UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2023 Radek Sahliger

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Automatické testování GUI aplikací

Bakalářská práce

2023 Radek Sahliger

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Radek Sahliger**
Osobní číslo:	**I20151**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Automatické testování GUI aplikací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvoření manuálu pro automatické testování GUI aplikací pro operační systém Linux. V teoretické části práce bude popsána problematika automatického testování za pomocí nástroje xdotool. V praktické části práce student vytvoří testovací bash skripty pomocí kterých otestuje některé aplikace, např. Calculator, Thunderbird.

Rozsah pracovní zprávy: **min 30. stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

MASTERS, Jon a Richard BLUM. *Linux profesionálně: programování aplikací*. Brno: Zoner Press, 2008. Encyklopedie Zoner Press. ISBN 978-80-86815-71-8.

MITCHELL, Mark, Alex SAMUEL a Jeffrey OLDHAM. *Pokročilé programování v operačním systému Linux*. Přeložil Miroslav DRESSLER. Praha: SoftPress, [2002]. ISBN 80-86497-29-1.

Vedoucí bakalářské práce: **Ing. Miroslav Dvořák, Dipl.tech.**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem Automatické testování GUI aplikací jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze 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, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 5. 6. 2023

Radek Sahliger

ANOTACE

Bakalářská práce se zabývá vytvářením testovacích skriptů pro různé aplikace s grafickým uživatelským rozhraním. Úkony, které provádí uživatel, se simulují za pomoci nástrojů, které provádí klikání myší a stisky kláves.

V této práci jsem vytvářel skripty se dvěma nástroji. Prvním je nástroj xdotool, který lze využívat v terminálu operačního systému, či vytvářet bash skripty, které využívají tento nástroj. Druhý nástroj využívá skriptovací jazyk Python a knihovnu PyAutoGUI.

Tyto nástroje fungují na velmi podobné bázi, ale každý má svoje výhody a nevýhody.

KLÍČOVÁ SLOVA

Grafické uživatelské rozhraní, Linux, Bash, Python, PyCharm, Visual Studio Code, aplikace, PyQT5, Qt Designer, xdotool, PyAutoGUI, Xdo

TITLE

Automatic testing of GUI applications

ANNOTATION

Bachelor thesis is focused on creating of testing scripts for various applications with graphical user interface. The tasks, that user does, are simulated using tools, which take care of clicking the mouse buttons and pressing keys on the keyboard.

I this thesis i created scripts using two different tools. The first one is xdotool, which can be used directly from the operating system terminal, or to create bash scripts. The second one is the programming language Python and its library PyAutoGUI.

These tool work on the same principle, but each one has its pros and cons.

KEYWORDS

Graphical user interface, Linux, Bash, Python, PyCharm, Visual Studio Code, application, PyQT5, Qt Designer, xdotool, PyAutoGUI, Xdo

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK A ZNAČEK

GUI	Graphical User Interface
GNU	Gnu's Not Unix!
IDE	Integrated Development Environment
SQL	Structured Query Language
SVG	Scalable Vector Graphics
VS	Visual Studio
WYSIWYG	What-You-See-Is-What-You-Get
XML	Extensible Markup Language

ÚVOD

V dnešní době je velmi běžné vyvíjet aplikace s GUI, kvůli dostupnosti a jednoduchosti používání. Pří vyvíjení i po ukončení vývoje je potřeba aplikaci testovat a zkoušet její funkcionalitu.

Pokud si chce vývojář testovat svoji GUI aplikaci, musí si neustále dokola zkoušet určité úkony s aplikací, dokud se aplikace nechová, tak jak si vývojář představuje.

Tento proces vývoje aplikace by bylo vhodné usnadnit jeho automatizací. Pokud vývojář využije automatické testy, může lépe sledovat postupnou práci aplikace a sledovat případné chyby. Vývojář může také být produktivnější, pokud spustí automatické testování a poté se může věnovat jiným problémům.

Cílem této bakalářské práce je takové skripty vytvořit. Nejprve se vytvořily dva testovací skripty pomocí jazyka Bash pro aplikace kalkulátoru a e-mailového klienta Mozilla Thunderbird. Dále pro tyto aplikace byly vytvořeny dva testovací skripty v jazyce Python za pomoci knihovny PyAutoGUI.

V poslední části práce byla vytvořena jednoduchá aplikace pro zaznamenávání poznámek a následně otestování dvěma skripty obou nástrojů.

1 POUŽITÉ TECHNOLOGIE

1.1 Operační systém

Prvním krokem této práce bylo zvolit vhodný operační systém pro práci s danými nástroji. Jelikož nástroj xdotool je určený pro unixové systémy, byl výběr omezený na tento typ systémů. Po zhodnocení jednotlivých alternativ bylo rozhodnuto využívat operační systém Kali Linux. Tento systém byl zvolen kvůli své robustnosti, velké škále užitečných nástrojů, které jsou v základní instalaci systému zahrnuty, a také kvůli předchozím zkušenostem s tímto systémem. Systém byl provozován pomocí programu VirtualBox ve virtuálním prostředí.

1.1.1 Kali Linux

Kali Linux (dříve známý jako BackTrack Linux) je open-source distribuce založená na Debianu, která se primárně zaměřuje na penetrační testování a bezpečnostní auditování. Toto zajišťuje poskytováním běžných nástrojů, konfigurací a automatizací, které umožňují uživateli zaměřit se na úkon, který je potřeba vyřešit.

Kali Linux obsahuje specifické úpravy a také stovky nástrojů zaměřených na různé úkony z oboru Bezpečnosti informací, jako například penetrační testování, zkoumání bezpečnosti, počítačová forenzika, reverzní inženýrství, řízení zranitelností a testování pomocí červeného týmu. [1]

1.1.2 VirtualBox

Pro běh operačního systému Kali Linux na hostitelském počítači byl využitý software VirtualBox od firmy Oracle.

VirtualBox je výkonný x86 a AMD64/Intel64 virtualizační software pro firmy i domácí uživatele. VirtualBox je volný zdarma jako open-source software za podmínek GNU General Public Licence (GPL) verze 3. [2]

1.2 Vývojové prostředí

Pro vývoj aplikace a tvorbu skriptů bylo využito několik vývojových prostředí.

Prvním vývojovým prostředím byl Visual Studio Code. V tomto IDE byly vytvářeny skripty ve skriptovacím jazyce Bash. Druhým vývojovým prostředím byl PyCharm, ve kterém byla vytvořena aplikace a také testy knihovnou PyAutoGUI. Posledním prostředím byl Qt Designer, ve kterém byly vytvořeny .ui soubory, které se využívají pro definici rozložení, vzhledu a funkcionality uživatelského rozhraní.

1.2.1 Visual Studio Code

Visual Studio Code je lehký, ale výkonný editor zdrojového kódu. Je dostupný pro Windows, macOS a Linux. VS Code obsahuje zabudovanou podporou pro JavaScript, TypeScript a Node.js. Má velmi velké množství rozšíření určené pro další jazyky a runtimy (například C++, C#, Java, Python, Go, .NET). [3]

Visual Studio Code byl využit při vytváření bashových testovacích skriptů pro aplikace kalkulátoru, e-mailového klienta a aplikace vytvořené pro tuto práci. Pro práci s Bashem bylo potřeba stažení Bash rozšíření, jelikož toto prostředí nemá podporu jazyka Bash předinstalovanou.

1.2.2 PyCharm

PyCharm je integrované vývojové prostředí (IDE) pro Python. Je vyvíjený českou softwarovou společností JetBrains. Samotné prostředí je napsané v kombinaci Javy a Pythonu a je multiplatformní (vydávané pro Linux, Microsoft Windows a macOS). [4]

V PyCharmu byla vyvíjena samotná aplikace pro tuto bakalářskou práci a také testovací skripty pro danou aplikaci a také pro již zmíněné ostatní aplikace.

PyCharm byl zvolen pro svou intuitivnost používání a jednoduchou instalaci dodatečných rozšiřujících knihoven pomocí nástroje Python Packages.

1.2.3 Qt Designer

Qt Designer je Qt nástroj, který poskytuje WYSIWYG uživatelské rozhraní k vývoji GUI pro PyQt aplikace produktivně a efektivně. S tímto nástrojem lze vytvářet GUI metodou drag and drop. Tyto rozhraní se skládají z objektů QWidget, které se vkládají do formulářů. Poté lze tyto objekty rozmísťovat pomocí různých manažerů rozložení. [5]

Qt Designer produkuje soubory .ui, které se v pythonovém kódu za pomocí knihovny PyQt načítají k následující práci s uživatelským rozhraním.

1.3 Jazyky

Bakalářská práce byla provedena ve dvou jazycích. Jedním z nich je jazyk Bash, který sloužil k testování různých aplikací. Dalším byl jazyk Python, který sloužil k vývoji aplikace poznámek a také následnému testování této aplikace i aplikací ostatních.

1.3.1 Bash

Bash je skriptovací jazyk, který je součástí systému Kali Linux, což znamená, že není nutnost instalovat tento jazyk do systému. Další výhodou tohoto jazyka je možnost využívat terminálové příkazy i nástroje, které se ovládají pomocí terminálu.

Bash je shell vytvořený pro operační systémy GNU. Dnes je to jeden z nejpoužívanějších shellů na Linuxu. Shell je program, který v unixových operačních systémech vytváří základní textové rozhraní mezi uživatelem a operačním systémem. Když si spustíte v Linuxu textovou konzoli, textový kurzor, který uvidíte, je vlastně součástí shellu, který tím dává uživateli na vědomí, že očekává příkazy. Základní funkčností shellu je spouštět uživatelské příkazy (většinou programy), ale dnešní shelly toho umí mnohem více. V unixovém světě neexistuje jediný shell, existuje jich poměrně hodně a vzájemně se liší ovládáním a schopnostmi, ačkoli základní funkcionalita je stejná. Všechny shelly umí pracovat jak v interaktivním, tak v dávkovém režimu. [6]

1.3.2 Python

Python je velmi efektivní a přehledný jazyk. Pro tento jazyk existuje nespočet knihoven s velmi širokým záběrem působnosti.

Python je vysokoúrovňový programovací jazyk, který v roce 1991 navrhl Guido van Rossum. Nabízí dynamickou kontrolu datových typů a podporuje různá programovací paradigmata, včetně objektově orientovaného, imperativního nebo funkcionálního. V roce 2018 vzrostla jeho popularita a zařadil se mezi nejoblíbenější jazyky. V řadě různých žebříčků dosahuje jedno z prvních třech míst, výjimkou nebývají první místa. Python je vyvíjen jako open source projekt, který zdarma nabízí instalační balíčky pro většinu běžných platforem (Unix, MS Windows, macOS, Android). Ve většině distribucí systému GNU/Linux je Python součástí základní instalace. [7]

1.4 Důležité nástroje a knihovny

Jednotlivé testy a aplikace bylo možné naprogramovat pomocí několika stěžejních knihoven a systémových nástrojů. Konkrétně se jedná o knihovnu PyQt, knihovnu PyAutoGUI a linuxový nástroj xdotool.

1.4.1 PyQt

PyQt je Python binding pro Qt, což je soubor C++ knihoven a vývojových nástrojů poskytující vývoj GUI aplikací nezávislých na systému. Qt také poskytuje nástroje pro síťování, práci s vlákny, regulárními výrazy, SQL databázemi, SVG, OpenGL, XML a další užitečné prvky. [8]

Nejnovějšími verzemi jsou:

- PyQt5
- PyQt6

Aplikace pro tuto bakalářskou práci byla vyvinuta pomocí verze PyQt5

1.4.2 PyAutoGUI

PyAutoGUI je knihovna pro jazyk Python, která umožňuje ovládání počítače pomocí pythonového programu. Knihovna simuluje vstup myši a klávesnice, čímž si programátor usnadní své testování GUI aplikace.

Výhodou této knihovny je fakt, že se používá s jazykem Python, který je velmi přehledný a poměrně jednoduchý na naučení. Také je možnost tuto knihovnu kombinovat s ostatními knihovnami.

Nevýhodou této knihovny je to, že v systému Linux tato knihovna nedisponuje možností ovládat okna. Nelze ručně nastavit aktivní okno, či okno posunout na potřebné místo. Pokud je tato funkcionalita žádoucí, je nutné využití jiné knihovny, která tuto funkcionalitu zajišťuje. V této práci jsem jako náhradu využil knihovnu libxdo, která umožnuje využívat funkce nástroje Xdotool z prostředí pythonového programu.

1.4.3 Xdotool

Xdotool je linuxový nástroj pro simulaci uživatelského vstupu. Lze pomocí tohoto nástroje simulovat kliknutí myší nebo stisky kláves. Na rozdíl od knihovny PyAutoGUI disponuje tento nástroj možností manipulovat spuštěná okna v systému.

Oproti již zmíněné knihovně není nutná instalace žádného vývojového prostředí. Funkce nástroje lze využívat i v terminálu linuxového systému. Také je tento způsob automatizace vhodný pro vykonávání pravidelných úkonů, které mohou být zdlouhavé. V kombinaci s nástrojem crontab je tento nástroj velmi silným kandidátem pro tyto účely.

Nevýhodou tohoto řešení může být neznalost jazyka Bash, který je dle mého názoru komplikovanějším jazykem na naučení oproti jazyku Python.

2 PRAKTICKÁ ČÁST

2.1 Aplikace poznámek pro účely automatického testování

Cílem této práce je automatizace testování GUI aplikací. Jelikož je testování součástí náplně programování, byla pro tuto práci vytvořena jednoduchá aplikace na poznámky, která byla následně testována za pomocí nástroje xdotool a knihovny PyAutoGUI.

Aplikace se stará o ukládání a zobrazování poznámek uživatele. Uživatel si může pomocí tlačítka přidávat jednotlivé poznámky, které si pojmenuje. Tyto poznámky uchovávají kromě svého názvu také text samotných poznámek. Aplikace umožnuje uživateli také uložené poznámky upravovat a mazat.

Aplikaci lze ovládat pomocí tlačítek uživatelského rozhraní, ale také pomocí tlačítek klávesnice pro rychlejší ovládání. Jelikož ne každý uživatel chce ovládat aplikaci pouze pomocí tlačítek rozhraní, umožnuje aplikace mazání, vybírání a vytváření nových poznámek vstupem z klávesnice.

2.2 Vzhled aplikace

Vzhled pro tuto aplikaci byl definován pomocí nástroje Qt Designer, který umožnuje jednoduché rozložení ovládacích prvků aplikace. Toto rozložení prvků je uloženo do .ui souboru, ze kterého se v programové části využívá dané rozložení a pojmenování jednotlivých komponent.

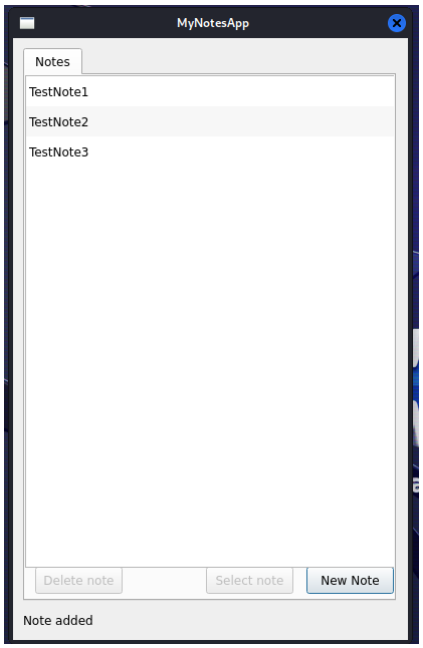
Aplikace se skládá ze dvou oken, která zajištují svou funkcionalitu aplikace.

2.2.1 Hlavní okno aplikace

Hlavní okno se skládá z prvku QTabWidget, což je prvek, který umožňuje vytvářet různé rozložení prvků do jednotlivých záložek. Tato aplikace se skládá pouze z jedné záložky, takže tento potenciál nebyl využit. Je ovšem možnost aplikaci rozšiřovat o další funkcionalitu, na což je velmi vhodný tento prvek. V tomto okně nalezneme také seznam jednotlivých poznámek.

Tímto prvkem je QListView, což je jednoduchý seznam položek. V tomto prvku lze vybírat poznámky určené k úpravě, či ke smazání.

Dále tato záložka obsahuje samotná tlačítka pro vybírání určené poznámky, tlačítko pro mazání vybrané poznámky a tlačítko pro vytváření nové poznámky. Tato tlačítka jsou komponentou QPushButton.



Obrázek 1 - Hlavní okno aplikace

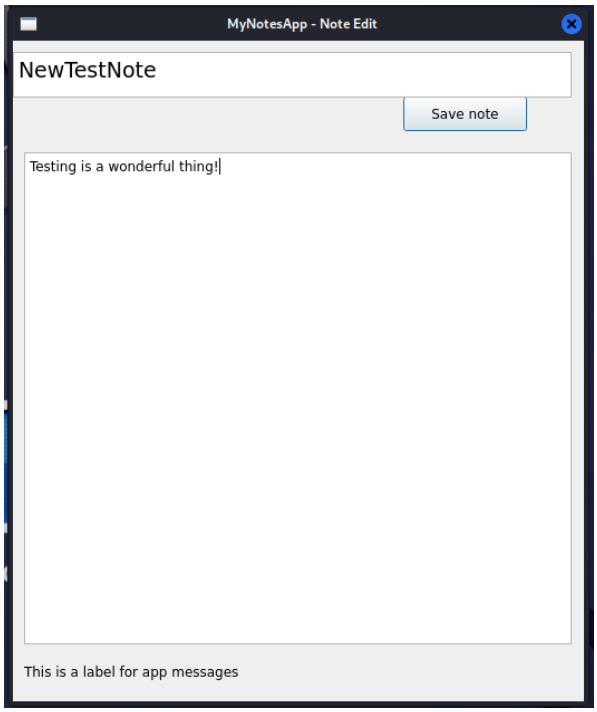
2.2.2 Okno nové/detailu poznámky

Druhé okno se stará o funkci vytváření nových poznámek a také úpravy již uložených poznámek. Toto okno se skládá ze tří hlavních komponent.

První komponentou je QTextEdit, což je Qt editor textu. Do této komponenty patří pojmenování poznámky. Při úpravě poznámky se zde nachází jméno poznámky, které již nelze změnit. Při vytváření poznámky zde uživatel zadává pojmenování poznámky.

Další komponentou je druhý objekt QTextEdit, který obsahuje samotný text poznámky. Do tohoto textového editoru uživatel zadává jednotlivé řádky poznámky, které si přeje uchovat pod daným pojmenováním.

Poslední komponentou je tlačítko QPushButton, které se stará o ukládání hotových poznámek.



Obrázek 2 - Okno nové poznámky

2.3 Vývoj aplikace

Jednotlivé soubory s kódem aplikace a datovými strukturami, se kterými aplikace pracuje, se nacházejí v jedné složce dohromady.

Konkrétně se jedná o soubory main.py, Note.py, NoteList.py, NoteGUITest.py, MyNotesBashTest.sh, notes.ui a noteWindow.ui.

V souboru main.py se skrývá veškerá funkcionalita aplikace. V tomto souboru se volají jednotlivé funkce pro načítání .ui souborů a následně se specifikují funkce a propojení mezi komponentami aplikace.

V souboru Note.py se nachází datová struktura pro jednotlivé poznámky. Tato třída uchovává název poznámky, poznámku samotnou a pro lepší porovnávání jednotlivých poznámek uchovává i hodnotu noteHash.

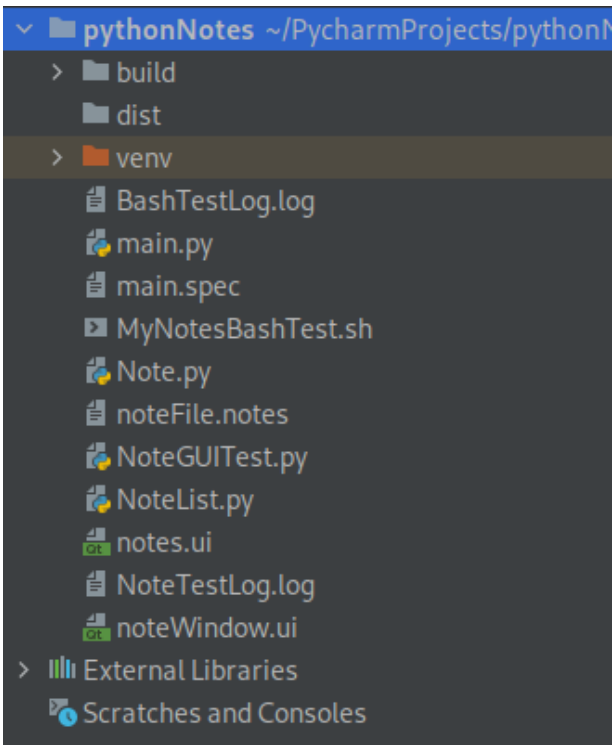
V souboru NoteList.py se nachází třída zajišťující funkce manipulace seznamu poznámek. Třída si uchovává list, do kterého ukládá poznámky.

Soubor NoteGUITest.py obsahuje kód první části testu aplikace pomocí knihovny PyAutoGUI.

Soubor MyNotesBashTest.sh obsahuje skript v jazyce Bash, který zajišťuje druhou část testování aplikace pomocí nástroje xdotool.

Soubor notes.ui obsahuje rozložení komponent v hlavním okně aplikace.

Soubor noteWindow.ui obsahuje rozložení komponent v okně konkrétní poznámky.



Obrázek 3 - Soubory poznámkové aplikace

2.3.1 Třída Note

Třída Note je základním stavebním prvkem celé aplikace. Jedná se o třídu, která uchovává údaje o jednotlivých poznámkách. Uchovává se název poznámky, obsah poznámky a hodnota noteHash.

Pro vytváření instancí této třídy je definována funkce init, která je zavolána při vytváření nové instance této třídy.

```
class Note:
    def __init__(self, name, content):
        self.name = name
        self.content = content
        self.noteHash = hash(self.name) + hash(self.content)
```

Obrázek 4 - Třída Note

Pokud je potřebné, aby se tento objekt zobrazoval jako text, tak je definována funkce str, která převádí tento objekt na jednoduchý text. V tomto případě se jedná o název poznámky.

```
def __str__(self):
    return self.name
```

Obrázek 5 - Funkce str

Dále je zde definována funkce rehash. Tato funkce se volá, pokud je vhodné přepočítat hash dané poznámky například při změně obsahu poznámky. Hodnota noteHash je součtem hashových hodnot názvu a obsahu poznámky. Výpočet hashové hodnoty je zajištěn funkcí hash, která je standardní funkcí jazyka Python.

```
def rehash(self):
    self.noteHash = hash(self.name) + hash(self.content)
```

Obrázek 6 - Funkce rehash

2.3.2 Třída NoteList

Tato třída obsahuje seznam uložených poznámek. Pro práci se seznamem využívá své funkce, které zajišťují správnou funkčnost aplikace. Tento seznam se inicializuje ve funkci init, která je zavolána při vytvoření instance této třídy.

```
class NoteList:
    def __init__(self):
        self.thisList = list()
```

Obrázek 7 - Třída NoteList

Zásadní funkcí, která je potřebná k tomu, aby nebyly v seznamu duplikátní hodnoty je funkce checkDuplicate. Tato funkce obdrží název poznámky, a poté zkontroluje každou položku v seznamu a vyhledá, zda seznam již takto pojmenovanou položku obsahuje.

```
def checkDuplicate(self, notename):
    check = -1
    if len(self.thisList) > 0:
        for x in self.thisList:
            if x.name == notename:
                check = 1
                break
    return check
```

Obrázek 8 - Funkce checkDuplicate

Pomocnou funkcí, která je několikrát využívaná v programu, je funkce getList, která vrací seznam, který si tato třída uchovává.

```
def getList(self):
    return self.thisList
```

Obrázek 9 - Funkce getList

Dvě velmi podobné funkce jsou ty, které vrací jednotlivé poznámky ze seznamu. Jednou z těchto funkcí je getFromName, která vrací poznámku na základě názvu poznámky. Druhou je funkce get, která vrací poznámku na základě hashové hodnoty poznámky.

```
def getFromName(self, notename):
    if len(self.thisList) > 0:
        for x in self.thisList:
            if x.name == notename:
                return x
    return None
```

Obrázek 10 - Funkce getFromName

```
def get(self, note: Note):
    if len(self.thisList) > 0:
        for x in self.thisList:
            if x.noteHash == note.noteHash:
                return x
    return None
```

Obrázek 11 - Funkce get

Důležitou funkcí je také funkce addNote, která zajišťuje přidávání poznámek do seznamu. Za pomoci funkce checkDuplicate zkontroluje přítomnost existující poznámky a v případě, že se jedná o ojedinělý název, přidá poznámku do seznamu.

```
def addNote(self, note: Note):
    if self.checkDuplicate(note.name) == -1:
        self.thisList.append(note)
        return 0
    else:
        return -1
```

Obrázek 12 - Funkce addNote

Další dvě podobné funkce zajišťují úpravu a mazání poznámek ze seznamu. Funkce changeNote na základě názvu poznámky její záznam vyhledá a změní původní poznámku na novou. Funkce deleteNote vyhledá na základě hashové hodnoty správnou poznámku a vymaže danou poznámku ze seznamu.

```
def changeNote(self, note: Note):
    if len(self.thisList) > 0:
        for x in range(len(self.thisList)):
            if self.thisList[x].name == note.name:
                self.thisList[x] = note
```

Obrázek 13 - Funkce changeNote

```
def deleteNote(self, note: Note):
    if len(self.thisList) > 0:
        for x in self.thisList:
            if x.noteHash == note.noteHash:
                self.thisList.remove(x)
```

Obrázek 14 - Funkce deleteNote

Poslední funkcí je funkce pomocná a jedná se o čistění seznamu. Tato funkce není teoreticky důležitá pro funkčnost aplikace, ale je užitečná při testování aplikace, kde je využívaná.

```
def clearList(self):
    self.thisList.clear()
```

Obrázek 15 - Funkce clearList

2.3.3 Třída UI

Pro správnou funkcionalitu této třídy a ostatních tříd uchovávaných v souboru main.py je nutný import následujících knihoven a funkcí. Třídy v tomto souboru také používají globální proměnnou, která uchovává instanci třídy NoteList.

```
import threading

from PyQt5.QtWidgets import *
from PyQt5 import uic
from PyQt5.QtGui import *
from PyQt5.QtCore import *

import NoteGUITest
from Note import Note
from NoteList import NoteList

import sys
import os

noteList = NoteList()
```

Obrázek 16 - Importované knihovny

Třída UI je základním stavebním kamenem celé aplikace. Tato třída zajištuje nahrání .ui souboru s rozložením aplikace a následným mapováním daných ovládacích prvků k patřičným funkcím třídy. Toto načítání se provádí ve funkci init, která je spuštěna při vytvoření instance třídy.

```
class UI(QDialog):
    def __init__(self):
        super(UI, self).__init__()
        uic.loadUi("notes.ui", self)
```

Obrázek 17 - Třída UI

Dále tato funkce vytváří lokální proměnné, do kterých vkládá instance potřebných tříd ovládacích prvků. Do těchto proměnných ukládá instance pomocí funkce findChild, která vyhledá ovládací prvek podle jména a typu prvku.

```
self.noteList: QListView = self.findChild(QListView, "notesList")
self.deleteButton: QPushButton = self.findChild(QPushButton, "deleteButton")
self.newButton: QPushButton = self.findChild(QPushButton, "newButton")
self.selectButton: QPushButton = self.findChild(QPushButton, "selectButton")
self.logLabel: QLabel = self.findChild(QLabel, "logLabel")
```

Obrázek 18 - Vytvoření instancí ovládacích prvků

Poté se těmto instancím prvků přiřadí jednotlivé funkce, které jim připadají. Toto propojení zajišťuje kombinace funkcí clicked a connect, které přiřadí funkci danému ovládacímu prvku. Tyto funkce se volají po stisknutí daného tlačítka, či seznamu.

```
self.newButton.clicked.connect(self.newClick)
self.deleteButton.clicked.connect(self.deleteClick)
self.noteList.clicked.connect(self.listClick)
self.selectButton.clicked.connect(self.selectClick)
```

Obrázek 19 - Spojení ovládacích prvků s funkcemi

Dalším krokem je vytvoření klávesových zkratek, díky kterým se usnadňuje a zrychluje ovládání aplikace při testování i při přípravě testovacích skriptů.

Konkrétně se jedná o tyto zkratky:

- Ctrl + Shift + T – Zkratka pro spuštění první části testu aplikace
- Ctrl + Shift + B – Zkratka pro spuštění druhé části testu aplikace
- Ctrl + S – Uložení stávajících poznámek do souboru
- Delete – Smazání určené poznámky ze seznamu
- S – Vybrání poznámky k úpravě
- N – Vytvoření nové poznámky

```
self.testShotcut = QShortcut(QKeySequence("Ctrl+Shift+T"), self)
self.testShotcut.activated.connect(self.runTest)
self.testBashShortcut = QShortcut(QKeySequence("Ctrl+Shift+B"), self)
self.testBashShortcut.activated.connect(self.runBashTest)
self.saveShortcut = QShortcut(QKeySequence("Ctrl+S"), self)
self.saveShortcut.activated.connect(self.saveContent)
self.deleteShortcut = QShortcut(QKeySequence("Delete"), self)
self.deleteShortcut.activated.connect(self.deleteClick)
self.selectShortcut = QShortcut(QKeySequence("S"), self)
self.selectShortcut.activated.connect(self.selectClick)
self.newShortcut = QShortcut(QKeySequence("N"), self)
self.newShortcut.activated.connect(self.newClick)
```

Obrázek 20 - Vytvoření klávesových zkratek

Poté se hlavní okno aplikace zobrazí, tlačítka pro vybírání a mazání poznámky se vypnou a nastaví se stylesheet pro QListView, aby byl seznam lépe přehledný. Dále se vytvoří lokální proměnné, které budou dále uchovávat případné instance oken pro nové poznámky a úpravy poznámek. Pokud je na počítači již uložen nějaký seznam poznámek, vyvolaná funkce loadContent jej načte do aplikace ze souboru.

```
self.show()
self.deleteButton.setEnabled(False)
self.selectButton.setEnabled(False)
self.noteList.setStyleSheet("alternate-background-color: white;background-color: lightgrey;")
self.noteList.setStyleSheet("QListView::item { height: 30px; }")
self.refreshList()

self.newWindow = None
self.changeWindow = None

self.loadContent()
```

Obrázek 21 - Úprava stylů

Jednou z důležitých funkcí je funkce selectClick. Tato funkce je vyvolána, pokud uživatel stiskne tlačítko pro úpravu poznámky nebo stiskne příslušnou klávesovou zkratku. Tato funkce zjišťuje, zda je tlačítko pro výběr povoleno a pokud tomu tak je, vybere poznámku ze seznamu a otevře nové okno pro editaci poznámky.

```
def selectClick(self):
    if self.selectButton.isEnabled():
        for index in self.noteList.selectedIndexes():
            item = self.noteList.model().itemFromIndex(index)
            self.changeWindow = ChangeNote(item.text(), self)
            self.changeWindow.show()
```

Obrázek 22 - Funkce selectClick

Funkce deleteClick se stará o mazání záznamů ze seznamu poznámek. Funkce je opět vyvolána příslušným tlačítkem nebo klávesovou zkratkou. Také kontroluje, zda je tlačítko povolené a poté smaže záznam a vyvolá obnovení seznamu poznámek.

```
def deleteClick(self):
    if self.deleteButton.isEnabled():
        for index in self.noteList.selectedIndexes():
            item = self.noteList.model().itemFromIndex(index)
            noteList.deleteNote(noteList.getFromName(item.text()))
        self.refreshList()
        self.changeLabel("Note deleted")
        self.listClick()
```

Obrázek 23 - Funkce deleteClick

Následuje pomocná funkce listClick. Tato funkce zjišťuje, zda je ze seznamu vybraná poznámka a v kladném případě povolí funkci tlačítek výběru a mazání.

```
def listClick(self):
    self.deleteButton.setEnabled(bool(self.noteList.selectedIndexes()))
    self.selectButton.setEnabled(bool(self.noteList.selectedIndexes()))
```

Obrázek 24 - Funkce listClick

Jelikož není nutné vybírání poznámky pro vytvoření nové, funkce newClick vytvoří instanci okna pro vytváření poznámky a zobrazí toto okno.

```
def newClick(self):
    self.newWindow = NewNote(self)
    self.newWindow.show()
```

Obrázek 25 - Funkce newClick

Další pomocnou funkcí je funkce refreshList. Pomocí této funkce se aktualizuje seznam poznámek po přidání, či smazání poznámky. Funkce si zažádá o seznam poznámek z třídy NoteList, převede tyto poznámky na typ QStandardItem pomocí jejich názvu a přidá tento objekt do vizuálního seznamu poznámek.

```
def refreshList(self):
    model = QStandardItemModel()
    for i in noteList.getList():
        item = QStandardItem(i.name)
        model.appendRow(item)

    self.noteList.setModel(model)
    self.listClick()
```

Obrázek 26 - Funkce refreshList

Funkci pro uložení poznámek do souboru lze vyvolat stiskem kombinace kláves výše uvedených. Funkce closeEvent ovšem tuto funkci vynutí při uzavření hlavního okna aplikace.

```
def closeEvent(self, event):
    self.saveContent()
    self.close()
```

Obrázek 27 - Funkce closeEvent

Funkce saveContent zajišťuje samotné ukládání do souboru za účelem perzistence aplikace. Záznamy v souboru se dále využívají při načítání záznamů do aktivního seznamu poznámek a také při testování bashovým skriptem. V souboru jsou jednotlivé záznamy od sebe odděleny speciálním řádkem, který je zakázáno do poznámky zapsat ručně. Jedná se o deset pomlček za sebou, pokud následuje řádek s názvem poznámky. Pokud následuje text poznámky, je před tímto řádkem řádek s pěti pomlčkami za sebou.

```
def saveContent(self):
    with open("noteFile.notes", "w") as file:
        file.write("----------\n")
        for item in noteList.getList():
            tmpNote: Note = item
            file.write(tmpNote.name.rstrip() + "\n")
            file.write("-----\n")
            file.write(tmpNote.content.rstrip() + "\n")
            file.write("----------\n")
        file.close()
        self.changeLabel("Notes saved")
```

Obrázek 28 - Funkce saveContent

Funkce loadContent zajišťuje následné načítání poznámek ze souboru. Tato funkce načítá ze souboru pouze, pokud je dodržen formát ukládání záznamů tak, jak je definováno ve funkci

saveContent a pokud tento soubor již existuje. Funkce postupně načte všechny poznámky a uloží je do seznamu a poté vyvolá aktualizaci QListView.

```
def loadContent(self):
    if not os.path.exists("noteFile.notes"):
        return

    with open("noteFile.notes", "r") as file:
        tmpLines = file.read()
        tmpLines = tmpLines.splitlines()

        tmpName = ""
        tmpContent = ""
        tmpNote: Note = None
        contentFlag = False
        if len(tmpLines) >= 4:
            for line in tmpLines:
                if line == "----------":
                    if tmpNote is not None:
                        tmpNote.changeContent(tmpContent.rstrip())
                        tmpNote.rehash()
                        noteList.addNote(tmpNote)
                        tmpName = ""
                        tmpContent = ""
                        tmpNote = None
                    contentFlag = False
                    continue
                elif line == "-----":
                    contentFlag = True
                    tmpNote = Note(tmpName.rstrip(), "")
                    continue
                else:
                    if not contentFlag:
                        tmpName += line + "\n"
                    else:
                        tmpContent += line + "\n"
            self.changeLabel("Notes loaded")
        file.close()
        self.refreshList()
```

Obrázek 29 - Funkce loadContent

Další pomocnou funkcí je changeLabel. Tato funkce je využívaná pro změnu nápisu na spodní části aplikace. Tento nápis slouží pro informování uživatele o provedených změnách.

```
def changeLabel(self, logMessage):
    self.logLabel.setText(logMessage)
```

Obrázek 30 - Funkce changeLabel

Kombinace funkcí runTest a doTest slouží pro spuštění testu pomocí knihovny PyAutoGUI. Ve funkci runTest se vymaže celý seznam poznámek a následně je vytvořeno a spuštěno vlákno s funkcí doTest, která vytvoří instanci třídy pro testování a spustí jeho průběh. Díky vytvoření vlákna je možné tento test spustit na pozadí. Bez tohoto kroku by kroky testu nefungovaly.

Obdobně se chová kombinace funkcí runBashTest a doBashTest, které jak z názvu vyplývá, spouští test pomocí nástroje xdotool, který je uložen v jiném souboru.

```
def runTest(self):
    noteList.clearList()
    self.refreshList()
    thread = threading.Thread(target=self.doTest)
    thread.start()

1 usage
def doTest(self):
    guiTest = NoteGUITest.GUITest()
    guiTest.runTest(noteList)
    self.logLabel.setText("Test was successfully completed")

1 usage
def runBashTest(self):
    thread = threading.Thread(target=self.doBashTest)
    thread.start()

1 usage
def doBashTest(self):
    guiTest = NoteGUITest.GUITest()
    guiTest.runBashTest()
    self.logLabel.setText("Bash test was successfully completed")
```

Obrázek 31 - Funkce pro spuštění testů

2.3.4 Třída NewNote

Tato třída se stará o spuštění a funkci okna pro zadávání nové poznámky do systému. V inicializační funkci init se opět načte patřičný .ui soubor a vytvoří se instance jednotlivých ovládacích prvků.

```
class NewNote(QDialog):
    def __init__(self, before: QDialog):
        self.beforeUI = before
        super(NewNote, self).__init__()
        uic.loadUi("noteWindow.ui", self)

        self.nameText: QTextEdit = self.findChild(QTextEdit, "nameText")
        self.noteText: QTextEdit = self.findChild(QTextEdit, "noteText")
        self.saveButton: QPushButton = self.findChild(QPushButton, "saveButton")
        self.logLabel: QLabel = self.findChild(QLabel, "logLabel")

        self.saveButton.clicked.connect(self.saveClicked)
```

Obrázek 32 - Třída NewNote

Jelikož toto okno obsahuje pouze jedno tlačítko, je potřebná pouze jediná funkce, která zajišťuje vytvoření instance třídy Note a její následné uložení do seznamu poznámek. Této třídě je v inicializační funkci předáno hlavní okno jako argument. Tento argument umožňuje volání funkce aktualizace seznamu poznámek z jiné třídy. Tyto kroky zajišťuje funkce saveClicked.

```
def saveClicked(self):
    tmpContent = self.noteText.toPlainText()
    contentError = None

    for line in tmpContent.splitlines():
        if line == "-----" or line == "----------":
            contentError = line

    if noteList.addNote(Note(self.nameText.toPlainText(), self.noteText.toPlainText())) == -1:
        self.logLabel.setText("Error, note was not added! Choose a different name")
    elif contentError is not None:
        self.logLabel.setText("Error, note was not added! Remove this line: " + contentError)
    else:
        UI.refreshList(self.beforeUI)
        UI.changeLabel(self.beforeUI, "Note added")
        self.close()
```

Obrázek 33 - Funkce saveClicked

2.3.5 Třída ChangeNote

Tato třída je funkčně velmi podobná třídě NewNote. Tato třída volá stejný .ui soubor, ale přidává již známé údaje do ovládacích prvků tohoto okna. Inicializační funkce si vybere ze seznamu vybranou poznámku a vyplní jméno a text poznámky do patřičných ovládacích prvků.

```
class ChangeNote(QDialog):
    def __init__(self, notename, before: QDialog):
        super(ChangeNote, self).__init__()
        uic.loadUi("noteWindow.ui", self)

        self.beforeUI = before
        self.nameText: QTextEdit = self.findChild(QTextEdit, "nameText")
        self.noteText: QTextEdit = self.findChild(QTextEdit, "noteText")
        self.saveButton: QPushButton = self.findChild(QPushButton, "saveButton")

        note = None
        self.notes = noteList
        note = self.notes.getFromName(notename)

        self.nameText.setText(note.name)
        self.nameText.setEnabled(False)
        self.noteText.setPlainText(note.content)

        self.saveButton.clicked.connect(self.saveClicked)
```

Obrázek 34 - Třída ChangeNote

Tato třída také obsahuje funkci saveClicked. Tato funkce funguje téměř stejně jako ve třídě NewNote. Rozdílem je to, že tato funkce volá změnu textu v poznámce místo přidávání poznámky do seznamu.

```
def saveClicked(self):
    tmpContent = self.noteText.toPlainText()
    contentError = None

    for line in tmpContent.splitlines():
        if line == "-----" or line == "----------":
            contentError = line
    if contentError is not None:
        self.logLabel.setText("Error, note was not updated! Remove this line: " + contentError)
    else:
        self.notes.changeNote(Note(self.nameText.toPlainText(), self.noteText.toPlainText()))
        UI.refreshList(self.beforeUI)
        UI.changeLabel(self.beforeUI, "Note updated")
        self.close()
```

Obrázek 35 - Funkce saveClicked

2.4 Testovací skripty

Následuje kapitola zaměřená na vytvořené testovací skripty. Tyto skripty jsou stěžejní části této bakalářské práce, jelikož je tato práce zaměřená na automatické testování pomocí skriptů.

2.4.1 Testované aplikace

Kromě poznámkové aplikace vyvinuté pro tuto práci byly testovací skripty použity pro otestování již existujících aplikací. Pro tyto aplikace byly vytvořeny testovací skripty, aby byly ukázány další způsoby, kterými lze testovat aplikace. Také je zde příklad testovacího skriptu, který je také možné využít pro automatizaci úkonů.

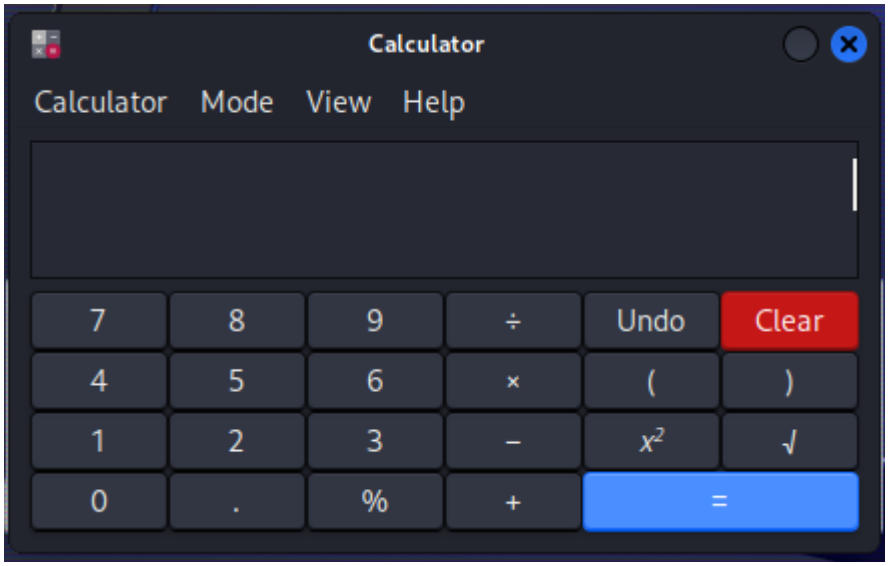
Tyto aplikace jsou:

- MATE Calculator – Kalkulátor, který je součástí instalace Kali Linux
- Mozilla Thunderbird – E-mailový klient

2.4.2 MATE Calculator

Tato aplikace byla zvolená pro počáteční testovací skripty. Na kalkulátoru lze jednoduše ukázat potenciál automatizovaných testů. GUI této aplikace je přehledné a vhodné pro automatizaci úkonů.

Vzhled této aplikace je na následujícím obrázku.

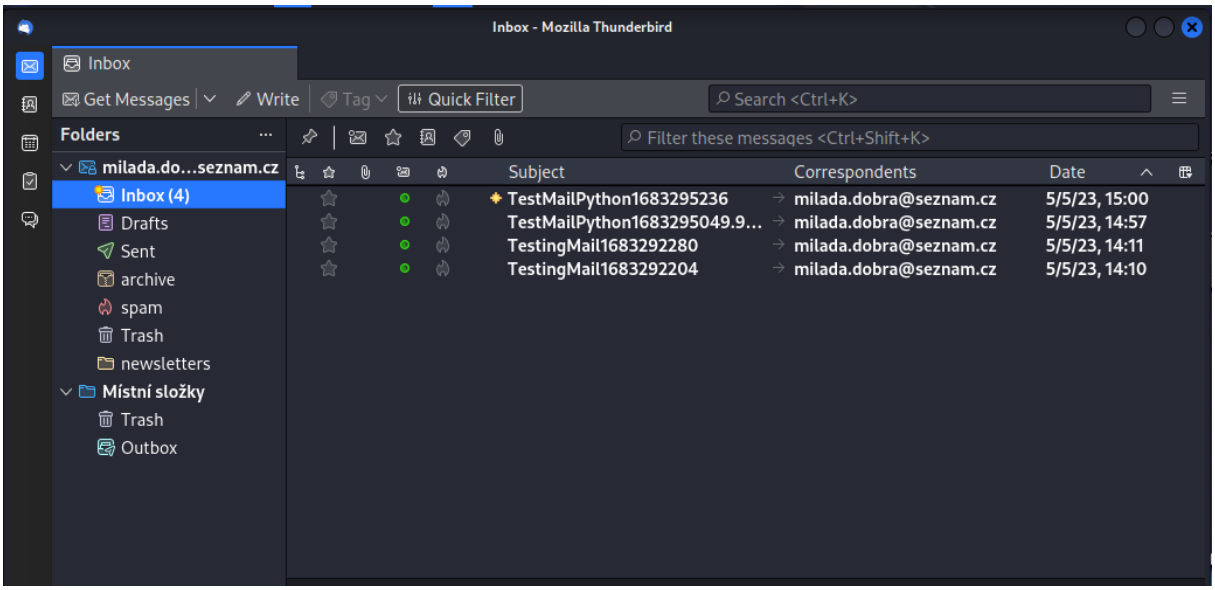


Obrázek 36 - Aplikace MATE Calculator

Myšlenkou testování této aplikace je vytvořit skript, který přijímá vstup uživatele ve formě matematické formule a následně v jednotlivých krocích tuto formuli vypočítá za pomoci GUI a následně tento výsledek porovná s výsledkem z terminálového příkazu. V tomto příkladě byl pro oba výsledky použit ten samý kalkulátor, ale je možné využívat jiný kalkulátor, kterému uživatel důvěřuje.

2.4.3 Mozilla Thunderbird

V této e-mailové aplikaci bylo testováno odesílání elektronických zpráv. Pro toto testování byl vytvořen nový e-mailový účet pro smyšlenou osobu pracující v neznámé firmě. Tento e-mail byl vytvořen na stránce seznam.cz a nazývá se milada.dobra@seznam.cz. Jelikož se tato práce krátce věnuje automatizaci práce, byl vytvořen skript, který je možný spouštět pravidelně pomocí nástroje crontab a hromadně odesílá pracovní e-maily lidem, kteří mají v daný den narozeniny.

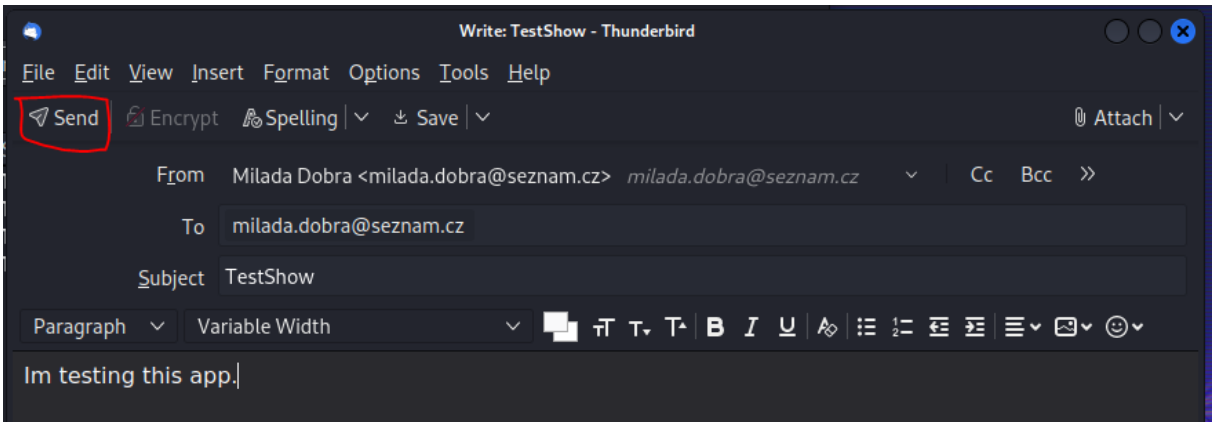


Obrázek 37 - Aplikace Mozilla Thunderbird

Prvním krokem tohoto testu bylo kliknutí tlačítka pro vytváření nové zprávy.

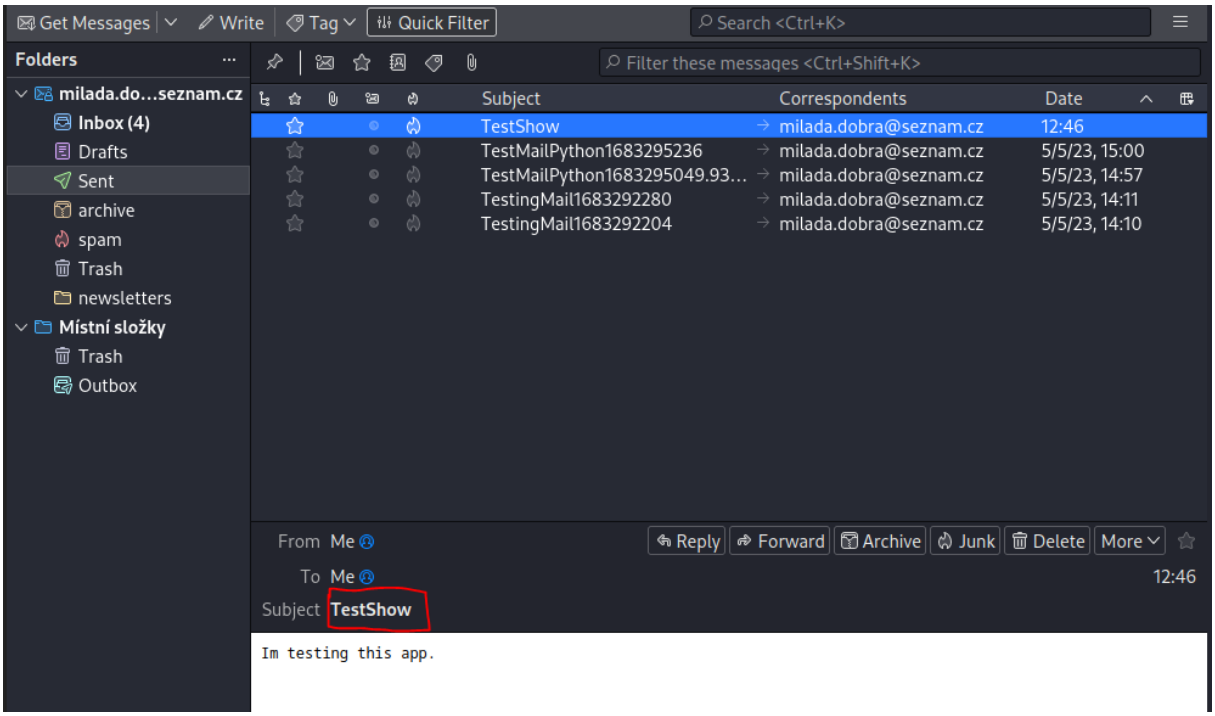
Obrázek 38 - Tlačítko Write

Dalším krokem bylo vyplnění testovací zprávy. Předmět této zprávy je při testování opatřen datumem ve formátu sekund, aby byl zajištěn jedinečný předmět pro každou testovací zprávu pro konečnou kontrolu. Po vyplnění patřičných polí je testovací zpráva odeslána.



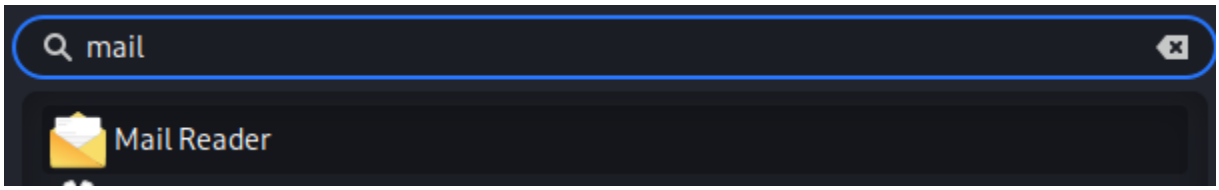
Obrázek 39 - Okno nové zprávy

Po odeslání zprávy se aplikace přepne na záložku odeslaných zpráv. Z těchto zpráv se vybere ta nejnovější, což je naše poslední testovací zpráva. Z této zprávy se zkopíruje předmět a porovná se s uloženým předmětem při odesílání zprávy. Pokud se tyto předměty shodují, je tento test úspěšný.



Obrázek 40 - Záložka odeslaných zpráv

Pro správnou funkci testovacího skriptu byl tento klient nastaven jako výchozí aplikace e-mailového klienta v systému Kali Linux.



Obrázek 41 - Mail Reader

2.4.4 Skript pro testování kalkulátoru v jazyce Bash

Pro zajištění správné funkcionality je každý bashový skript opatřen těmito prvními řádky.

```
1  #!/bin/bash
2  PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
3  export DISPLAY=:0.0
```

Obrázek 42 - Bash první řádky

Nejdříve je v tomto skriptu uloženo ID okna terminálu, ze kterého je tento skript spouštěn. Toto se vykoná využitím funkce getactivewindow nástroje xdotool. Po uložení tohoto ID je vyvolaná aplikace kalkulačky na pozadí a poté je proces skriptu pozastaven, aby bylo zajištěno, že se aplikace skutečně spustila. Aplikace kalkulačky se spouští na pozadí z důvodu pokračování skriptu. Pokud by se tato aplikace nespustila na pozadí, čekal by skript na ukončení aplikace před dalšími kroky, jelikož aplikace vrací ukončující kód pouze po uzavření aplikace.

```
5  terminalId=$(xdotool getactivewindow)
6
7  mate-calc &
8
9  sleep 2
```

Obrázek 43 - Spuštění mate-calc na pozadí

Následuje definice jednotlivých pixelů pro každý využívaný řádek a sloupec. Kombinace těchto souřadnic se používá při kliknutí na dané číslo, či speciální znak.

```
12  row1pixel=255
13  row2pixel=285
14  row3pixel=315
15  row4pixel=345
16
17  col1pixel=142
18  col2pixel=213
19  col3pixel=285
20  col4pixel=353
21  col5pixel=421
22  col6pixel=491
23
24  calcxpixel=147
25  calcypixel=150
26  copyypixel=169
```

Obrázek 44 - Definice souřadnic

Dále se definuje matematická formule pro zpracování. Uživatel si může specifikovat svoji formuli zadáním této formule jako argument při volání testovacího skriptu. Pokud si ovšem uživatel tuto formuli nespecifikoval, je skriptu přiřazena výchozí formule, se kterou skript pracuje.

```
30  formula="nothing yet"
31
32  if [ $# -eq 0 ]
33      then
34          formula="((2+5)*(154-133))/20"
35      elif [ $# -eq 1 ]
36          then
37          formula=$1
38      else
39          echo "Too many arguments!"
40  fi
```

Obrázek 45 - Definice matematické formule

Dále se zjistí ID okna kalkulátoru a uloží do proměnné. Poté se oknu nastaví vlastnost, která zajišťuje, že toto okno bude stále na vrchu všech ostatních oken, aby skript neztratil plochu pro klikání. Poté je okno přesunuto na specifikované místo na ploše.

```
42  windowId=$(xdotool search --name Calculator)
43  sleep 0.2
44  xdotool search --name "Calculator" windowactivate --sync key --clearmodifiers "ctrl+super+Up"
45  xdotool windowmove $windowId 100 100
```

Obrázek 46 - Přesunutí okna na vrchol

Jelikož je znak „*“ speciálním znakem jazyka Bash, je nutné tento znak zaměnit za běžný použitelný znak. V tomto případě došlo k záměně tohoto znaku na písmeno x. Pokud by k této záměně nedošlo, skript by nebyl schopný dokončit své úkony kvůli objevujícím se chybám v této formuli.

```
49    formula=$(echo $formula | tr '*' 'x')
```

Obrázek 47 - Nahrazení znaků

Dále následuje for cyklus, který prochází jednotlivé znaky matematické formule a provádí kliknutí na jednotlivá tlačítka kalkulátoru podle již definovaných souřadnic. Pokud se v tomto cyklu nalezne nepodporovaný znak, cyklus se ukončí a vypíše se daná chybová hláška. Jelikož je tento cyklus poměrně dlouhý, je zde ukázka prvních několika podmínek cyklu.

```
51  #Check for numbers and symbols
52  for (( i=0; i<${#formula}; i++)); do
53      sleep 0.5
54      checkstring=${formula:$i:1}
55      if [ $checkstring = "1" ]
56          then
57          xdotool mousemove $col1pixel $row3pixel && xdotool click 1
58      elif [ $checkstring = "0" ]
59          then
60          xdotool mousemove $col1pixel $row4pixel && xdotool click 1
61      elif [ $checkstring = "4" ]
62          then
63          xdotool mousemove $col1pixel $row2pixel && xdotool click 1
64      elif [ $checkstring = "7" ]
65          then
66          xdotool mousemove $col1pixel $row1pixel && xdotool click 1
67          #First column
68      elif [ $checkstring = "8" ]
69          then
70          xdotool mousemove $col2pixel $row1pixel && xdotool click 1
71      elif [ $checkstring = "5" ]
```

Obrázek 48 - For cyklus testu

Po vypočítání formule je stisknuté tlačítko „=“ a pomocí GUI aplikace je výsledek formule zkopírován do schránky systému.

```
123  xdotool mousemove $col5pixel $row4pixel && xdotool click 1
124
125  sleep 0.2
126
127  xdotool mousemove $calcxpixel $calcypixel && xdotool click 1
128
129  sleep 0.2
130
131  xdotool mousemove $calcxpixel $copyypixel && xdotool click 1
```

Obrázek 49 - Kopírování výsledku

Po zkopírování výsledku se tento výsledek uloží do proměnné pomocí nástroje xclip, který umožnuje získání hodnoty z kopírovací schránky systému. Po uložení se opět zamění násobící znaky zpět na původní a pomocí terminálové aplikace mate-calc je vypočítaná tato formule zvlášť.

```
135  guiResult=$(xclip -selection c -o)
136
137  formula=$(echo $formula | tr 'x' '*')
138  terminalResult=$(mate-calc -s $formula)
```

Obrázek 50 - Získání výsledku

Posledním krokem je samotná kontrola těchto dvou čísel. Pokud se tato čísla shodují, je uživateli oznámen správný výpočet. Pokud tomu tak není, je to také uživateli oznámeno. Tento skript se ukončuje stiskem klávesy Enter.

```
140  if [ $guiResult = $terminalResult ]
141  then
142      echo "$guiResult is indeed the same as $terminalResult"
143      echo "Test OK"
144  else
145      echo "Something went wrong, the numbers are not the same"
146  fi
147
148  xdotool windowfocus $terminalId
149
150  echo "Press enter to finish"
151  read ending
152
153  pkill -x mate-calc
```

Obrázek 51 - Kontrola čísel

2.4.5 Skript pro testování kalkulátoru v jazyce Python

Tento skript také začíná definicí jednotlivých souřadnic tlačítek kalkulátoru. Pro ojedinělé znaky jsou vytvořena pole dvou hodnot x a y souřadnic. Tato pole jsou užitečná, jelikož funkce knihovny PyAutoGUI podporují předávání parametrů pomocí pole hodnot.

```
col1 = 787
col2 = 857
col3 = 921
col4 = 993

row1 = 577
row2 = 606
row3 = 636
row4 = 671


leftBracket = [1063, 606]
rightBracket = [1133, 606]
equals = [1099, 667]
```

Obrázek 52 - Definice souřadnic

Obdobným způsobem je definována výchozí matematická formule, pokud uživatel nezadá při výzvě žádnou formuli. Po definici matematické formule je tento příklad vypočítán. Toto je zajištěno využitím pythonové funkce eval, která vypočítá formuli v textovém formátu. Funkce eval ovšem může vracet chyby při vypočítávání, pokud dostane nepatřičné znaky. Z tohoto důvodu je tento výpočet chráněn blokem try-except.

```
defaultFormula = "((2+5)*(154-133))/20"
formula = input("Enter the formula: ")
formulaCalculated = 0
if formula == "":
    formula = defaultFormula
    formulaCalculated = eval(formula)
else:
    try:
        formulaCalculated = eval(formula)
    except:
        print("This is not a valid mathematical formula!")
        exit(9)
```

Obrázek 53 - Definice matematické formule

Následuje spuštění testované aplikace. Po zpuštění je okno opět přesunuto na vrchol ostatních. Tentokrát je toto zajištěno klávesovými zkratkami.

```
pg.click(17, 17)
pg.write("calculator")
time.sleep(0.5)
pg.press("enter")
time.sleep(1)
pg.hotkey("alt", "space")
time.sleep(0.2)
pg.press("t")
```

Obrázek 54 - Spuštění kalkulátoru

Dále následuje velmi podobný for cyklus, který opět prochází zadanou matematickou formuli a kliká na tlačítka pomocí funkce click z knihovny PyAutoGUI.

```
for char in formula:
    time.sleep(0.5)
    if char == "7":
        pg.click(col1, row1)
    elif char == "4":
        pg.click(col1, row2)
    elif char == "1":
        pg.click(col1, row3)
    elif char == "0":
        pg.click(col1, row4)
    elif char == "8":
        pg.click(col2, row1)
    elif char == "5":
```

Obrázek 55 - Hlavní cyklus testu

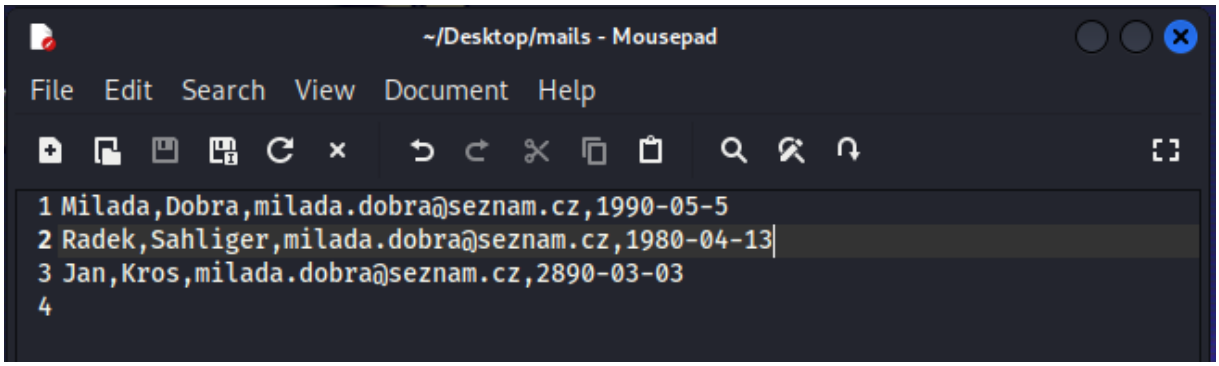
Po vypočítání příkladu se výsledek kalkulátoru zkopíruje klávesovou zkratkou. Dále se tento výsledek uloží do proměnné. Tento přenos zajišťuje knihovna pyperclip, která je obdobou nástroje xclip a umožňuje získávání kopírovaných hodnot. Po zkopírování jsou hodnoty opět porovnány a výsledek vypsán.

```
pg.click(equals)
time.sleep(0.2)
pg.hotkey("ctrl", "c")
calculated = pc.paste()
print("Internal: ", formulaCalculated, " Calculator: ", calculated)
if float(formulaCalculated) == float(calculated):
    print("Test Ok: Calculated value is correct")
else:
    print("Test Error: Calculated value is not correct")
```

Obrázek 56 - Porovnání výsledků

2.4.6 Skript pro testování aplikace Mozilla Thunderbird v jazyce Bash

Tento skript je pojatý jako ukázkový příklad pro vytvořený automatizačního skriptu pro hromadné odesílání například narozeninových přání ostatním pracovníkům. K tomuto skriptu patří soubor, ve kterém jsou uloženy informace o ostatních zaměstnancích. Za účelem testování je vytvořen ukázkový soubor, kde jsou jednotlivé údaje odděleny čárkami a každá osoba se nachází na samostatném řádku souboru.



Obrázek 57 - Soubor s údaji

Na začátku skriptu je opět definice potřebných x a y souřadnic jednotlivých tlačítek e-mailového klienta a ID terminálového okna.

```
8   mailX=103
9   mailY=105
10
11  newMailX=241
12  newMailY=111
13
14  recipientX=422
15  recipientY=191
16
17  subjectY=228
18
19  writeY=334
20
21  toprightX=0
22  toprightY=35
23
24  sendMailX=55
25  sendMailY=113
26
27  sentX=146
28  sentY=247
29
30  sentMailX=515
31  sentMailY=202
32
33  sentSubjectX=399
34  sentSubjectY=594
35
36  terminalId=$(xdotool getactivewindow)
37
```

Obrázek 58 - Definice souřadnic

Následuje spuštění e-mailového klienta a po vyčkání se uloží ID jeho okna.

```
41  xdotool mousemove 0 0 && xdotool click 1
42  sleep 0.2
43  xdotool type "mail"
44  sleep 0.2
45  xdotool mousemove $mailX $mailY && xdotool click 1
46  sleep 10
47  mailId=$(xdotool search --name "Mozilla Thunderbird")
```

Obrázek 59 - Spuštění mailového klienta

Poté dochází k výčtu souboru s uloženými údaji pracovníků. Soubor je rozřazen na jednotlivé řádky a poté se tyto řádky dále zpracovávají.

```
49  input="/home/kali/Desktop/mails"
50  while IFS= read -r line
51  do
52      IN=$line
53
54  IFS="," read -a arrIN <<< $IN
```

Obrázek 60 - Načtení údajů

Jednotlivé údaje člověka se uloží do proměnné typu pole. Údaj o datumu narození je nutné převést na typ date, aby bylo možné jej porovnávat s aktuálním datumem systému. Pokud se datum v prozkoumávaném řádku shoduje se systémovým datumem, pokračuje skript se sepsáním gratulační zprávy. Pro reálné využití by bylo vhodné zajistit vypisování jména recipienta a také lepší formulace zprávy. Pro testovací účely ovšem velmi hrubá zpráva postačí, jelikož se jedná pouze o ukázku.

```
56  mail=${arrIN[2]}
57  date=${arrIN[3]}
58  dateD=$(date -d "$date" "+%d.%m")
59  now=$(date "+%d.%m")
60
61  if [ $now = $dateD ]
62  then
63      echo "send mail to $mail"
64      let "mailNo++"
65
66      if [ -e "$mailId" ]
67          then
68              echo "Error: Mail window not found!"
69              exit 404
70      fi
```

Obrázek 61 – Kontrola údajů

Po zjištění správné e-mailové adresy se pokračuje otevřením nového okna aplikace. ID tohoto okna se opět pro manipulaci uloží do proměnné.

```
72    xdotool windowmove $mailId $toprightX $toprightY
73    sleep 0.2
74    xdotool mousemove $newMailX $newMailY && xdotool click 1
75    sleep 2
76    newmailId=$(xdotool search --name "Write")
```

Obrázek 62 - Otevření okna nového mailu

Následuje samotný zápis nové zprávy. Pro testování odeslání správné zprávy se vytvoří text předmětu, který obsahuje specifický údaj, který se skládá ze sekund aktuálního systémového datumu. Tímto se zajistí jedinečnost testovacího textu. Po zapsání zprávy a náležitých údajů se zpráva odesílá.

```
93    xdotool mousemove $recipientX $recipientY && xdotool click 1
94    xdotool type "$mail"
95    xdotool mousemove $recipientX $subjectY && xdotool click 1
96    xdotool type "TestingMail$sendTime"
97    xdotool mousemove $recipientX $writeY && xdotool click 1
98    xdotool type "This is just a test mail for your birthday. Congratulations"
99    xdotool mousemove $sendMailX $sendMailY && xdotool click 1
```

Obrázek 63 - Psaní zprávy

Po vyčkání na odeslání zprávy se skript opět přesune do hlavního okna aplikace. Z jednotlivých záložek klienta se vybere záložka odeslaných zpráv. Klient je nastaven tak, aby se odeslané zprávy zobrazovaly od nejnovějších po nejstarší. Z výpisu e-mailů se vybere první čili nejnovější zpráva. Z detailu zprávy se zkopíruje předmět zprávy a porovná se s uloženým textem předmětu, pokud se tyto údaje shodují, je test proveden úspěšně. Pokud se neshodují, nastala v některém procesu chyba.

```
103   xdotool mousemove $sentX $sentY && xdotool click 1
104   sleep 2
105   xdotool mousemove $sentMailX $sentMailY && xdotool click 1
106   sleep 0.2
107   xdotool mousemove $sentSubjectX $sentSubjectY && xdotool click 1 && xdotool click 1
108
109   testString="TestingMail$sendTime"
110
111   copyString=$(xclip -selection c -o)
112
113   if [ $testString = $copyString ]
114     then
115       echo "Mail sent succesfully"
116       let "testingOk++"
117     else
118       echo "Error: mail not found!"
119   fi
```

Obrázek 64 - Kontrola zprávy

Na úplný konec tento skript vypíše konečnou statistiku. Za běhu skriptu se aktualizovaly hodnoty proměnných mailNo a testingOk. Proměnná mailNo určuje celkový počet zpráv, které se mají odeslat. Proměnná testingOk určuje počet skutečně odeslaných zpráv. Pokud se tyto proměnné na konci skriptu shodují, je vše v pořádku.

```
124 if [ $testingOk = $mailNo ]
125   then
126     echo "Test Ok: $testingOk mail(s) were sent succesfully."
127   else
128     echo "Test Failed: $mailNo mail(s) were supposed to be sent, but $testingOk mail(s) were sent succesfully."
129 fi
```

Obrázek 65 - Celkový výpis testu

Jelikož se předpokládá, že tento skript bude spouštěn pravidelně každý den v určitou hodinu, je vhodné výpisy tohoto skriptu ukládat do logovacího souboru. Toto lze zajistit nastavením příslušného spouštění skriptu nástrojem crontab. Nástroj crontab bude pravidelně spouštět tento skript a zapisovat výsledky testování do logovacího souboru.

Takto může vypadat nastavení crontabu.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

* 9 * * * /home/kali/Desktop/sendMail.sh >> /home/kali/Desktop/mailLog.log

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

Obrázek 66 - Crontab soubor

2.4.7 Skript pro testování aplikace Mozilla Thunderbird v jazyce Python

Tento skript také testuje funkci e-mailového klienta, ale není zaměřen na automatické spouštění.

Nejdříve je opět definována řada proměnných typu pole pro jednotlivé souřadnice specifických tlačítek aplikace.

```
topLeft = [0, 35]
mail = [103, 105]
newMail = [241, 111]
recipient = [422, 191]
subject = [422, 228]
writing = [422, 334]
sendMail = [55, 113]
sent = [146, 247]
sentMail = [515, 202]
sentSubject = [399, 594]
```

Obrázek 67 - Definice souřadnic

Jelikož tento skript nebude kontrolovat soubor, kde jsou uložené údaje o lidech, stačí definovat na jakou adresu se bude zpráva odesílat, předmět zprávy opatřený kontrolním datumem v sekundách a text zprávy.

```
mailName = "milada.dobra@seznam.cz"
mailSubject = "TestMailPython" + str(round(datetime.datetime.now().timestamp()))
mailContent = "This is a test mail from a Python script, Congratulations!"
```

Obrázek 68 - Definice součástí zprávy

Spuštění e-mailového klienta vypadá v jazyce Python následovně.

```
pg.click(17, 17)
pg.write("mail")
time.sleep(0.5)
pg.press("enter")
time.sleep(10)
```

Obrázek 69 - Spuštění mailového klienta

Tato aplikace při spuštění v některých případech nedrží stále stejnou polohu na ploše systému. Aby byly zachovány správné souřadnice ovládacích prvků, bylo nutné použít další pomocnou knihovnu. Touto knihovnou je libxdo, která umožnuje manipulaci s okny aplikace. Tato

knihovna je založena na funkcích nástroje xdotool. Po získání okna a posunutí daného okna se otevře okno nové zprávy.

```
XD = Xdo()
mailID = XD.get_active_window()
XD.move_window(mailID, topLeft[0], topLeft[1])
XD.focus_window(mailID)

pg.click(newMail)
time.sleep(2)
newMailID = XD.get_active_window()
XD.move_window(newMailID, topLeft[0], topLeft[1])
XD.focus_window(newMailID)
```

Obrázek 70 - Posunování oken

Následuje zapsání potřebných údajů zprávy a odeslání. Poté se několik sekund čeká na samotné odeslání.

```
pg.click(recipient)
pg.write(mailName)
pg.click(subject)
pg.write(mailSubject)
pg.click(writing)
pg.write(mailContent)
pg.click(sendMail)

time.sleep(5)
```

Obrázek 71 - Zapsání zprávy

Po odeslání zprávy se skript opět přesune do záložky odeslaných zpráv a zkontroluje předmět naposledy odeslané zprávy. Tento předmět se zkopíruje pomocí klávesové zkratky kopírování a poté se uloží do proměnné využitím funkce paste z knihovny pyperclip. Pokud se tyto předměty shodují, je test v pořádku.

```
XD.focus_window(mailID)
pg.click(sent)
time.sleep(2)
pg.click(sentMail)
time.sleep(0.2)
pg.doubleClick(sentSubject)
pg.hotkey("ctrl", "c")

testSubject = str(pc.paste())

if testSubject == mailSubject:
    print("Test Ok: Mail sent successfully")
else:
    print("Test Error: Sending mail failed")
```

Obrázek 72 - Kontrola odeslání zprávy

2.4.8 Skript pro testování poznámkové aplikace v jazyce Python

Účel tohoto skriptu je testování jednotlivých funkcí poznámkové aplikace vytvořené pro tuto bakalářskou práci. Testování této aplikace je provedeno ve dvou krocích. Prvním krokem je tento test v jazyce Python. V tomto testu se otestují funkce aplikace a jednotlivé poznámky a seznam poznámek z pohledu proměnných. Je nutné tento test spouštět jako první, jelikož je druhý test v jazyce Bash závislý na výsledku z tohoto testového skriptu.

Jelikož je tento test v jazyce Python, který podporuje vytváření tříd a jejich instancí, je pro tento test vytvořeno několik pomocných funkcí, které jsou volány z hlavního toku testu. První touto funkcí je newNote. Tato funkce zajišťuje otevření okna nové poznámky a následné vyplnění definovaných parametrů poznámky

```
def newNote(self, noteName, noteContent):
    pg.click(1091, 761)
    time.sleep(0.5)
    pg.click(834, 259)
    time.sleep(0.5)
    pg.write(noteName)
    pg.click(834, 412)
    pg.write(noteContent)
    time.sleep(0.5)
    pg.click(1104, 297)
    time.sleep(0.5)
```

Obrázek 73 - Funkce newNote

Další pomocnou funkcí je updateNote. Tato funkce zajišťuje otevření existující poznámky pro editaci této poznámky a následnou úpravu existující poznámky novými údaji.

```
def updateNote(self, noteContent):
    self.chooseNote(0)
    pg.click(990, 757)
    time.sleep(0.5)
    pg.click(852, 343)
    time.sleep(0.5)
    pg.hotkey("ctrl", "a")
    time.sleep(0.5)
    pg.write(noteContent)
    time.sleep(0.5)
    pg.click(1104, 297)
    time.sleep(0.5)
```

Obrázek 74 - Funkce updateNote

Funkce deleteNote zajišťuje smazání poznámky na požadované pozici, kterou předáváme argumentem do funkce.

```
def deleteNote(self, number):
    self.chooseNote(number)
    pg.click(815, 757)
    time.sleep(0.5)
```

Obrázek 75 - Funkce deleteNote

Poslední pomocnou funkcí je chooseNote, která vybírá poznámku na specifikované pozici pomocí argumentu.

```
def chooseNote(self, number):
    pg.click(946, (270 + (number*30)))
    time.sleep(0.5)
```

Obrázek 76 - Funkce chooseNote

Hlavní funkcí tohoto testu je funkce runTest, která má za úkol samotné testování aplikaci a následné porovnávání výsledných poznámek s předem vytvořenými poznámkami.

Nejprve se vytvoří tři testovací poznámky a jejich instance se uloží do proměnných.

```
def runTest(self, noteList: NoteList):
    time.sleep(0.5)
    testNote1: Note = Note("TestNote1", "This is the first test note.")
    testNote2: Note = Note("TestNote2", "This is the second test note.")
    testNote3: Note = Note("TestNote3", "This is the third test note.")
```

Obrázek 77 - Vytvoření nových poznámek

Dále se pomocí funkce newNote tyto tři poznámky zavedou do seznamu poznámek pomocí ovládacích prvků aplikace.

```
self.newNote(testNote1.name, testNote1.content)
self.newNote(testNote2.name, testNote2.content)
self.newNote(testNote3.name, testNote3.content)
```

Obrázek 78 - Přidání nových poznámek

Dalším krokem je otevření souboru NoteTestLog.log. Poté se porovnávají hashové hodnoty poznámek uložených v seznamu a poznámek předem vytvořených. Pokud se tyto hodnoty shodují, je tento výsledek vypsán do logovacího souboru.

```
with open("NoteTestLog.log", "w") as file:
    if noteList.get(testNote1).noteHash == testNote1.noteHash:
        file.write("Test Ok: Note 1 was added\n")
    else:
        file.write("Test Error: Note 1 not found!\n")

    if noteList.get(testNote2).noteHash == testNote2.noteHash:
        file.write("Test Ok: Note 2 was added\n")
    else:
        file.write("Test Error: Note 2 not found!\n")

    if noteList.get(testNote3).noteHash == testNote3.noteHash:
        file.write("Test Ok: Note 3 was added\n")
    else:
        file.write("Test Error: Note 3 not found!\n")
    file.close()
```

Obrázek 79 - Kontrola poznámek

Dále se vytvoří nová poznámka, která se nazývá stejně jako první testovací poznámka, ale má jiný text. Pomocí této poznámky se upraví již existující poznámka funkcí updateNote. Poté s opět testuje hodnota hashe poznámky a připadající výsledek se zapíše do logovacího souboru.

```
testNote1Changed: Note = Note("TestNote1", "This note was changed")

self.updateNote(testNote1Changed.content)

with open("NoteTestLog.log", "a") as file:
    if noteList.get(testNote1Changed).noteHash == testNote1Changed.noteHash:
        file.write("Test Ok: Note was updated\n")
    else:
        file.write("Test Error: Note was not updated\n")
    file.close()
```

Obrázek 80 - Kontrola editace

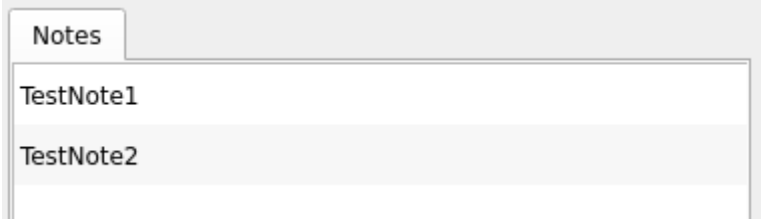
Jako poslední krok se provede vymazání poslední poznámky ze seznamu pomocí funkce deleteNote. Poté skript testuje seznam poznámek na to, zda najde předem smazanou poznámku. Pokud poznámku nenajde, tak je test v pořádku a zapíše výsledek.

```
self.deleteNote(2)

with open("NoteTestLog.log", "a") as file:
    if noteList.get(testNote3) is None:
        file.write("Test Ok: Note 3 was deleted\n")
    else:
        file.write("Test Error: Note 3 was not deleted")
    file.close()
```

Obrázek 81 - Kontrola mazání

Tímto končí první část testu a výsledný seznam by měl vypadat následovně.



Obrázek 82 - Výsledný seznam

2.4.9 Skript pro testování poznámkové aplikace v jazyce Bash

Tento skript je druhou částí testovacího procesu pro poznámkovou aplikaci. Tento test předpokládá, že výsledek předchozího testu je nezměněný a bude dále pracovat s těmito poznámkami. Tato část provádí podobné úkony jako první část testu. Jelikož se ale z prostředí bashového skriptu nedostaneme na proměnné pythonového programu, je tento test zaměřen na perzistenci aplikace. Tento skript provede dané úpravy poznámek a poté tento seznam uloží do souboru a zkontroluje, zda se tyto poznámky skutečně správně uložily do souboru.

Prvním krokem je opět definice souřadnic pro kliknutí myši na ovládací prvek aplikace.

```
notesX=929
notesY=270

newNoteX=1090
newNoteY=756

selectNoteX=991
selectNoteY=757

deleteNoteX=819
deleteNoteY=757

nameBoxX=931
nameBoxY=261

saveNoteX=1103
saveNoteY=295

contentTextX=925
contentTextY=402
```

Obrázek 83 - Definice souřadnic

Dále se vytvoří proměnná uchovávající text, kterým se upraví existující poznámka a název s textem nové poznámky, která bude do seznamu přidána.

```
content1="This note was changed in Bash."
bashName="BashTestNote"
bashContent="This is a bash test note."
```

Obrázek 84 - Definice nové a editované poznámky

Nejdříve skript vybere existující poznámku a poté ji upraví novým textem, který byl definován před zahájením tohoto kroku.

```
34  sleep 0.2
35  notesID=$(xdotool search --name "MyNotesApp")
36  xdotool windowfocus $notesID
37
38  xdotool mousemove $notesX $notesY && xdotool click 1
39  xdotool mousemove $selectNoteX $selectNoteY && xdotool click 1
40  sleep 0.2
41  xdotool mousemove $contentTextX $contentTextY && xdotool click 1
42
43  sleep 0.2
44  noteID=$(xdotool search --name "MyNotesApp - Note Edit")
45  xdotool windowfocus $noteID
46  xdotool key ctrl+a && xdotool type "$content1"
47  xdotool mousemove $saveNoteX $saveNoteY && xdotool click 1
48  sleep 0.2
```

Obrázek 85 - Úprava poznámky

Dále skript vytvoří novou poznámku a uloží tuto poznámku do seznamu aplikace.

```
50  xdotool mousemove $newNoteX $newNoteY && xdotool click 1
51  sleep 0.2
52  noteID=$(xdotool search --name "MyNotesApp - Note Edit")
53  xdotool windowfocus $noteID
54  xdotool mousemove $nameBoxX $nameBoxY && xdotool click 1
55  xdotool type "$bashName"
56  xdotool mousemove $contentTextX $contentTextY && xdotool click 1
57  xdotool type "$bashContent"
58  xdotool mousemove $saveNoteX $saveNoteY && xdotool click 1
59  sleep 0.2
```

Obrázek 86 - Přidání poznámky

Dalším aplikačním krokem je vymazání prostřední poznámky ze seznamu aplikace. Poté se vynutí uložení stávajícího seznamu do souboru klávesovou zkratkou.

```
61  xdotool windowfocus $notesID
62  xdotool mousemove $notesX $((notesY + 30)) && xdotool click 1
63  xdotool mousemove $deleteNoteX $deleteNoteY && xdotool click 1
64  sleep 0.2
65  xdotool key ctrl+s
```

Obrázek 87 - Mazání poznámky

Poté se ze souboru, do kterého aplikace ukládá poznámky, vyčtou všechny řádky a názvy s textem poznámek se uloží do pole.

```
67  i=0
68  file="noteFile.notes"
69  while read -r line
70      do
71      if [ "$line" = "----------" ]
72          then
73              continue
74      elif [ "$line" = "-----" ]
75          then
76              continue
77      else
78          fileArray[i]=$line
79          i=$((i+1))
80      fi
81  done < $file
```

Obrázek 88 - Nahrání souboru do pole

Posledními kroky je kontrola, zda se v souboru skutečně nachází 4 řádky údajů o dvou poznámkách. Poté se zkontrolují jednotlivé údaje, zda souhlasí s přidanými a upravovanými poznámkami.

```
83  if [ ${#fileArray[@]} -eq 4 ]
84      then
85          echo "Test Ok: There are two notes present" >> BashTestLog.log
86  fi
87
88  if [ "${fileArray[0]}" = "TestNote1" ]
89      then
90          if [ "${fileArray[1]}" = "This note was changed in Bash." ]
91              then
92                  if [ "${fileArray[2]}" = "BashTestNote" ]
93                      then
94                          if [ "${fileArray[3]}" = "This is a bash test note." ]
95                              then
96                              echo "Test Ok: Only the expected notes are present" >> BashTestLog.log
97                          fi
98                  fi
99          fi
100
101 fi
```

Obrázek 89 - Kontrola souboru

Tímto je tento test ukončen a patřičné výsledky jsou zapsány do daných logovacích souborů.

ZÁVĚR

Tato práce byla pojata ve větším záběru tématu automatizace kroků v GUI aplikacích. Kromě testování vyvíjené aplikace jsem také ukázal příklad na již hotových a již zkoušených aplikacích. Také jsem se částečně zaměřil na využití automatizačních skriptů k vytvoření plánovaných úkonů, jako je například hromadné odesílání narozeninových e-mailů v pracovním prostředí.

Tyto jednotlivé úkony ukazují, že je velmi výhodné vytváření automatizačních skriptů, jelikož si programátor, či běžný uživatel ušetří drahocenný čas, který může produktivně strávit něčím jiným.

Oba způsoby testování mají své výhody a nevýhody. Výběr jedné z variant je již na subjektivním názoru uživatele či programátora, který se hodlá do testování ponořit.

Hlavní nevýhodou bashového testování může být neznalost jazyka Bash, který má svou ojedinělou syntaxi. Zároveň jsou ale skripty tohoto jazyka propojené přímo k nástroji systému. Díky této vlastnosti lze vytvářet jednoduché skripty do jednoho spustitelného souboru bez nutnosti instalace IDE. Tyto skripty lze vytvářet jednoduše v jakémkoliv textovém editoru a posléze tyto skripty příkazem spouštět.

Výhodou pythonového testování je zajisté přehlednost pythonového kódu. Velmi velké množství začínajících programátorů si tento jazyk volí právě kvůli této vlastnosti. Jazyk Python obsahuje velké množství užitečných knihoven a funkcí, které usnadňují práci programátorovi. Nevýhodou knihovny PyAutoGUI je nesporně nemožnost práce s okny v určitých systémech. Tato knihovna příslušnými funkcemi sice disponuje, ovšem nelze tyto funkce využívat v linuxových systémech. Pro tuto funkcionalitu je nutné využít jiné knihovny.

Zkušenosti nabité touto bakalářskou prací plánuji využít zejména v pracovním prostředí, kde je tato automatizace velmi výhodná.

POUŽITÁ LITERATURA

[1] g0tm1k, „What is Kali Linux,“ © OffSec Services Limited 2023, [Online]. Available: https://www.kali.org/docs/introduction/what-is-kali-linux/. [Přístup získán 06 05 2023].

[2] Oracle, „Welcome to VirtualBox.org,“ Oracle, [Online]. Available: https://www.virtualbox.org/. [Přístup získán 06 05 2023].

[3] Microsoft, „Getting Started,“ [Online]. Available: https://code.visualstudio.com/docs. [Přístup získán 06 05 2023].

[4] A. Siobos, „Lekce 1- PyCharm - Úvod do vývojového prostředí,“ © 2023 itnetwork.cz, [Online]. Available: https://www.itnetwork.cz/python/pycharm-pokrocila-prace/pycharm-uvod-do-vyvojoveho-prostredi. [Přístup získán 06 05 2023].

[5] L. P. Ramos, „Qt Designer and Python: Build Your GUI Applications Faster,“ © 2012–2023 Real Python, [Online]. Available: https://realpython.com/qt-designer-python/. [Přístup získán 06 05 2023].

[6] D. Martinek, „Bourne Again SHell - BASH,“ [Online]. Available: http://www.fit.vutbr.cz/~martinek/gymnazium/bash.html. [Přístup získán 06 05 2023].

[7] P. n. s. Wikipedia, „Python,“ Creative Commons, [Online]. Available: https://cs.wikipedia.org/wiki/Python. [Přístup získán 06 05 2023].

[8] L. P. Ramos, „Python and PyQt: Build a GUI Desktop Calculator,“ [Online]. Available: https://realpython.com/python-pyqt-gui-calculator/. [Přístup získán 06 05 2023].