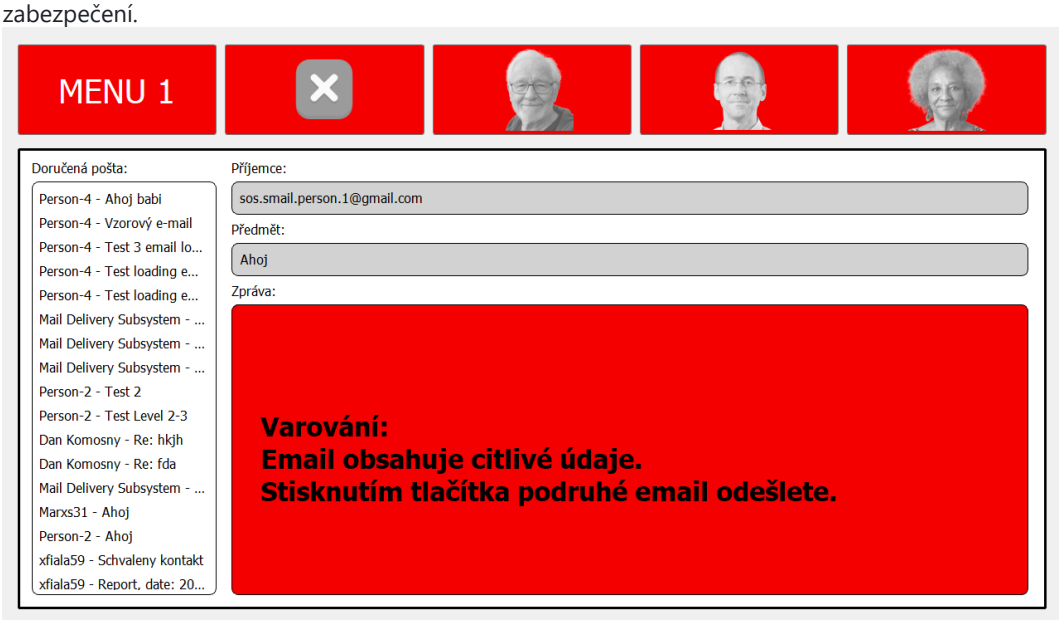
Potvrzení úspěšného odeslání e-mailu

Po úspěšném odeslání e-mailu zobrazí aplikace potvrzovací zprávu, že e-mail byl odeslán.



Varování při odesílání citlivých údajů

Aplikace upozorní uživatele, když se chystá odeslat citlivé informace, čímž přidává další vrstvu zabezpečení.



Varování při opuštění neuloženého e-mailu

Aplikace varuje uživatele při pokusu o opuštění rozepsané zprávy. Uživatel je vyzván k potvrzení, zda si přeje e-mail zrušit.

1) Spuštění aplikace – použijte následující postup:

Stáhněte si nejnovější ISO z repozitáře.
Vytvořte nový virtuální stroj ve vámi zvoleném virtualizačním prostředí (např. VirtualBox, VMware nebo QEMU) a přidejte stažené ISO.
Toto ISO je distribuováno jako systém založený na Linuxu.
Spusťte virtuální prostředí a pokud se aplikace nespustí automaticky, spusťte následující příkaz v terminálu:

```
srun
```

2) Pro snadnější úpravy a přehled o aplikaci postupujte následovně:

Instalace

Pro spuštění SMAIL postupujte podle následujících kroků pro naklonování repozitáře a instalaci závislostí:

Krok 1: Instalace Poetry

SMAIL používá Poetry pro správu závislostí a balení.
Pokud nemáte Poetry nainstalované, nejprve jej nainstalujte.
Můžete postupovat podle tohoto podrobného návodu k instalaci Poetry.

Krok 2: Klonování repozitáře a instalace závislostí

Po instalaci Poetry pokračujte těmito kroky:

```
# Klonování repozitáře
git clone https://github.com/forsenior/senior-os

# Přejděte do složky projektu
cd smail

# Sestavení a instalace závislostí pomocí Poetry
poetry build

poetry install
```

Tyto kroky opakujte i pro složky *sconf* a *srun* (cd .. -> cd sconf -> poetry build -> poetry install):

```
#Spusťte tento příkaz ve složce se stejným názvem:
poetry run srun
#Nebo můžete použít tento příkaz po nastavení konfiguračního souboru:
poetry run smail
```

Podporované verze Pythonu: Tento program je testován a optimalizován pro Python 3.12.

Požadovaná konfigurace pro SMAIL

Pro správné fungování e-mailového klienta SMAIL se doporučuje nejprve upravit konfigurační soubory.
Pro dočasnou konfiguraci můžete upravit hlavní konfigurační soubor config.json, který se nachází v domovském adresáři (např. C:\Users\user.sconf\config.json).
Pokud není přítomen, zkopírujte nebo vytvořte soubor config.json (soubor se automaticky vytvoří po spuštění *srun*).
Pro trvalé nastavení, které bude fungovat kontinuálně, je třeba upravit soubor `smail_configuration.py` , který se nachází v `sconf/src/sconf/models/` .
Během tohoto procesu zadejte svou vlastní e-mailovou adresu a heslo pro personalizaci aplikace.
Výchozí nastavení jsou k dispozici, ale pro lepší zabezpečení a přizpůsobení se doporučuje použít vlastní údaje.

Generování hesla pro SMAIL

Pro připojení ke svému Gmail účtu z newebového prostředí, jako je aplikace SMAIL, je potřeba vygenerovat heslo pro aplikaci.
Postupujte následovně:

1. Přejděte do nastavení účtu Google na stránce *Google účet*.
2. Otevřete záložku „Zabezpečení“.
3. V části „Přihlášení do Googlu“ vyhledejte a vyberte „Dvoufázové ověření“.
4. Po ověření identity přejděte dolů do části „Hesla pro aplikace“.
5. Vyberte možnost pro vytvoření nového hesla pro aplikaci.
6. Zvolte aplikaci nebo zařízení, pro které heslo vytváříte.
7. Postupujte podle pokynů pro vygenerování hesla.
8. Zkopírujte vygenerované heslo a použijte ho v aplikaci SGIVE pro bezpečné připojení e-mailového klienta k vašemu Gmail účtu.

Pokud nemůžete najít správnou sekci, použijte tento odkaz: Hesla pro aplikace

Spuštění aplikace SMAIL

Pro spuštění aplikace postupujte následovně:

1. **Otevřete své preferované IDE**
   Spusťte vámi zvolené vývojové prostředí, například PyCharm nebo jiné.
2. **Přejděte do adresáře smail**
   Pomocí správce souborů v IDE otevřete složku `smail` v rámci naklonovaného repozitáře.
3. **Otevřete terminál v této složce**
   Ujistěte se, že máte aktivní terminálovou relaci ve složce `smail` .
4. **Ujistěte se, že jsou nainstalovány všechny potřebné závislosti**
   Ověřte, že jsou všechny závislosti správně nainstalovány. Můžete postupovat dle sekce Instalace.
5. **Nastavte konfiguraci a heslo pro aplikaci**
   Před spuštěním aplikace nastavte konfiguraci a vygenerujte heslo dle sekce Generování hesla.
6. **Spusťte aplikaci**
   Jakmile je vše připraveno, spusťte aplikaci tímto příkazem:

```
cd smail
poetry run smail
```

(5/5)

Obr. 8.2: Obsah souboru README v češtině

Závěr

Cílem této diplomové práce bylo vytvořit základního e-mailového klienta přizpůsobeného potřebám seniorů vyššího věku a mentálně znevýhodněných uživatelů. Aplikace byla navržena tak, aby minimalizovala složitost ovládání, nabídla intuitivní a přehledné rozhraní a současně zahrnovala klíčové bezpečnostní funkce pro ochranu uživatele.

V teoretické části práce byly shrnuty základní principy fungování e-mailových systémů, struktura e-mailových zpráv a přenosové protokoly (SMTP, IMAP, POP3, MIME). Dále byla věnována pozornost otázkám bezpečnosti e-mailové komunikace, včetně nejčastějších hrozeb spojených s e-mailem.

Praktická část se zaměřila na implementaci e-mailového klienta pomocí programovacího jazyka Python a knihovny PyQt5. V rámci implementace byly kladeny důrazy na jednoduchost ovládání, přístupnost, přehlednost a bezpečnost.

Jednou z nejdůležitějších změn byla úprava GUI rozhraní aplikace. Došlo k přechodu z knihovny `tkinter` na modernější a uživatelsky přívětivější knihovnu `PyQt5`, čímž se sjednotil vizuální styl mezi jednotlivými aplikacemi v rámci integrovaného systému a zjednodušila se jejich údržba.

Do aplikace byla přidána funkcionalita, která upozorňuje na odesílání citlivých dat. V případě detekce údajů, jako jsou hesla nebo čísla karet, je uživatel varován a odeslání zprávy je možné až po potvrzení uživatelem. Tím se snižuje riziko sdílení citlivých informací.

Zavedena byla také ochrana proti nechtěnému smazání rozepsané zprávy. Uživatel je při opuštění rozpracovaného e-mailu výstražně upozorněn a musí své rozhodnutí potvrdit.

V rámci zvýšení bezpečnosti byla zavedena tzv. úroveň ochrany (Protection Level). Uživatelé s nižší úrovní mohou posílat zprávy komukoli, ale při detekci citlivých dat je kopie zprávy odeslána i vychovateli. U vyšší úrovně je umožněno komunikovat pouze se schválenými kontakty, a zprávy od neznámých odesílatelů se uživateli vůbec nezobrazují.

Součástí vylepšení byla také kontrola správného formátu e-mailové adresy. Díky tomu je možné předejít chybám při odesílání zpráv na neexistující adresy a zvyšuje se tak spolehlivost doručování.

V rámci refaktorizace kódu byla odstraněna skrytá cyklická závislost, která mohla způsobovat chyby při navazování spojení. Struktura zdrojového kódu byla zpřehledněna, a práce s konfiguračními daty byla zjednodušena díky využití datového poskytovatele.

Napříč celým projektem bylo sjednoceno pojmenování souborů a složek, což usnadnilo orientaci v projektu. Struktura repozitáře byla zorganizována do tematic-

kých celků, jako jsou `docs`, `screens` a `src`, což zjednodušuje jeho správu a případný další vývoj.

Na závěr byla provedena optimalizace celkové struktury kódu. Odstraněny byly nadbytečné části, jako například nepoužívané zvukové soubory nebo redundantní funkce, což přispělo k lepší přehlednosti a udržovatelnosti projektu.

Výsledná aplikace poskytuje bezpečné, přístupné a přehledné prostředí pro e-mailovou komunikaci, které je plně přizpůsobeno potřebám cílové skupiny. Všechny zdrojové kódy projektu jsou dostupné na GitHubu: `https://github.com/forsenior/senior-os`.

Literatura

[1] FERGUSON, Stuart a HEBELS, Rodney. *Computers for Librarians*. 3. vydání. Chandos Publishing, 2003. ISBN 978-1-876938-60-4. Dostupné z: https://www.sciencedirect.com/book/9781876938604/computers-for-librarians [cit. 2025-11-21].

[2] YANG, A.; STEELE, S.; FREED, N. *Internationalized Email Headers*. Online. RFC 6532. February 2012. Dostupné z: https://www.rfc-editor.org/rfc/rfc6532 [cit. 2024-11-21].

[3] CLOUDFLARE, Inc. *What is Email? | Email Definition*. Online. San Francisco: Cloudflare, Inc., 2024. Dostupné z: https://www.cloudflare.com/learning/email-security/what-is-email/ [cit. 2024-11-22].

[4] BANJAC, Gordan. *HTML vs. Rich Text vs. Plain Text Email Formats*. Online. Publikováno dne 03/19/2019. Gimmio. Dostupné z: https://blog.gimm.io/html-vs-rich-text-vs-plain-text-email-formats/ [cit. 2024-11-22].

[5] APPLE INC. *Používání prostého nebo formátovaného textu ve zprávách v Mailu na Macu*. Online. Cupertino: Apple Inc., 2024. Attn: Copyright Agent, One Apple Park Way, MS:39-1IPL, Cupertino, CA 95014. Dostupné z: https://support.apple.com/cs-cz/guide/mail/mlhlp1009/mac [cit. 2024-11-28].

[6] TRACY, Miles; JANSEN, Wayne; SCARFONE, Karen; BUTTERFIELD, Jason. *NIST Special Publication 800-45 Version 2, Guidelines on Electronic Mail Security*. Online. Gaithersburg: National Institute of Standards and Technology, 2007. Dostupné z: https://csrc.nist.gov/pubs/sp/800/45/ver2/final. [cit. 2024-11-28].

[7] RFC EDITOR. *Domain names - concepts and facilities*. Online. Series: Request for Comments, No. 1034. Publisher: RFC Editor, November 1987. Dostupné z: https://www.rfc-editor.org/info/rfc1034. [cit. 2024-12-05].

[8] MELNIKOV, Alexey a LEIBA, Barry. *Internet Message Access Protocol (IMAP) - Version 4rev2*. Online. Series: Request for Comments, No. 9051. Publisher: RFC Editor, August 2021. Dostupné z: https://www.rfc-editor.org/info/rfc9051. [cit. 2024-12-05].

[9] BORENSTEIN, Nathaniel S. a FREED, Ned. *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. Online. Series: Request for Comments, No. 1341. Publisher: RFC Editor, June 1992. Dostupné z: `https://www.rfc-editor.org/info/rfc1341`. [cit. 2024-12-05].

[10] ROSE, Marshall T. a MYERS, John G. *Post Office Protocol - Version 3*. Online. Series: Request for Comments, No. 1939. Publisher: RFC Editor, May 1996. Dostupné z: `https://www.rfc-editor.org/info/rfc1939`. [cit. 2024-12-05].

[11] PYTHON SOFTWARE FOUNDATION. *Python FAQ - General Information*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: `https://docs.python.org/3.12/faq/general.html`. [cit. 2024-12-05].

[12] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: os — Miscellaneous operating system interfaces*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: `https://docs.python.org/3/library/os.html`. [cit. 2024-12-05].

[13] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: sys — System-specific parameters and functions*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: `https://docs.python.org/3/library/sys.html`. [cit. 2024-12-05].

[14] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: datetime — Basic date and time types*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: `https://docs.python.org/3/library/datetime.html`. [cit. 2024-12-05].

[15] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: ssl — TLS/SSL wrapper for socket objects*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: `https://docs.python.org/3/library/ssl.html`. [cit. 2024-12-05].

[16] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: subprocess — Subprocess management*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: `https://docs.python.org/3/library/subprocess.html`. [cit. 2024-12-05].

[17] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: threading — Higher-level threading interface*. Online. Python Software Foundation,

2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3/library/threading.html. [cit. 2024-12-05].

[18] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: re — Regular expression operations*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3/library/re.html. [cit. 2024-12-05].

[19] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: email — Package for managing email messages*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3/library/email.html. [cit. 2024-12-05].

[20] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: email.mime — Subpackage for managing MIME-encoded email messages*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3/library/email.mime.html. [cit. 2024-12-05].

[21] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: imaplib — IMAP4 protocol client*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3/library/imaplib.html. [cit. 2024-12-05].

[22] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: smtplib — SMTP protocol client*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3.12/library/smtplib.html. [cit. 2024-12-05].

[23] PYTHON SOFTWARE FOUNDATION. *The Python Standard Library: runpy — Locating and executing Python modules*. Online. Python Software Foundation, 2001-2024. Poslední aktualizace: 5. 12. 2024. Dostupné z: https://docs.python.org/3.12/library/runpy.html. [cit. 2024-12-05].

[24] READ THE DOCS. *Chardet: The Universal Character Encoding Detector*. Online. © Copyright 2015, Mark Pilgrim, Dan Blanchard, Ian Cordasco. Dostupné z: https://chardet.readthedocs.io/en/latest/. [cit. 2024-12-05].

[25] RIVERBANK COMPUTING LIMITED. *PyQt5 Documentation*. Online. © Copyright 2023, Riverbank Computing Limited, The Qt Company. Dostupné z: https://www.riverbankcomputing.com/static/Docs/PyQt5/. [cit. 2024-12-05].

[26] LUNDH, Fredrik a CLARK, Jeffrey A. *Pillow Documentation*. Online. © Copyright 1995-2011 Fredrik Lundh and contributors, 2010-2024 Jeffrey A. Clark and contributors. Dostupné z: `https://pillow.readthedocs.io/en/stable/`. [cit. 2024-12-05].

[27] KURCZEWSKI, Marcin. *Screeninfo Documentation*. Online. © Copyright 2015 Marcin Kurczewski. Licence: MIT License (MIT). Dostupné z: `https://github.com/rr-/screeninfo`. [cit. 2024-12-05].

[28] AKBARY, Syrus. *Validate_email*. Online. © Copyright 2014 Syrus Akbary. Licence: GNU Lesser General Public License v3.0 (LGPL). Dostupné z: `https://github.com/husisusi/py3-validate-email`. [cit. 2024-12-05].

[29] NAG, Ritvik. *Dataclass_wizard*. Online. © Copyright 2021 Ritvik Nag. Licence: Apache Software License 2.0. Dostupné z: `https://github.com/rnag/dataclass-wizard/tree/main`. [cit. 2024-12-05].

[30] EUSTACE, Sébastien a Poetry Contributors. *Poetry: Python packaging and dependency management made easy*. Online. Copyright © 2018-2024. Licence: MIT. Dostupné z: `https://python-poetry.org/docs/`. [cit. 2024-12-05].

[31] VALA, Robin. *Emailový klient pro seniory*. Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, 2024. Diplomová práce. Vedoucí práce: prof. Ing. Dan Komosný, Ph.D. Dostupné z: `https://www.vut.cz/studenti/zav-prace/detail/159170` [cit. 2024-12-05].

[32] DASARI, Bhumika. *Mastering Python Dependency Management with Poetry*. Medium [online]. 2024-09-01 [cit. 2024-04-23]. Dostupné z: `https://medium.com/@bhumikadasari0/mastering-python-dependency-management-with-poetry-a` Obrázek převzat z: *Image taken from ML in Production: From Data Scientist to ML Engineer by Andrew Wolf, Ilya Fursov — Udemy.*

[33] CLOUDFLARE, Inc. *2023 Phishing Threats Report*. [online]. San Francisco: Cloudflare, Inc., 2023. [cit. 2024-04-23]. Dostupné z: `https://blog.cloudflare.com/2023-phishing-report/`

[34] CLOUDFLARE, Inc. *Cloudflare 2023 Phishing Threats Report*. [online]. San Francisco: Cloudflare, Inc., 2023. [cit. 2024-04-23]. Dostupné z: `https://www.cloudflare.com/lp/2023-phishing-report/`

[35] CLOUDFLARE, Inc. *What is a Phishing Attack?*. [online]. San Francisco: Cloudflare, Inc., 2024. [cit. 2024-04-23]. Dostupné z:

https://www.cloudflare.com/learning/access-management/phishing-attack/

[36] KPCS CZ, s.r.o. *Techniky sociálního inženýrství*. [online]. Praha: KPCS CZ, s.r.o., 2023. [cit. 2024-04-23]. Dostupné z: https://www.kpcs.cz/cs/novinky/blog/techniky-socialniho-inzenyrstvi.html

[37] VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. *Nenechte se (na)chytat: Jak se bránit phishingovým útokům*. [online]. Brno: Vysoké učení technické v Brně, 2023. [cit. 2024-04-23]. Dostupné z: https://www.vut.cz/vut/aktuality-f19528/nenechte-se-(na)chytat-jak-se-branit-phishingovym-utokum-d258664

[38] FAKULTA STROJNÍHO INŽENÝRSTVÍ VUT V BRNĚ. *Pozor na phishing – buďte obezřetní*. [online]. Brno: Fakulta strojního inženýrství VUT v Brně, 2023. [cit. 2024-04-23]. Dostupné z: https://www.fme.vutbr.cz/clanky/clanek/73716

[39] ITNETWORK.CZ. *Regulární výrazy v Pythonu*. [online]. ITnetwork.cz, 2024. [cit. 2024-04-23]. Dostupné z: https://www.itnetwork.cz/python/kolekce/regularni-vyrazy-v-pythonu

A Obsah elektronické přílohy

Elektronická příloha této práce obsahuje kompletní projekt vyvíjené aplikace Senior-OS včetně všech potřebných zdrojových souborů, dokumentace, grafických podkladů a návodů k použití. Součástí přílohy jsou jak nově vytvořené soubory, tak rozšířené části předchozích projektů a převzaté soubory, které nebyly měněny.

Struktura adresářů a souborů je popsána níže včetně barevného rozlišení, které pomáhá odlišit nově vytvořené soubory od těch rozšířených či převzatých. Součástí přílohy jsou také soubory typu `.diff`, které obsahují barevně vyznačené změny provedené v rozšířených souborech.

Legenda barev:

- Červeně – vlastní nově vytvořené soubory v rámci této práce a rozšířené soubory z předchozí práce.
- Modře – soubory s přehledem změn, které obsahují zvýraznění nově přidaných nebo upravených částí kódu.
- Černě – převzaté a automaticky generované soubory beze změn.

```
senior-os/ ..... kořenový adresář projektu
├── sconf/ ..... modul pro správu konfigurace
│   ├── icons/
│   │   ├── smail_person_cartoon_0.png ..... kreslený obrázek uživatele 0
│   │   ├── smail_person_cartoon_1.png ..... kreslený obrázek uživatele 1
│   │   ├── ...
│   │   └── smail_person_cartoon_15.png ..... kreslený obrázek uživatele 15
│   └── src/sconf/
│       └── configuration/
│           └── models/
│               ├── global_configuration.py .... globální nastavení parametrů platné
│               │   napříč aplikacemi
│               └── smail_configuration.py ..... konfigurace pro aplikaci SMAIL
└── smail/ ..... e-mailový klient pro seniory
    ├── docs/
    │   ├── smail_manual_cz.md ..... manuál k projektu SMAIL v češtině
    │   ├── smail_manual_de.md ..... manuál k projektu SMAIL v němčině
    │   ├── smail_manual_en.md ..... manuál k projektu SMAIL v angličtině
    │   ├── smail_manual_cz.pdf ..... pdf manuál k projektu SMAIL v češtině
    │   ├── smail_manual_de.pdf ..... pdf manuál k projektu SMAIL v němčině
    │   └── smail_manual_en.pdf ..... pdf manuál k projektu SMAIL v angličtině
    └── screens/
        ├── smail_email_send_cz.png ..... snímek s potvrzením úspěšného odeslání
        ├── smail_sensitive_data_alert_cz.png .. snímek s upozorněním na citlivé
        │   údaje
        └── smail_unconfirmed_email_cz.png .... snímek s varováním o rozepsaném
            konceptu e-mailu
```

- smail_screen1_cz.png ..... snímek přijatého e-mailu
- smail_screen2_cz.png ..... snímek prostředí pro psaní nového e-mailu
- src/smail/
  - connection/
    - command_line_mail.py ..... odesílání e-mailu z příkazové řádky
    - command_line_mail.py.diff ..... přehled změn souboru
    - mail_connection.py ..... soubor pro práci s e-mailovými servery
    - mail_connection.py.diff ..... přehled změn souboru
  - __main__.py ..... hlavní vstupní soubor při spuštění aplikace
  - layout.py ..... hlavní část aplikace
  - layout.py.diff ..... přehled změn souboru
  - style.py ..... definice stylů a vzhledu prvků GUI
  - style.py.diff ..... přehled změn souboru
- README.md ..... základní popis projektu
- credits.txt ..... seznam citací pro ikony
- poetry.lock ..... zamknuté verze knihoven používaných projektem
- pyproject.toml ..... nastavení projektu a seznam závislostí pro Poetry
- requirements.txt ..... seznam knihoven pro instalaci
- smail_run.py ..... skript pro spuštění aplikace SMAIL