МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра економічної кібернетики та інженерії програмного забезпечення

ДО ЗАХИСТУ ДОПУЩЕНА
Зав.кафедри _____
д.е.н., доц. Левицький С.І.

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

РОЗРОБКА УПРАВЛІНСЬКОЇ БАЗИ З ОБСЛУГОВУВАННЯ КЛІЄНТІВ ЗА ДОГОВОРАМИ НА ПРИКЛАДІ АТ КБ «МЕТАБАНК»

Виконав
ст. гр. ІПЗ-227 _____ Морев С. О.

Керівник
Доц. _____ Кривенко В. В.

Запоріжжя
2022

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра економічної кібернетики та інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри д.е.н., доц.
Левицький С.І._____

“ 17 ” січня 2022 року

ЗАВДАННЯ НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

Студента гр. *ІПЗ-227*

Спеціальності: *121 - Інженерія програмного забезпечення*

Мореву Сергію Олександровичу

(прізвище, ім'я, по батькові)

1. Тема: *Розробка управлінської бази з обслуговування клієнтів за договорами на прикладі АТ КБ «МетаБанк»*

затверджена наказом по інституту: № 06.2-14-2 від 18 лютого 2022 року

2. Термін здачі студентом закінченої роботи: 18 червня 2022 року

3. Перелік питань, що підлягають розробці:

1. Зібрати літературу та документацію присвячену тематиці бакалаврської роботи;

2. Провести бесіду з керівництвом банку з приводу дослідження предметної області та доступом до корпоративної таємниці;

3. Розглянути та проаналізувати дані банківських аналогів ПП;

4. Розробити на мові програмування Python проєтну частину кодингу;

5. Виконати всі поставлені задачі: розробці, оформленні та представленні бакалаврської роботи;

6. Впровадити програмний продукт в дію роботи АТ КБ «МетаБанк»

7. Оформити результати у вигляді пояснювальної записки до відповідних ДСТУ норм бакалаврських робіт.

Дата видачі завдання: 17 січня 2022 року

4. Календарний графік

№ ЕТАПУ	Зміст	Термін виконання	Готовність по графіку (%), підпис керівника	Підпис керівника про повну готовність етапу, Дата
1	Формування теми бакалаврської дипломної роботи. Збір практичного матеріалу за темою	17.01.22-26.02.22		
2	І атестація. І розділ бакалаврської дипломної роботи	28.03.22-02.04.22		
3	ІІ атестація. ІІ розділ бакалаврської дипломної роботи	10.05.22-14.05.22		
4	ІІІ атестація ІІІ розділ бакалаврської дипломної роботи, висновки, додатки, реферат, перевірка програмою «Антиплагіат»	30.05.22-04.06.22		
5	Доопрацювання бакалаврської дипломної роботи, підготовка презентації, отримання відгуку керівника та рецензії	30.05.22-18.06.22		
6	Попередній захист бакалаврської дипломної роботи	14.06.22-18.06.22		
7	Подача бакалаврської дипломної роботи на кафедру	За 3 дні до захисту		
8	Захист бакалаврської дипломної роботи	20.06.22-25.06.22		

Керівник бакалаврської дипломної роботи_____ Кривенко В.В.

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання_____ Морев С.О.

(підпис студента) (прізвище та ініціали)

РЕФЕРАТ

Бакалаврська дипломна робота складається з: 108 сторінок, рисунків – 45, таблиць – 1, додатків – 1, .

Об’єкт дослідження: Програма для підписання документів.

Мета роботи: розробка нового і підтримка існуючого функціоналу, узгодженого з замовником.

Результатом роботи стала програма для підписання документів і відправки їх до банку.

При розробці системи використана мова програмування Python та PyQt разом з бібліотекою IIT користувач. Також збиральник проектів PyCharm, MongoDB-база даних.

PYTHON, MONGODB,EUSINGPYTHON,PYCHARM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Слово/словосполучення	Скорочення	Умова використання
База даних	БД	В тексті
structured query language	SQL	В тексті
JavaScript Object Notation	JSON	В тексті
Binary JavaScript Object Notation	BSON	В тексті
eXtensible Markup Language	XML	В тексті
Електронний цифрови́й підпис	ЕЦП	В тексті
Акредитований центр сертифікації ключів	АЦСК	В тексті
Інформаційно-дорадчий центр	ІДЦ	В тексті
Державна фіскальна служба	ДФС	В тексті
Управління Носіями Інормаційной Системи	УНІС	В тексті
Українські Спеціальні Системи	УСС	В тексті
Міністерстві внутрішніх справ	МВС	В тексті
Національний банк України	НБУ	В тексті
Uniform Resource Locator	URL	В тексті

HyperText Transfer Protocol	HTTP	В тексті
метод зчитування даних з сайту	GET	В тексті
General Public License	GPL	В тексті
HyperText Markup Language	HTML	В тексті
Simple Mail Transfer Protocol	SMTP	В тексті
Secure Sockets Layer	SSL	В тексті
Lightweight Directory Access Protocol	LDAP	В тексті
transport layer security	TLS	В тексті

ВСТУП

Фінансова система упродовж усієї історії людства займає значну роль в житті людини. З давніх давен люди обмінювалися товарами та послугами в будь-яких галузях. У Вавилоні протягом XVIII століття до нашої ери вже відбулися обміни між купцями та священиками, подібно до того, як у Римі були також економічні обміни, але більше між самими жителями, а ніж установами. Однак лише до епохи Відродження виникли банки, як ми їх знаємо сьогодні. Почали існувати родини банкірів, а також власні банківські центри, де здійснювали переміщення срібних та золотих монет. Все це надзвичайно збільшилося протягом XVIII-XIX століття, а також у XX-му столітті, коли розпочалося регулювання фінансів практично на всій нашій планеті, намагаючись встановити правила та основи для банківської діяльності, які ми бачимо сьогодні. Гроші розповсюджуються, їх стає більше, з’являється безліч нових установ, що контролюють та полегшують обіг грошей в суспільстві. Чудовою ідеєю є створення зручної та гнучкої системи управління фінансами, якими зараз являються банки [1].

Комерційні банки є найважливішою ланкою економічної системи, які забезпечують перерозподіл грошових потоків ринку капіталів. Адже сьогодні кожен може власноруч відкрити собі рахунок в банку, пересилати гроші та контролювати їх кількість.

Банк — це вид кредитної фінансової установи, основною метою якого є контроль та управління коштами за допомогою різних послуг, таких як зберігання великих сум грошей на депозитних рахунках, в банківських ячейках, проведення фінансових операцій або надання позик чи кредитів, та інше. Звичайна практика банку — збирати капітал у фізичних осіб або підприємств. У той же час банк функціонує як просто ще одна компанія і має власні кошти. Звичайно, також із

власною бізнес-ідеєю, коли справа стосується кредитних чи інших операцій у сфері фінансів. Все більше людей мають змогу виконувати різні операції та отримувати певні послуги від банків. Але, незважаючи на це, не кожен має змогу особисто долучитися до цього процесу по різним причинам. На сьогоднішній день ми маємо змогу задовольняти майже будь які потреби за допомогою різних додатків та Інтернету. Та для більш швидкої та зручної роботи банку потрібні не лише програмні продукти для самих клієнтів, а й для самих співробітників, а би забезпечити більш швидке обслуговування та роботу банку [2].

Тому метою моєї роботи була розробка програмного забезпечення який дозволить банківським працівникам підписувати документи електронним підписом та відправити в потрібне місце, цим самим ми отримуємо більш швидку та якісну роботу персоналу, а також економію часу, та відмову від паперової зайвої документації [3].

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ ДЛЯ БАКАЛАВРСЬКОЇ РОБОТИ

1.1. Пошук бази даних для дипломної роботи

Прості структури бази даних. Перший спосіб це зберігання даних в текстові формі. На сьогоднішній день цей метод застосовується в роботі з невеликим обсягам даних. Для цього типу даних використовується спеціальний синтаксис: крапка або кома з комою в csv-файлах, двокрапка або пробіл в * nix системах.

Недоліки: обмежений тип і рівень складності інформації, що зберігається; важко встановити зв'язки між компонентами даних; відсутність функцій паралелізму; цей метод застосовується для систем з невеликими потребами до читання і запису; використовуються для збереження конфігураційних даних; немає потреби в іншому програмному продукту. Приклади: /etc/passwdi /etc/fstab в * nix-системах csv-файли [4].

Ієрархічні бази даних. На відміну від текстових таблиць, в наступному типі БД з'являються зв'язки між об'єктами. В ієрархічних базах даних кожен запис має одного «батька». Це утворює деревоподібну структуру, в якій записи класифікуються за їхніми зв`язками з ланцюжком батьківських записів.

Наслідки: інформація створена у вигляді дерева з відносинами «предок-нащадок»; кожен запис може мати не більше одного з батьків; зв'язки між записами створеними у вигляді фізичних показників; не має змоги реалізувати відносини «багатьох до багатьох».

Мережеві бази даних розширюють функціональність ієрархічних: записи можуть мати більше одного батька. А значить, можна моделювати складні відносини. Наслідки: мережеві бази даних подаються не деревом, а загальним графом обмежені тими ж шаблонами доступу, що ієрархічні. Реляційні бази даних

- найстаріший тип широко використовуваних БД загального призначення. Дані та зв'язки між даними організовані за допомогою таблиць. Кожен стовпик у таблиці має ім'я і тип. Кожен рядок представляє окремий запис або елемент даних в таблиці, який містить значення для кожного з стовпців [5].

Наслідки: поле в таблиці, називається зовнішнім ключем, може містити в собі посилання на стовбці в інших таблицях, що дозволяє їх з'єднувати; високоорганізована структура і гнучкість робить реляційні БД потужними і такими, що адаптуються до різних типів даних; для доступу до даних використовується мова структурованих запитів (SQL); надійний вибір для багатьох додатків. NoSQL - група типів БД, що пропонують підходи, відмінні від стандартного реляційного шаблону. Говорячи NoSQL, мають на увазі або "не-SQL», або «не тільки SQL», щоб уточнити, що іноді допускається SQL-подібний запит. 5. Бази даних «ключ-значення» У базах даних «ключ-значення» для зберігання інформації ви надаєте ключ та об'єкт даних, який потрібно зберегти. Наприклад, JSON-об'єкт, зображення або текст. Щоб виконати запит, відправляєте ключ і отримуєте blob-об'єкт . Наслідки: сховища забезпечують швидкий доступ з незначними витратами; часто зберігають дані конфігурацій і інформацію про стан даних, представлених списками або хешем; немає жорсткої схеми відносин між даними, тому в таких БД часто зберігають одночасно різні типи даних; розробник відповідає за визначення схеми найменування ключів і за те, щоб значення мало відповідний тип / формат.

Документи бази даних (також документ-орієнтовані БД або сховища документів), спільно використовують базову семантику доступу і пошуку сховищ ключів і значень. Такі БД також застосовують ключ для унікальної ідентифікації даних. Різниця між сховищами «ключ-значення» і документ БД полягає в тому, що замість зберігання blob-об'єктів, документ-орієнтовані бази зберігають дані в структурованих форматах - JSON, BSON або XML [6].

Наслідки:

- База даних не виділяє окремий формат або схему;
- Кожен документ може мати свою внутрішню структуру;
- Документн БД є хорошим вибором для швидкої розробки;
- В будь-який момент можна змінювати властивості даних, не змінюючи структуру або самі дані.

Приклади: MongoDB RethinkDB. Графові бази даних Замість зіставлення зв'язків з таблицями і зовнішніми ключами, графові бази даних встановлюють зв'язки, використовуючи вузли, ребра і властивості. Графові бази представляють дані у вигляді окремих вузлів, які можуть мати будь-яку кількість пов'язаних з ними властивостей.

1.2. Вибір мови програмування

Останнім часом дуже багато мов програмування які розвиваються швидкими темпами в сфері інформаційних технологій. На Україні популярні мови програмування: Java, javaScript, React, Python, PHP. Ці мови програмування мають свої специфічні цінності в певних галузях промисловості. Наприклад для розробки сайтів за часту використовують мови: react, vue, javascript. Які забезпечують швидку і якісну розробку додатків але вони мають свої недоліки а саме потрібно уйма часу щоб освоїти їх, знання основ програмування, вміння швидко шукати вірне рішення в купі рішень.

Люди які хочуть стати професійним програмістом починають вчити з менш складних мов програмування PHP, Python, Ruby, тому що наприклад Python не має строгої прив’язки до типу даних такі як int, booleon, string і так далі. Саме із за цього їх вибирають більшість людей що вони прости в освоєні синтаксису програмування. Для дипломної роботи була вибрана мова програмування python так як вона більш підходить до мого проекту [7].

Python не має швидкості виконання коду, оскільки це не скомпільована мова. Python спочатку компілює свій код у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків використання Python повільніше, ніж використання мов, таких як C. Проте з урахуванням обчислювальної потужності сучасних комп’ютерів швидкість розробки важливіша за швидкість виконання для більшості програм, а програми на Python часто пишуться швидше. Крім того, Python легко розширюється за допомогою модулів, написаних на C або C++. Такі модулі можна використовувати для виконання частин програм, які дають велике навантаження на процесор.

У багатьох мовах програмування існує концепція бібліотеки або блоку коду, призначеного для повторного використання. У бібліотеках Python також є великий вибір кодових баз. Базовий дистрибутив Python включає повністю функціональну стандартну бібліотеку, яка дозволяє виконувати різні завдання без встановлення додаткових бібліотек. Те, що вважається стандартною бібліотекою в Python, складається з кількох компонентів, включаючи вбудовані типи даних і константи, які можна використовувати без імпорту команд, таких як числа та списки. Будь-яка програма на Python має базовий набір вбудованих функцій, серед яких print(), input(), len() тощо, які можна використовувати в програмі, як кажуть, «з коробки». Але більшість стандартної бібліотеки — це велика колекція модулів для роботи з різними типами даних, взаємодії з операційною системою, написання серверів і клієнтів для багатьох Інтернет-протоколів, розробки та налагодження коду [8].

Програми Python можуть мати різні простори імен - частини, де одне ім'я є унікальним і не пов'язане з тим самим ім'ям в інших просторах імен. Простір імен Python — це спосіб Python для зберігання інформації про активні змінні та об’єкти, на які вони вказують. У блоках коду, які виконуються в Python, є три простори імен: локальний, глобальний і вбудований. Коли змінна з’являється під час виконання програми, Python спочатку шукає її в локальному просторі імен. Якщо змінну не знайдено, пошук продовжується в глобальному просторі імен.

Якщо змінна там не знайдена, перевіряється вбудований простір імен. Якщо змінна не існує, виникає помилка і повертається виняток NamErr. Для модулів, команд, що виконуються в інтерактивному сеансі, або сценаріїв, запущених з файлів, глобальний і локальний простори імен збігаються. Створення будь-якої змінної чи функції або імпорт чогось з іншого модуля призводить до створення нового елемента в цьому просторі імен. Наприклад, математичний модуль містить математичні функції, а модуль random – функції для роботи з випадковими числами.

Кожна інструкція в програмі є командою, яка «повідає» процесору, що він має робити. Ось приклад директиви, яка може існувати в програмі Для нас це просто послідовність 0 і 1. Однак для процесора це інструкція виконати певну операцію. Процесор вашого комп’ютера може розуміти лише інструкції, написані машинною мовою. Машинна мова — це штучна мова, призначена для надсилання команд на комп’ютери. За допомогою машинної мови можна створювати ефективні програми, оскільки розробник має доступ до всіх функцій процесора. Машинна мова — мова низького рівня. Для кожної операції, яку може виконати процесор, є інструкції на машинній мові — інструкції для додавання, інструкції для віднімання тощо. Весь набір інструкцій, який може виконувати центральний процесор, називається набором інструкцій процесора [9].

Проект створювався для програми яка могла брати вхідні документи підписувати і відправляти підписані документи до центрального банку. Частковим аналогом такої програми є Дія, M.E.Doc, СОТА. M.E.Doc є найпопулярнішою програмою на українському ринку труда. За допомогою неї створюються різні типи документів які потім відправляються в налогову та в різни банки в Україні за потребою або в різноманітні компанії. Ця програма складається с багатьох модулів що постійно обновлюються декілька раз на місяць.

1.3. Аналоги програм для відправки підписаних документів.

За основу взятий модуль документообігу який підписую і відправляє документи для обробки далі або кінцевому отримувачу документа. M.E.Doc містить усі актуальні форми звітності, які швидко оновлюються відповідно до законодавства. Програма підтримує роботу з ЕЦП найбільш використовуваних центрів сертифікації, а також із захищеними носіями ключів SecureToken [10].

Над програмою працює численний штат: розробники, аналітиків, тестувальники, фахівців технічної підтримки та інших напрямків. M.E.Doc отримав позитивний експертний висновок у сфері технічного захисту інформації рівня ГЗ від Держспецзв'язку, що свідчить про високий рівень безпеки і захищеності програми.

У програмі M.E.Doc можна використовувати сертифікати ЕЦП, отримані в таких акредитованих центрах сертифікації ключів (АЦСК):

- АЦСК «Україна» (Центр сертифікації ключів «Україна»);
- АЦСК ІДЦ ДФС (Інформаційно-довідковий центр ДФС);
- АЦСК Укрзалізниця;
- АЦСК Держінформюст;
- АЦСК Укрсиббанк;
- АЦСК «Masterkey»;
- АЦСК ТОВ НВФ «УНІС» (Українські національні інформаційні системи);
- АЦСК УСС (Українські спеціальні системи);
- АЦСК МВС України;
- АЦСК НБУ;
- АЦСК Збройних сил;
- АЦСК Приватбанк.

Програма підтримує роботу з захищеними носіями для ЕЦП. У Програмі M.E.Doc також реалізовано автоматичне продовження сертифікатів цифрових підписів.

1.4. Бібліотеки мови програмування Python.

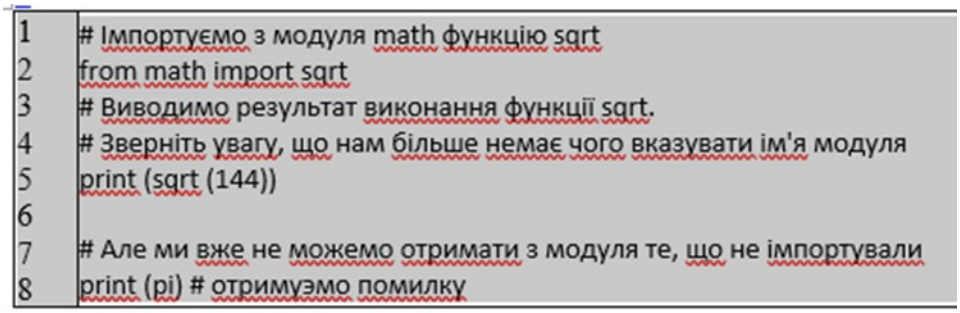
У Python дуже багато різноманітних бібліотек, які роблять універсальним його в програмуванні. Бібліотеки Python ще називають модулями за допомоги яких реалізуються складні та прості задачі. Система модулів дозволяє Логічно організувати свій код на Python. Групування коду в модулі значно спрощує процес написання та розуміння програм. Простіше кажучи, модуль в Python - це просто файл, який містить код Python. Кожен модуль в Python може містити змінні, оголошення класів і функції. Крім того, модуль може містити виконуваний код. Наприклад, щоб використовувати будь-який файл Python як модуль в іншому файлі, вам потрібно запустити в ньому команду імпорту. Команда імпорту в Python має такий синтаксис: Імпорт модуль_1[,модуль_2[,...модуль_N] [11].

Коли інтерпретатор Python зустрічає команду імпорту, він імпортує (отримує доступ) модуль (якщо він існує в шляху пошуку Python). Шлях пошуку Python – це список каталогів, які шукає інтерпретатор перед спробою завантажити модуль. Наприклад, щоб використовувати математичний модуль, ви повинні написати, дивитися рисунок 1.1.

```
1  import math
2  # Используем функцию sqrt из модуля math
3  print (math.sqrt(9))
4  # Печатаем значение переменной pi, определенной в math
5  print (math.pi)
```

Рис. 1.1 – Підключення модуля

Команда from ... import дозволяє вам імпортувати не увесь модуль цілком, а тільки певний його вміст дивитися рисунок 1.2 наприклад.



```
1  # Імпортуємо з модуля math функцію sqrt
2  from math import sqrt
3  # Виводимо результат виконання функції sqrt.
4  # Зверніть увагу, що нам більше немає чого вказувати ім'я модуля
5  print (sqrt (144))
6
7  # Але ми вже не можемо отримати з модуля те, що не імпортували
8  print (pi) # отримуэмо помилку
```

Рис. 1.2 – Команда from ... import

Вираз from ... import не імпортує увесь модуль, а тільки надає доступ до конкретних об'єктів, які ми вказали.

Команда: from ... import * в Python.

У Python так само можливо імпортувати все (змінні, функції, класи) за раз з модуля, для цього використовується конструкція from ... import * дивитися рисунок 1.3. наприклад.

```
1  from math import *
2
3  # Тепер у нас є доступ до усіх функція і змінним, визначеним в модулі math
4
5  print (sqrt(121))
6  print (pi)
7  print (e)
```

Рис.1.3 – Команда from ... import *

Проте це конструкцію слід використати з обережністю, оскільки при імпортуванні декількох модулів можна заплутатися у своєму власному коді.

Місцезнаходження модулів в Python:

Коли ви імпортуєте модуль, інтерпретатор Python шукає цей модуль в наступних місцях:

- Директорія, в якій знаходиться файл, в якому викликається команда імпорту;
- Якщо модуль не знайдений, Python шукає в кожній директорії, визначеній в консольній змінній PYTHONPATH;
- Якщо і там модуль не знайдений, Python перевіряє шлях заданий за умовчанням Шлях пошуку модулів збережений в системному модулі sys в змінній path. Змінна sys.path містить усі три вищеописані місця пошуку [12].

1.4.1. URLlib.

Urllib - це стандартна бібліотека, яка поставляється з Python і може використовуватися безпосередньо без установки надає наступні функції:

- Отримання відповіді;
- Запит сторінки;
- Налаштування проксі і файлів cookie;
- Обробка виключень.

Функції, які необхідні пошуковому роботу, які можна знайти в urllib. Ця стандартна бібліотека може дати глибоке уявлення про зручну бібліотеку запитів. Зв'язок між urllib, urllib2 у Python2 він розділений на бібліотеку urllib і бібліотеку urllib2. Бібліотека urllib2 є оновленням бібліотеки urllib. Ці два модулі відповідають за різні функції, складний в створені зв’язків між версіями, є проблеми з кодуванням. На практиці точно визначити, яку бібліотеку слід використати, - головний біль.

У python3 urllib і urllib2 були об'єднані у бібліотеку urllib, які включають в собі наступні чотири підмодулі:

- urllib.request: бібліотека запиту;
- urllib.parse: бібліотека розбору URL;
- urllib.error: бібліотека обробки виключень;
- urllib.robotparser: бібліотека розбору robots.txt. request.urlopen() використовується для відкриття URL- адреси, URL- адреса може бути рядком або об'єктом запиту. Приклад GET запиту рисунок 1.4.

```
import urllib.request
response = urllib.request.urlopen('http://www.baidu.com')
print(response.read().decode('utf-8'))
```

Рис. 1.4 – GET запиту

Модуль помилок в основному використовується для перехоплення виключень. Є два типи помилок: URLError (повідомлення про помилку) і HTTPError (код помилки), де HTTPError - це підклас URLError. У URLError є тільки один атрибут: reason, який може виводити інформацію про помилку тільки при виявленні виключення. У HTTPError є три атрибути: код, reason, headers, де code - це код помилки 404/403, reason - повідомлення про помилку, headers - інформація заголовків. Наприклад дивитися рисунок 1.5.

```
from urllib import request,error
try:
    response = request.urlopen("http://pythonsite.com/1111.html")
except error.HTTPError as e:
    print(e.reason)
    # Not Found
    print(e.code)
    # 404
    print(e.headers)
```

Рис. 1.5 – HTTPError (код помилки)

Функції розбору URL- адрес зосереджені на розділенні рядка URL- адреси на її компоненти або на об'єднанні компонентів URL- адреси в рядок URL- адреси. Urllib.parse розберає адреса на шість компонентів, повернувши кортеж з 6 елементів. Це відповідає загальній структурі URL:

test:-urlll/path111;parameters?query#fragment. Кожен елемент кортежу є рядком, може бути порожнім. Компоненти не розбиваються на дрібніші частини (наприклад, місце розташування в мережі є одним рядком). Вказані вище роздільники не є частиною результату, за винятком ведучої косої риски в компоненті шляху, яка зберігається, якщо є присутня. Приклад лістингу коду показно на рисунку 1.6.

```
>>> from urllib.parse import urlparse
>>> urlparse("scheme://netloc/path;parameters?query#fragment")
ParseResult(scheme='scheme', netloc='netloc', path='/path;parameters', params='',
            query='query', fragment='fragment')
>>> o = urlparse("http://docs.python.org:80/3/library/urllib.parse.html?"
...              "highlight=params#url-parsing")
>>> o
ParseResult(scheme='http', netloc='docs.python.org:80',
            path='/3/library/urllib.parse.html', params='',
            query='highlight=params', fragment='url-parsing')
>>> o.scheme
'http'
>>> o.netloc
'docs.python.org:80'
```

Рис. 1.6 – Функції розбору URL

1.4.2. Бібліотека PyQt.

Не дивлячись на те, що веб-сервера, вебдодатки і мобільні застосування обганяють ринок розробки програмного забезпечення, як і раніше існує попит на настільні застосування з традиційним графічним призначеним для користувача інтерфейсом (GUI). Для розробників, зацікавлених в створенні подібних

застосувань на Python, є широкий вибір бібліотек, включаючи Tkinter, wxpython, Pyqt, Pyside2 та інші [13].

PyQt - це бібліотека Python для створення додатків з графічним інтерфейсом з використанням інструментарію Qt. PyQt, створений Riverbank Computing, є безкоштовним програмним забезпеченням (під ліцензією GPL) і знаходиться в розробці з 1999 року. PyQt5 був випущений в 2016 році і останній раз оновлювався в жовтні 2021 року.

Pyqt5 заснований на Qt v5 і включає класи, які охоплюють графічні інтерфейси користувача, а також обробку XML, мережеву взаємодію, регулярні вирази, потоки, бази даних SQL, мультимедіа, перегляд веб-сторінок і інші технології, доступні в Qt. Pyqt5 реалізує більше тисячі таких класів Qt в наборі модулів Python, кожен з яких міститься в пакеті Python верхнього рівня під назвою Pyqt5. Цей пакет сумісний з Windows, Unix, Linux, macos, ios і Android. Це може бути привабливою функцією, якщо шукаєте бібліотеку або інфраструктуру для розробки багатоплатформених застосувань з рідним зовнішнім виглядом для кожної платформи.

Віджети QWidgetis базовий клас для всіх об'єктів інтерфейсу користувача або віджетів. Це прямокутні графічні компоненти, які можна розмістити у вашій програмі для створення графічного інтерфейсу. Віджети містять ряд атрибутів і методів для імітації їх вигляду і поведінки. Вони також можуть малювати себе на екрані. Віджети також отримують клацання мишею, натискання клавіш та інші події від користувача, віконної системи та багатьох інших джерел. Щоразу, коли віджет перехоплює подію, він випромінює сигнал, щоб оголосити про зміну свого стану. PyQt5 має багату і сучасну колекцію віджетів, які служать для декількох цілей. Деякі з найпоширеніших і корисних віджетів:. Кнопки. Лейблами. Редагування рядків. Комбінованих полів. Кнопки радіо. Давайте поглянемо ближче на кожен з цих віджетів. Першим з’являється кнопка . Ви можете створити

кнопку, створивши екземпляр QPusButtonclass, який надає класичну кнопку команди [14].

Типовими кнопками є кнопки:

- OK – Гаразд;
- Cancal – Скасувати;
- Apply – Застосувати;
- Yes – Так;
- No – Ні;
- Close - Закрити.

Даний опис подібної панелі в Linux-системі зображено на рисунку 1.7.

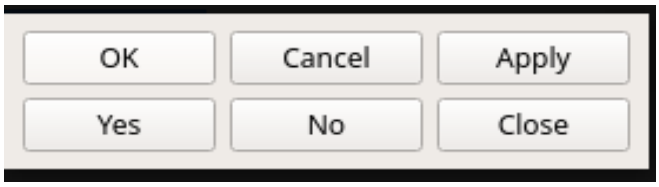


Рис.1.7 – Кнопки на панелі ОС Linux

Мітки, які можна створити за допомогою QLabel, представлено рисунком 1.8. Мітки дозволяють відображати корисну інформацію як текст або зображення.

Рис. 1.8 – Вид міток QLabel

Такі мітки можна використовувати для кращого пояснення мети або використання графічного інтерфейсу. Можна налаштувати їх вигляд декількома

способами, і вони можуть навіть приймати HTML-текст, як ви бачили раніше. Крім того, мітки можна використовувати для визначення ключа Mnemonic Focus для іншого віджета.

Іншим поширеним віджетом є редагування рядків (рисунок 1.9), однорядкового текстового поля, яке можна створити за допомогою QLineEdit. Редагування рядків буде корисним, коли користувач повинен ввести або редагувати дані у текстовому форматі. Ось як вони виглядають на Linux-системі.

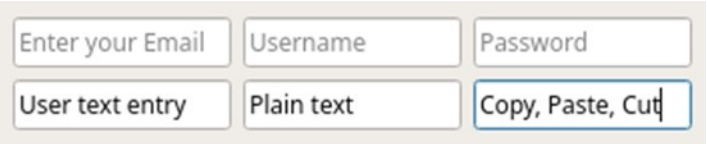


Рис. 1.9 – Вид QLine Edit

Таке лінійне редагування забезпечує базові операції редагування, такі як копіювання, вставка, скасування, повторення, перетягування, капля, і так далі На наведеному вище малюнку також можна побачити, що об'єкти в першому рядку відображають текст-шпаклівка, щоб повідомити користувачеві про те, який вхід потрібний.

Спадний список (рисунок 1.10) є ще одним корисним віджетом, який можна створити за допомогою QComboBox. Спадний список надасть вашому користувачеві список параметрів, які займуть мінімальну кількість місця на екрані. Ось приклад спадного списку на системі Linux.

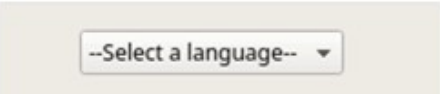


Рис. 1.10 – Спадний список

Цей спадний список є лише для читання, це означає, що користувач може вибрати один з декількох варіантів, але не може додати свій. Крім того, ви можете

змінити спадний список, за допомогою якого користувач може додати нові параметри. Вони можуть містити растрові зображення, рядки або обидва.

Останнім віджетом, про який зараз говоритимемо, є кнопка радіо (рисунок 1.11), яку можна створити за допомогою QRadioButton. QRadioButton — це кнопка вибору, яку можна увімкнути (позначити) або вимкнути (вимкнути). Радіо-кнопки корисні тоді, коли потрібно зробити вибір користувачем одного з багатьох варіантів. У цьому випадку на екрані одночасно видно всі параметри.

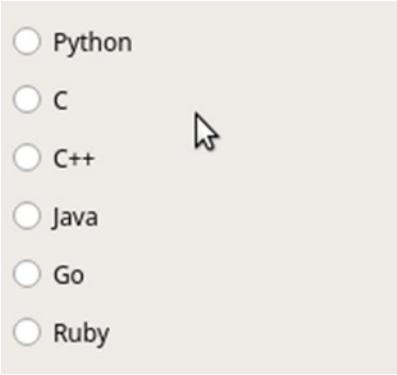


Рисунок 1.11 – QRadioButton

У цій групі кнопок у цей час можна перевірити лише одну кнопку. Якщо користувач вибирає інший перемикач, попередньо вибрану кнопку буде автоматично вимкнено.

Менеджери компонувань — це класи, які дозволяють змінювати розмір і розміщувати віджети там, де вони мають бути у формі програми. Менеджери конфігурації автоматично пристосовуються до змін розміру і вмісту. Вони також контролюють розмір віджетів всередині них. Це означає, що віджети у компонуванні автоматично змінюються після зміни розміру форми. PyQt надає чотири базові класи керування розкладки:

- QHBoxLayout;
- QVBoxLayout;
- QGridLayout;

- QFormLayout;

Першим класом менеджера розкладок є QHBoxLayout, рисунок 1.12, який впорядковує віджети горизонтально зліва направо.

Рис. 1.12 – QHBoxLayout

Наступним класом менеджера розкладок є QVBoxLayout (рисунок1.13), який впорядсовує віджети вертикально згори вниз.

Рис. 1.13 – QVBoxLayout

Кожен новий віджет буде показано під попереднім. Ви можете скористатися цим класом для створення об’ єктів компонування для вертикального блоку і впорядкування віджета згори вниз.

Третім класом Layout Manager є QGridLayout (рисунок 1.14), який організовує віджети у вигляді сітки рядків і стовпчиків. Кожен віджет матиме відносне розташування у сітці. Визначити положення віджета можна, передавши йому пару координат у вигляді (рядок, стовпчик). Ці координати мають бути

дійсними. Вони визначають, на якій клітинці ґратки ви збираєтеся розмістити віджет. Схема сітки працює так, як показано на рисунку 1.14.

Widget (0, 0)	Widget (0, 1)	...Widget (0, n)
Widget (1, 0)	Widget (1, 1)	...Widget (1, n)
...Widget (n, 0)	...Widget (n, 1)	...Widget (n, n)

Рис. 1.14 – QGridLayout

QGridLayout займає місце, дане їй батьком, розділяє його на рядки і стовпчики розміщує кожен віджет у своїй комірці [15].

Останнім класом менеджера розкладок є QFormLayout (рисунок 1.15), який організовує віджети в двоколонковий макет. У першому стовпчику типово буде показано повідомлення у мітках. У другому стовпчику зазвичай містяться такі віджети, як QLineEdit, QComboBox, QSpinBox тощо. Вони дозволяють користувачеві вводити або редагувати дані, пов'язані з інформацією в першому стовпчику. На наступній діаграмі показано, як макети форм працюють на практиці.

Label 1	Widget 1
Label 2	Widget 2
...Label n	...Widget n

Рис. 1.15 – QFormLayout

Лівий стовпчик складається з міток, а правий — з віджетів полів. Якщо ви маєте справу з програмою для роботи з базами даних, таке компонування може бути привабливим варіантом для збільшення продуктивності при створенні форм.

За допомогою PyQt можна розробляти два типи настільних застосунків з GUI. Залежно від класу, який ви використовуєте для створення головної форми або вікна, ви матимете один з таких варіантів:

У головному вікні програми: Головне вікно програми успадковано від QMainWindow [16].

Застосунок у стилі діалогів (англ. Dialog-style): Головне вікно програми успадковане від QDialog.

Спочатку програма розпочне роботу з програмами у стилі діалогових вікон. У наступному розділі ви побачите основні програми у стилі вікон.

Для розробки застосунку в стилі діалогів потрібно створити клас GUI, успадкований від QDialog, який є базовим класом всіх діалогових вікон. Діалогове вікно завжди є вікном верхнього рівня, яке ви можете використовувати як головне вікно для вашої програми у стилі діалогів.

Діалогові вікна також часто використовуються в основних віконних програмах для короткого спілкування і взаємодії користувача. Коли діалогові вікна використовуються для спілкування з користувачем, вони можуть бути:

- Модальні діалогові вікна: Блокують будь-які інші видимі вікна в одному додатку. Показати модальне діалогове вікно можна за допомогою виклику . exec_();
- The Non-modal Dialogs: Працює незалежно від інших вікон в одному додатку. Показати немодальне діалогове вікно можна за допомогою . show ();

- У діалоговому вікні на зразок кнопки Гаразд і Скасувати можна також вказати значення повернення і типові кнопки (наприклад, OK і Скасувати) [17].

Діалогове вікно завжди є віджетом верхнього рівня. Якщо у нього є батьківський елемент, його типове розташування знаходиться посередині верхнього батьківського віджета. Цей тип діалогового вікна також матиме спільний запис на батьківській панелі задач. Якщо ви не встановите визначення походження цього діалогового вікна, діалогове вікно отримає власний запис у системній панелі задач. Ось приклад того, як скористаєтеся QDialog (рисунок 1.16) для розробки програми у стилі діалогів.

```
5  import sys
6
7  from PyQt5.QtWidgets import QApplication
8  from PyQt5.QtWidgets import QDialog
9  from PyQt5.QtWidgets import QDialogButtonBox
10 from PyQt5.QtWidgets import QFormLayout
11 from PyQt5.QtWidgets import QLineEdit
12 from PyQt5.QtWidgets import QVBoxLayout
13
14 class Dialog(QDialog):
15     """Dialog."""
16     def __init__(self, parent=None):
17         """Initializer."""
18         super().__init__(parent)
19         self.setWindowTitle('QDialog')
20         dlgLayout = QVBoxLayout()
21         formLayout = QFormLayout()
22         formLayout.addRow('Name:', QLineEdit())
23         formLayout.addRow('Age:', QLineEdit())
24         formLayout.addRow('Job:', QLineEdit())
25         formLayout.addRow('Hobbies:', QLineEdit())
26         dlgLayout.addLayout(formLayout)
27         btns = QDialogButtonBox()
28         btns.setStandardButtons(
29             QDialogButtonBox.Cancel | QDialogButtonBox.Ok)
30         dlgLayout.addWidget(btns)
31         self.setLayout(dlgLayout)
32
33 if __name__ == '__main__':
34     app = QApplication(sys.argv)
35     dlg = Dialog()
36     dlg.show()
37     sys.exit(app.exec_())
```

Рис.1.16 – Вид лістингу коду діалогового вікна

У рядку 14 створює повний клас Dialogg для графічного інтерфейсу, успадкованого від QDialog.

У рядку 20 присвоюється QVBoxLayoutobject dlgLaout.

У рядку 21 у формLayout визначено QVFormLayoutobject.

До формLayout додають віджети на рядки 22—25.

Для роботи з лінійним рядком 26 використовується dlgLayoutto розміщувати всі віджети на формі.

У рядку 27 передбачено зручний об’єкт для кнопок діалогового вікна.

На ці рядки 28 і 29 додано дві стандартні кнопки: OK I Cancel.

Рядки з 33 по 37 обгортають код шаблону в if __name_ == '__main__'.

Головні вікна. Більшість часу GUI-застосунки будуть працювати в основному віконному стилі. Це означає, що у них буде смужка меню , декілька панелей інструментів, смужка стану і центральний віджет, який буде основним елементом графічного інтерфейсу. Часто ваші програми також матимуть декілька діалогових вікон для виконання додаткових дій, які залежать від введеного користувачем.

Цей клас QMainWindow (рисунок 1.17) використовується для розробки застосунків в основному віконному стилі. Він повинен бути успадкований від QMainWindow для створення свого первинного класу GUI. Прикладом класу, що походить від QMainWindow, вважається головне вікно. QMainWindow надає фреймворк для створення графічного інтерфейсу для вашої програми. Клас має власну вбудовану компоновку, яку можна використовувати для розміщення наступних:

- У верхній частині вікна розташовано одну смужку меню. На панелі меню міститься головне меню програми;
- По боках вікна розташовано декілька панелей інструментів. Панелі інструментів придатні для зберігання кнопок інструментів та інших віджетів, таких як QComboBox, QSpinBox тощо:
- У центрі вікна розташовано один центральний віджет. Центральним віджетом може бути будь-який тип або композитний віджет;

- Навколо центрального віджета розташовано декілька доку. Віджети док — це невеликі рухомі вікна;
- У нижній частині вікна розташовано смужку стану.

На панелі стану буде показано відомості щодо загального стану програми.

Цей клас не може створити головне вікно без першого налаштування центрального віджета. Слід мати центральний віджет, навіть якщо він є лише заповнювачем. У цьому випадку слід використовувати QWidgetobject як центральний віджет. Цей параметр можно встановити вибравши центральний віджет головного вікна за допомогою параметра . setCentralWidget (). У головному компонуванні вікна передбачено лише один центральний віджет, але він може бути єдиним або композитним віджетом [18].

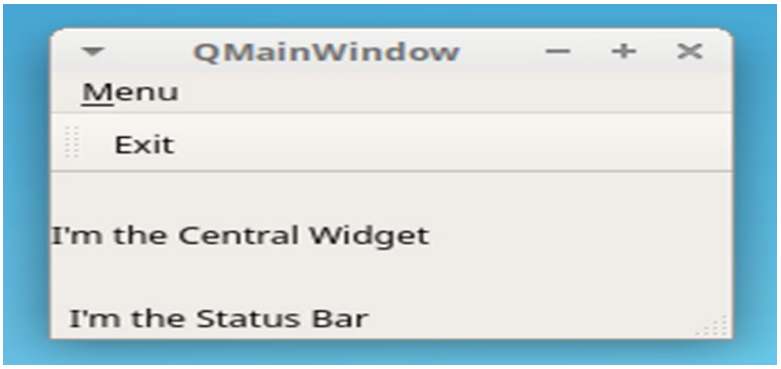


Рис. 1.17 – QMainWindow

У головному вікні програми передбачено такі компоненти:

- Одне головне меню називається Menu;
- Одна панель інструментів з функціональною кнопкою «вийти»;
- Один центральний віджет (QLabelobject);
- Один рядок стану в нижній частині вікна.

Цикл подій. Програми для GUI є керованим подіями. Це означає, що функції і методи виконуються у відповідь на дії користувача, наприклад, натискання

кнопки, вибір елемента зі списку, введення або оновлення тексту при редагуванні тексту, натискання клавіш тощо ці дії користувача зазвичай називають подіями.

Події зазвичай обробляються циклом подій (також називається базовим циклом). Цикл подій — нескінченний цикл, в якому обробляються і відправляються всі події від користувача, віконної системи і будь-яких інших джерел. Цикл подій чекає, коли відбудеться подія, а потім посилає її виконати якесь завдання. Цикл подій триває до завершення застосунку.

Цикли подій використовуються всіма програмами GUI. Цикл подій — це своєрідний нескінченний цикл, що чекає на події, що відбуваються. Якщо подія відбувається, цикл перевіряє, чи є подія подією припинення. У цьому випадку петля переривається і застосунок закривається. В іншому випадку подія відправляється в чергу подій застосунку для подальшої обробки, і цикл починається знову.

Сигнали і слоти. Віджети PyQt діють як перехоплювачі подій. Це означає, що кожен віджет може перехоплювати певну кількість подій, таких як клацання мишею, натискання клавіш тощо. Ці події віджети завжди випромінюють сигнал, який є своєрідним повідомленням, що сповіщатиме про зміну його стану.

Сам сигнал нічого не робить. Якщо ви хочете, щоб сигнал активували дію, вам потрібно підключити його до гнізда. Це функція або метод, який буде діяти при випромінюванні з'єднувальних сигналів. Ви можете використовувати будь-який callable (або callback) Python як слот.

Якщо сигнал підключений до паза, то щілину називають всякий раз, коли випромінюється сигнал. Якщо сигнал не підключений до будь-якого слота, то нічого не відбувається і сигнал ігнорується. Ось деякі з найкорисніших ознак цього механізму:

- Сигнал може бути підключений до одного або декількох слотів;
- Сигнал також може бути підключений до іншого сигналу;
- Слот може бути пов'язаний з одним або більше сигналами.

Для підключення сигналу до слота можна використовувати такий синтаксис: widget.signal.connect(slot_function).

Це з’єднує sot_ function з widget.signal. Кожний раз, коли signalis випромінює сигнал, буде викликать функцію slot_function(). Приклад сигналів,рисунок 1.18.

```
def greeting():     """Slot function."""
    if msg.text():
        msg.setText("")
    else:
        msg.setText("Hello World!")

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('Signals and slots')
layout = QVBoxLayout()

btn = QPushButton('Greet')
btn.clicked.connect(greeting) # Connect clicked to greeting()
layout.addWidget(btn)
msg = QLabel('')
layout.addWidget(msg)
window.setLayout(layout)
window.show()
sys.exit(app.exec_())
```

Рис. 1.18 – Вид лістингу коду «Приклад сигналів»

1.4.3. smtp

Python постачається з вбудованим модулем smtplib для надсилання повідомлень електронною поштою за допомогою протоколу SMTP. SmtpliBuses RFC 821 протокол серверу SMTP Gmail для надсилання листів електронною поштою, але ті ж принципи поширюються і на інші поштові служби. Хоча більшість провайдерів електронної пошти використовують ті ж самі порти з'єднання, що і Google Gmail.

Як запустити захищене SMTP-з'єднання. Коли надсилаєте листа через Python, слід переконатися, що SMTP-з'єднання зашифровано так, щоб інші не могли легко отримати доступ до вашого повідомлення і реєстраційних даних. SSL (Secure Sockets Layer) і TLS (Transport Layer Security) — два протоколи, які можуть бути використані для шифрування SMTP-з'єднань. Не потрібно використовувати будь-яку з них під час використання локального сервера відкладки.

Існує два способи безпечного з'єднання з поштовим сервером:

- SMTP-з'єднання, захищене від початку за допомогою протоколу SMTP_SSL ();
- Незахищене SMTP-з'єднання, яке потім можна зашифрувати за допомогою starttls ().

В обох випадках Gmail буде шифрувати листи з використанням TLS, оскільки це більш безпечний наступник SSL. З міркувань безпеки Python настійно рекомендує використовувати create_default_context()from. За допомогою цього пункту можна завантажити довірені сертифікати CA, увімкнути перевірку вузлів і сертифікатів, а також спробувати вибрати достатньо безпечний протокол і параметри шифрування.

Smtplib — вбудований модуль Python для відправки електронної пошти на будь-який інтернет-комп'ютер з фоновою службою SMTP або ESMTP-прослуховування.

SMTP_SSL() потрібно для безпечного з'єднання з почтовим сервисом. starttls()є альтернативою. Майю на увазі, що Gmail вимагає підключення до порту 465 при використанні протоколу SMTP_SSL() і порту 587 при використанні . starttls().

Варіант 1: Використовувати SMTP_ SSSL().

Наступний приклад коду, рисунок 1.19 створює безпечне з'єднання з сервером SMTP Gmail за допомогою SMTP_SSL()smtplib для ініціювання

з'єднання з TLS-шифруванням. Типовий контекст sslets ім'я вузла та його сертифікати та оптимізує безпеку з'єднання.

```
import smtplib, ssl

port = 465  # For SSL
password = input("Type your password and press enter: ")

# Create a secure SSL context
context = ssl.create_default_context()

with smtplib.SMTP_SSL("smtp.gmail.com", port, context=context) as server:
    server.login("my@gmail.com", password)
    # TODO: Send email here
```

Рис. 1.19 – Вид лістингу коду варіант № 1: Використовувати SMTP_ SSSL ()

Використання smtplib.SMTP_SSL() як сервера: забезпечує автоматичне закриття з'єднання в кінці відступного блоку коду. Якщо portis zero або не вказано, . SMTP_SSL () використовуватиме стандартний порт SMTP через SSL (порт 465).

Зберігання пароля електронної пошти в коді не є безпечним, особливо якщо ви збираєтеся поділитися ним з іншими. Замість цього, використовуйте input(), щоб користувач міг вводити свій пароль під час запуску скрипту, як у прикладі вище. Якщо ви не бажаєте, щоб ваш пароль з’ явився на екрані під час введення пароля, ви можете імпортувати модуль getpassmodule і використовувати його. Getpass () замість сліпого введення пароля.

Варіант № 2: Використовувати starttls().

Замість використання . SMTP_SSL () для створення безпечного з'єднання з початку ми можемо створити незахищене SMTP-з'єднання і зашифрувати його за допомогою starttls(), рисунок 1.20.

Для цього створюють екземпляр smtplib.SMTP, який інкапсулює SMTP-з'єднання і дозволяє отримати доступ до його методів. Рекомендуємо вам вказати

ваш сервер SMTP і порт на початку вашого скрипту, щоб його можна було легко налаштувати.

Фрагмент коду, наведений нижче, використовує дизайн сервера = SMTP(), а не формат з SMTP() як сервер, який ми використовували в попередньому прикладі. Щоб переконатися, що ваш код не зависає, коли щось піде не так, вставте головний код в триблок і дозволяйте відмовоблоку відображати будь-які повідомлення про помилку в stdout.

```
smtp_server = "smtp.gmail.com"
port = 587  # For starttls
sender_email = "my@gmail.com"
password = input("Type your password and press enter: ")

# Create a secure SSL context
context = ssl.create_default_context()

# Try to log in to server and send email
try:
    server = smtplib.SMTP(smtp_server,port)
    server.ehlo() # Can be omitted
    server.starttls(context=context) # Secure the connection
    server.ehlo() # Can be omitted
    server.login(sender_email, password)
    # TODO: Send email here
except Exception as e:
    # Print any error messages to stdout
    print(e)
finally:
    server.quit()
```

Рис. 1.20 – Вид лістингу віріант № 2: використовувати starttls()

1.5. ІТТ користувач.

ІТТ - ця програма призначена для застосування підпису на комп’ютерній техніці документів клієнтів та виконує наступні функції:

- Накладення універсального електронного підпису чи печатки на будь-яку інформацію в електронному вигляді, а також для криптографічного захисту інформації, шляхом її направленого шифрування;

- Генерацію ключів клієнтів АЦСК, резервне копіювання особистого ключа з одного носія ключової інформації на інший, знищення особистого ключа;
- перевірку сертифіката користувача;
- Формування та передачу до АЦСК запиту на скасування або блокування сертифіката користувача.

Доступ до сертифікатів АЦСК ІДД ДФС, серверів АЦСК ІДД ДФС, сертифікатів інших користувачів та списку відкликаних сертифікатів з метою перегляду, пошук сертифікатів користувачів у файловому сховищі, визначення статусу сертифікатів користувачів, перевірку цілісності сертифікатів.

1.5.1. Бібліотека IIT користувач.

Бібліотека IIT, рисунок 1.21 – це розширення веб браузера для підпису та управління особистими ключами і сертифікатами, а також EPS і шифрування даних. Її часто використовують в документообігу та в банківських установах. Ця бібліотека стала розповсюджена та території України яка була затверджена департаментом ДПС і в багатьох інших державних службах.

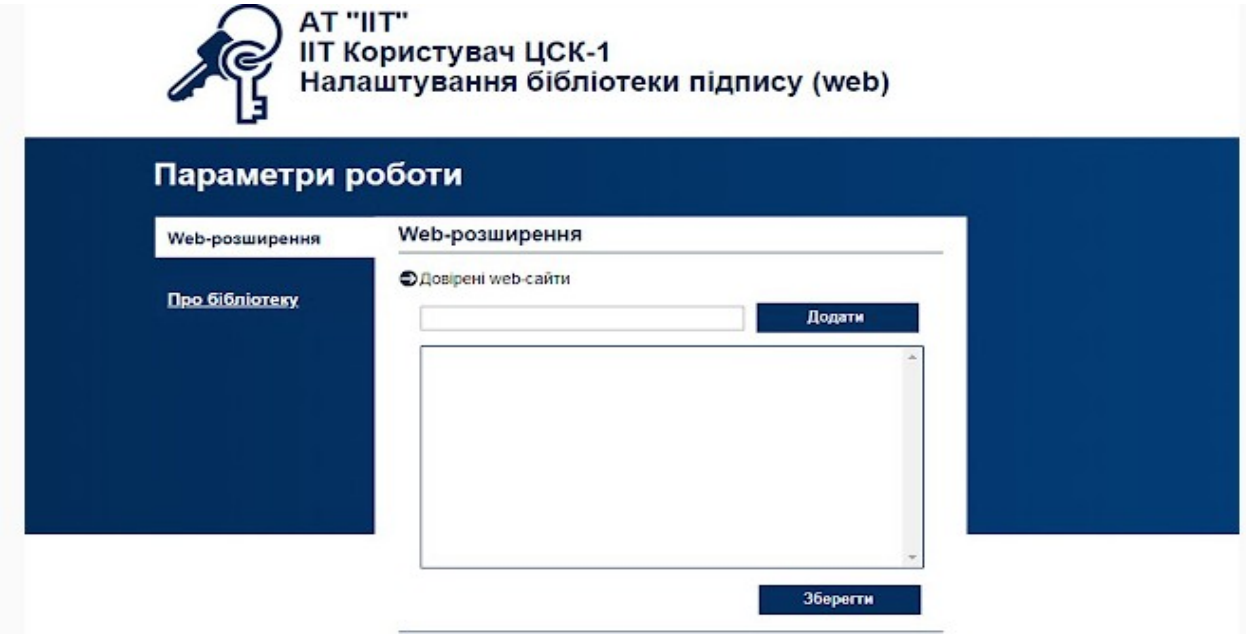


Рис. 1.21 – Вид інтефейсі бібліотеки ІІТ

1.5.2. Агент підпису ІІТ користувач.

Агент підпису ІІТ, рисунок 1.22 користувач –це додатковий пакет від акціонерного товариства ІІТ для налаштування ключів та накладання електронного підпису документу.

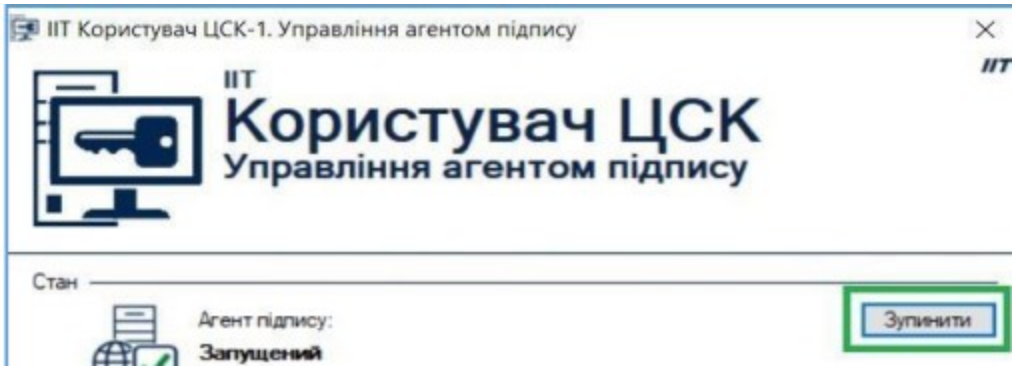


Рис. 1.22 – Агент підпису ІТТ

Цей додатковий пакет працює з двома типами підпису документа а саме:

- Кристал 1K, який вбудований в малогабаритний USB флеш-накопичувач, що містить в собі інформацію про ключ;
- Алмаз-1K" (пристрій Bluetooth) - це токен, який працює на Bluetooth версії 4.0 або 5.0, з підтримкою технології Low Energy (LE).

РОЗДІЛ 2
ПРОЕКТУВАНЯ ТА РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ПРОЄКТУ БАКАЛАВРСЬКОЇ РОБОТИ

2.1. Налагодження IDE PyCharm

PyCharm - це продукт який випускається компанією під назвою “Jetbrains”. Це дуже гарний у використані редактор коду так як компанія спеціалізується на випуску програмного забезпечення для програмістів різних сферах інформаційної інфраструктури. Цей продукт використовується в розробці програмного продукту обраної тематики дипломної роботи, тому-що легкий в використані та багатий на функціонал.

PyCharm аналізує розроблений код в першу чергу, результати буде негайно показано на індикаторі перевірки у верхньому правому куті редактора. Ця

індикація перевірки працює подібно світлофору: Коли вона зелена, все добре і можна продовжити код; жовте світло означає деякі незначні проблеми, які, однак, не вплинуть на компіляцію; але коли червоне світло, це означає, що у вас серйозні помилки. Натисніть її, щоб переглянути подробиці у вікні проблеми, на рисунку 2.1.

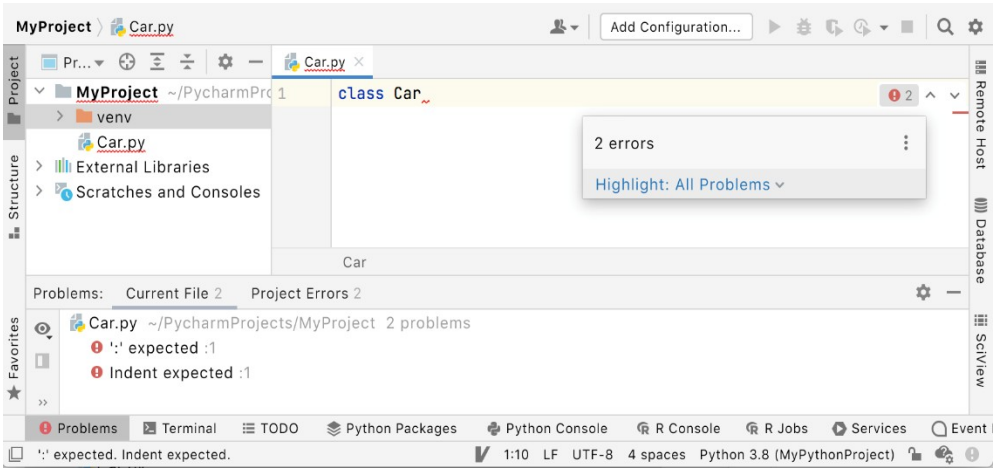


Рис. 2.1 – PyCharm відображення помилок

Створення функції __init__function, рисунок 2.2: коли просто набираємо відкриваючу фігурну дужку, PyCharm створює всю конструкцію коду (обов'язковий параметр self, закрита фігурна дужка і двокрапка) і забезпечує правильний відступ.

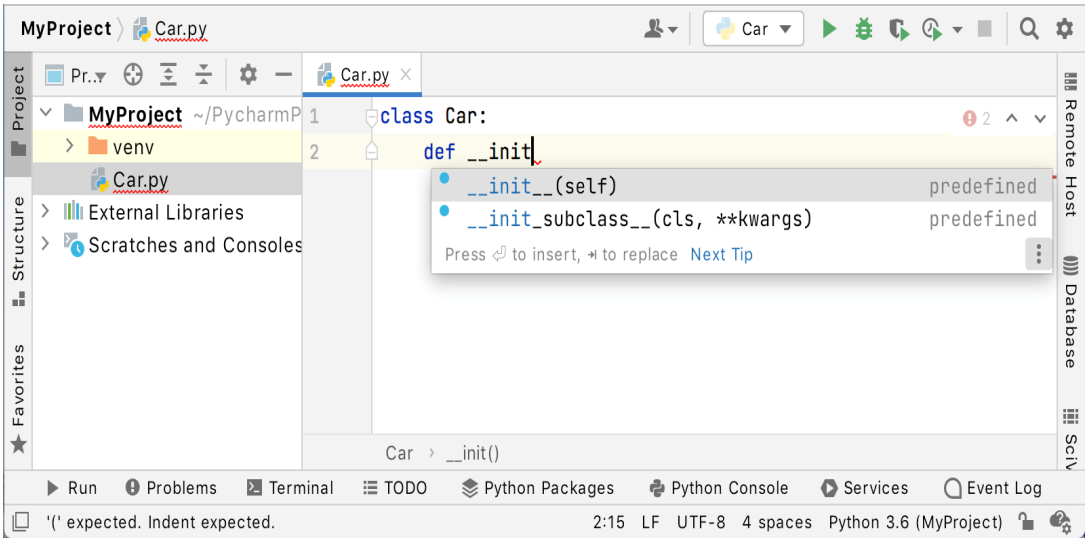


Рис. 2.2 – Створення функції __init__function

Якщо під час редагування коду помітити будь-які попередження про перевірку, натисніть піктограму лампочки, щоб переглянути список можливих виправлень і рекомендованих дій, рисунок 2.3:

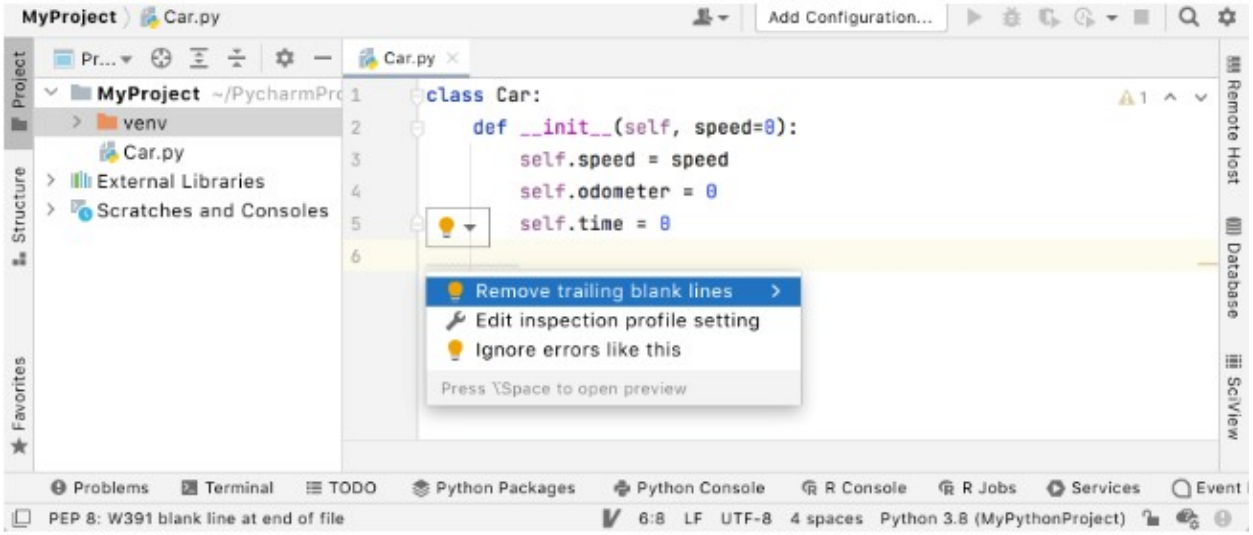


Рис. 2.3 – Приклад редагування коду

Способи запуску коду. Клацніть правою кнопкою миші редактор і виберіть пункт «Запустити машину» в контекстному меню, рисунок 2.4.

Натисніть Ctrl+Shift+F10. Оскільки цей скрипт МП Python містить основну функцію, можна натиснути значок у полі. Якщо навести на нього вказівник миші, відобразяться доступні команди.

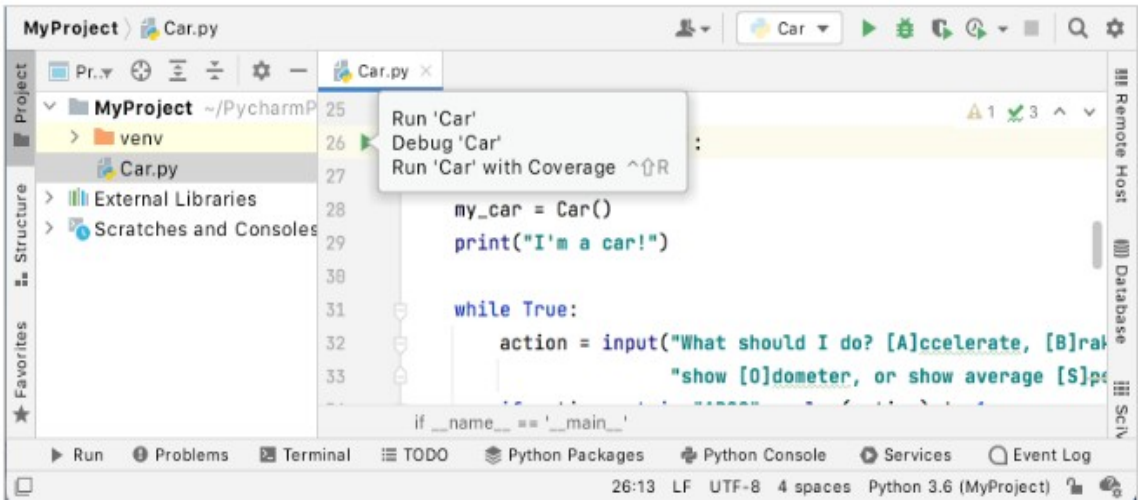


Рис. 2.4 – Вид запуску коду

Якщо натиснути піктограму, побачимо спливаюче меню доступних команд. Виберіть «Запустити машину», рисунок 2.5.

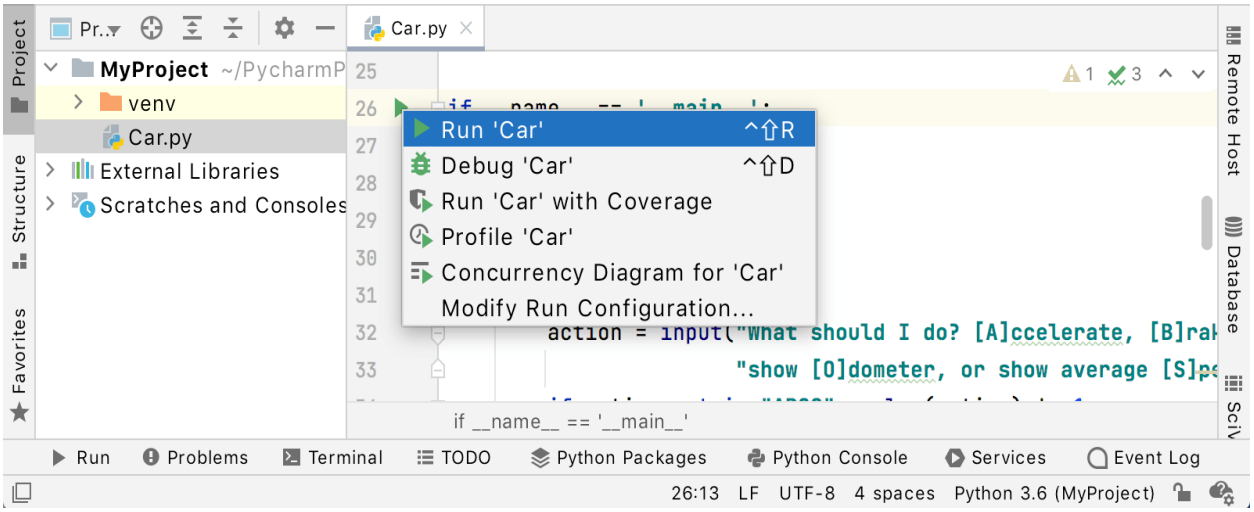


Рис. 2.5 – Види запуску коду

PyCharm - виконує код у вікні інструмента «Виконати», рисунок 2.6.

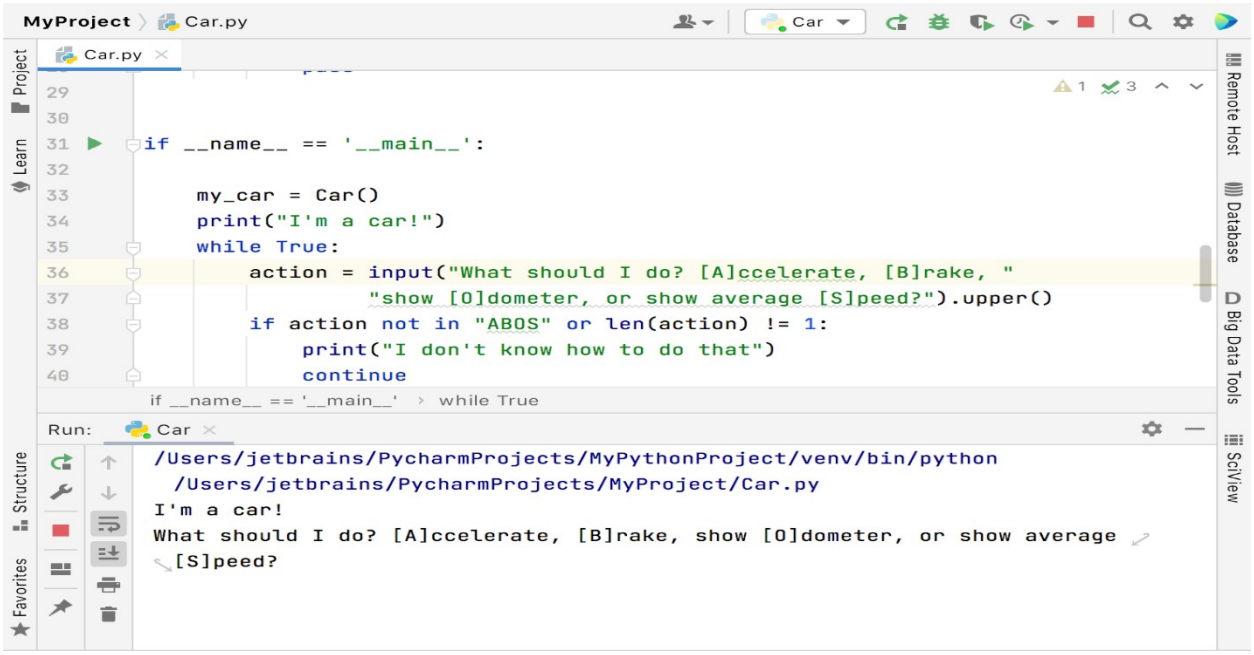


Рис. 2.6 – Вікно виконання коду

Тут можно ввести очікуванні значення та переглянути висновок скрипта. Зверніть увагу, що PyCharm створив тимчасову конфігурацію запуску/налагодження файлу Car. Цей параметр можно зберегти, щоб зробити його постійною конфігурацією, або змінити параметри відладки. Налаштування параметрів PyCharm.

У PyCharm можна налаштовувати параметри на двох рівнях: на рівні проекту та глобальному, рисунок 2.7.

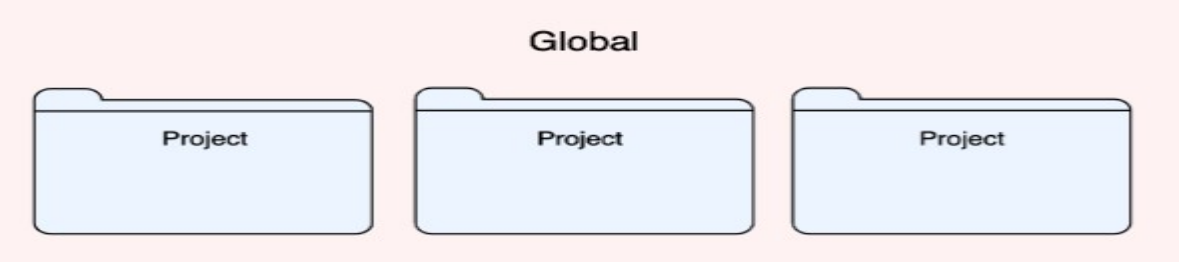


Рис. 2.7 – Вид структури глобального рівня проєкту

Глобальні налаштування застосовуються до всіх проектів, які ви відкриваєте з певним встановленням або версією PyCharm. До таких настройок відносяться

зовнішній вигляд IDE (теми, колірні схеми, меню та панелі інструментів), налаштування повідомлень, набір встановлених та увімкнених плагінів, налаштування відладчика, завершення коду тощо.

Щоб налаштувати IDE, виберіть PyCharm | Налаштування для macOS або Файл | Установки для Windows та Linux. Або натисніть Ctrl+Alt+Sабо клацніть на панелі інструментів.

Налаштування, не позначені відповідним значком у діалоговому вікні «Налаштування/Налаштування», є глобальними та застосовуються до всіх існуючих проектів поточної версії PyCharm, показано на рисунку 2.8.

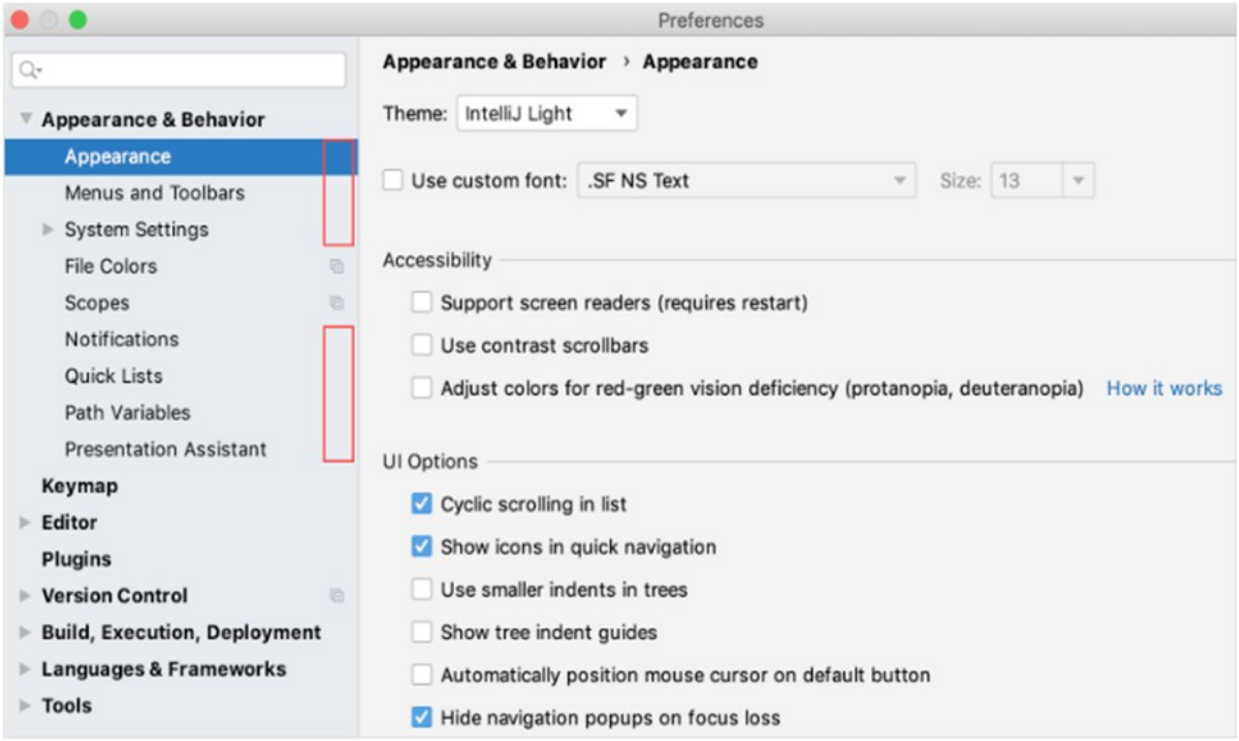


Рис. 2.8 – Інтерфейс глобальних налаштувань PyCharm

Відновити налаштування IDE. Коли відновлюється налаштування IDE за замовчуванням, PyCharm створює резервну копію конфігурації в каталозі. Ви завжди можете відновити налаштування з цієї резервної копії. Зробіть резервну копію ваших налаштувань і відновіть значення за промовчанням.

У головному меню виберіть Файл | Керування налаштуваннями IDE | Відновити стандартні налаштування .Або Shift двічі натисніть та введіть Restore default settings. Коли PyCharm відновлює налаштування IDE за замовчуванням, він створює резервний каталог із вашою конфігурацією в: «C:\Users\User\AppData\ Roaming\JetBrains\PyCharm2018.6-backup».

2.2. Використання бази даних MongoDB Python

Для роботи MongoDB в середовищі пайтон необхідно встановити офіціальний дистрибутив PyMongo. Його можно встановити за допомогою вбудованого пакету pip Python3.9.

Першим кроком при роботі з PyMongo є створення MongoClient працюючого екземпляра mongod. Зробити це легко: >>> from pymongo import MongoClient>>> client = MongoClient().

Наведений вище код підключатиметься до хосту та порту за замовчуванням. Ми також можемо вказати хост та порт явно, як показано нижче: >>> client = MongoClient('localhost', 27017). Або використовувати формат URI MongoDB:

>>> client = MongoClient('mongodb:-localhost:27017/').

2.3. Отримання даних

Один екземпляр MongoDB може підтримувати кілька незалежних баз даних. При роботі з PyMongo ви отримуєте доступ до баз даних, використовуючи доступ у стилі атрибутів до MongoClient копії модулю: >>> db = client.test_database

Якщо ім'я бази даних, що використання доступу в стилі атрибута не буде працювати (наприклад, test-database,), замість цього можна використовувати доступ у стилі словника: >>> db = client['test-database']

Отримання колекції. Колекція - це група документів, що зберігаються в MongoDB, і її можна розглядати як еквівалент таблиці в реляційній базі даних.

Отримання колекції в PyMongo працює так само, як отримання бази даних: >>> collection = db.test_collection. або (використовуючи доступ до стилю словника): >>> collection = db['test-collection'].

Важливе зауваження про колекції (і бази даних) в MongoDB полягає в тому, що вони створюються ліниво - жодна з наведених вище команд фактично не виконувала жодних операцій на сервері MongoDB. Колекції та бази даних створюються при вставці у них першого документа.

Дані MongoDB представлені (і збережені) з використанням документів у стилі JSON (рисунок 2.9). У PyMongo ми використовуємо словники для представлення документів. Як приклад можна використовувати наступний словник для представлення повідомлення у блозі.

```
>>> import datetime
>>> post = {"author": "Mike",
...         "text": "My first blog post!",
...         "tags": ["mongodb", "python", "pymongo"],
...         "date": datetime.datetime.utcnow()}
```

Рис. 2.9 – Дані MongoDB

Зверніть увагу, що документи можуть містити власні типи Python (наприклад, datetime.datetime, екземпляри), які будуть автоматично перетворені на відповідні типи BSON і назад.

Вставка документа

Щоб вставити документ у колекцію, ми можемо використовувати insert_one()метод:

>>> posts = db.posts

>>> post_id = posts.insert_one(post).inserted_id

>>> post_id

ObjectId('...')

Коли документ вставляється "_id", автоматично додається спеціальний ключ, якщо документ ще не містить "_id" ключа. Значення "_id" має бути унікальним у колекції. insert_one() повертає екземпляр InsertOneResult.

Після вставки першого документа, колекція постів фактично створена на сервері. Ми можемо переконатися в цьому, перерахувавши всі колекції нашої бази даних, рисунок 2.10.

```
>>> db.list_collection_names()
['posts']
```

Рис. 2.10 – Список колекцій

Отримання єдиного документа за допомогою find_one(), рисунок 2.11.

```
>>> import pprint
>>> pprint.pprint(posts.find_one())
{'_id': ObjectId('...'),
 'author': 'Mike',
 'date': datetime.datetime(...),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
```

Рис. 2.11 – Отримання одного документа колекції

Найпростіший тип запиту, який можна виконати в MongoDB, це find_one(). Цей метод повертає один документ, що відповідає запиту (або None якщо збігів немає).Це корисно, коли ви знаєте, що є лише один збігаючий документ або вас цікавить лише перший збіг. Тут ми використовуємо find_one(), щоб отримати перший документ із колекції повідомлень, рисунок 2.12.

```
>>> import pprint
>>> pprint.pprint(posts.find_one())
{'_id': ObjectId('...'),
 'author': 'Mike',
 'date': datetime.datetime(...),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
```

Рис. 2.12 – Перший документ з колеекції

Масові вставки. Крім вставки одного документа, ми можемо також виконувати масові операції вставки, передаючи список як перший аргумент функції insert_many(). Це вставить кожен документ у список, надіславши на сервер лише одну команду, рисунок 2.13.

```
>>> new_posts = [{"author": "Mike",
...               "text": "Another post!",
...               "tags": ["bulk", "insert"],
...               "date": datetime.datetime(2009, 11, 12, 11, 14)},
...              {"author": "Eliot",
...               "title": "MongoDB is fun",
...               "text": "and pretty easy too!",
...               "date": datetime.datetime(2009, 11, 10, 10, 45)}]
>>> result = posts.insert_many(new_posts)
>>> result.inserted_ids
[ObjectId('...'), ObjectId('...')]
```

Рис. 2.13 – Масові встаки

У цьому прикладі є кілька цікавих моментів.

Результат з insert_many() цього моменту повертає два ObjectId, по одному для кожного вставленого документа.

New_posts[1] має іншу "форму", ніж інші повідомлення - тут немає "tags" поля, і ми додали нове поле, "title". Саме це ми маємо на увазі, коли говоримо, що MongoDB не містить схеми. Запит більш ніж одного документа.

Щоб отримати більше одного документа в результаті запиту, ми використовуємо метод find(). find() повертає Cursor примірник, який дозволяє нам

перебирати всі відповідні документи. Наприклад (рисунок 2.14), ми можемо перебрати кожен документ у posts колекції.

```
>>> for post in posts.find():
...   pprint.pprint(post)
...
{'_id': ObjectId('...'),
 'author': 'Mike',
 'date': datetime.datetime(...),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
{'_id': ObjectId('...'),
 'author': 'Mike',
 'date': datetime.datetime(...),
 'tags': ['bulk', 'insert'],
 'text': 'Another post!'}
{'_id': ObjectId('...'),
 'author': 'Eliot',
 'date': datetime.datetime(...),
 'text': 'and pretty easy too!',
 'title': 'MongoDB is fun'}
```

Рис. 2.14 – Отримання більше одного документа з колекції

Як і у випадку з find_one(), ми можемо передати документ, find() щоб обмежити результати, що повертаються (рисунок 2.15). Тут ми отримуємо ті документи, автором яких є «Майк».

```
>>> for post in posts.find({"author": "Mike"}):
...   pprint.pprint(post)
...
{'_id': ObjectId('...'),
 'author': 'Mike',
 'date': datetime.datetime(...),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
{'_id': ObjectId('...'),
 'author': 'Mike',
 'date': datetime.datetime(...),
 'tags': ['bulk', 'insert'],
 'text': 'Another post!'}
```

Рис. 2.15 – Отримання всіх документів по певному параметру

Якщо ми просто хочемо дізнатися скільки документів відповідає запиту, ми можемо виконати count_documents()операцію замість повного запиту. Ми можемо отримати кількість усіх документів у колекції:

>>> posts.count_documents({})3

або лише з тих документів, які відповідають конкретному запиту:

>>> posts.count_documents({"author": "Mike"})2.

2.4. Огляд бібліотеки ITT Python

Опис інтерфейсу бібліотеки наведений у файлі EUSignCP.py. Файл _EUSignCP.pyd містить скомпільований код для роботи з бібліотекою підпису на мові Python.

Бібліотека завантажується та звільняється за допомогою функцій EULoad та EUUnload.

Функція завантаження бібліотеки. Повертає TRUE у разі успіху, та FALSE, якщо виникає помилка: def EULoad() -> Boolean.

Функція звільнення бібліотеки: def EUUnload().

Формальний опис інтерфейсу програмування бібліотеки наведений на мові Python.

Інтерфейс програмування бібліотеки представлений у вигляді класу EU_INTERFACE, яка містить доступні функції роботи з бібліотекою. Отримати інтерфейс бібліотеки, можна за допомогою функції EUGetInterface(). Коди помилок.

Функції у разі успішного виконання повертають результат виконання операції, а у разі виникнення помилки генерують виняток (exception) з описом помилки. Опис помилок та можливі причини виникнення наведені у таблиці. 2.1

Таблиця 2.1 – Опис помилок та можливі причини виникнення

Назва	Опис	Можливі причини виникнення
EU_ERROR_UNKNOWN	Невідома помилка	Відмова бібліотеки
EU_ERROR_NOT_SUPPORTED	Бібліотеку не ініціалізовано	Не був викликаний метод Initialize
EU_ERROR_NOT_INITIALIZED	Операція не підтримується	
EU_ERROR_BAD_PARAMETER	Невірний параметр	Переданий у виклик метода параметр має невірний формат
EU_ERROR_LIBRARY_LOAD	Виникла помилка при завантаженні базових бібліотек	Одна з базових бібліотек не завантажена або виникла помилка при її ініціалізації
EU_ERROR_READ_SETTINGS	Виникла помилка при зчитуванні параметрів з системного реєстру	Параметри не встановлені або пошкоджені чи переданий неправильний шлях їх розміщення у системному реєстрі
EU_ERROR_TRANSMIT_REQUEST	Виникла помилка при передачі запиту на сервер ЦСК за протоколом HTTP	Сервер ЦСК не доступний або не пройдено автентифікацію на proxy-сервері
EU_ERROR_MEMORY_ALLOCATION	Виникла помилка при виділенні пам'яті	

Продовження таблиці 2.1.

EU_WARNING_END_OF_ENUM	Перелічення закінчено	Не є помилкою. Повідомляє про завершення списку переліку.
EU_ERROR_PROXY_NOT_AUTHORIZED	Автентифікація на proxy-сервері не можлива	Параметри автентифікації на proxy-сервері не встановлені в реєстрі або автентифікацію не пройдено
EU_ERROR_NO_GUI_DIALOGS	Діалог з оператором не підтримується	Виникла необхідність виведення діалогу з оператором який не підтримується
EU_ERROR_DOWNLOAD_FILE	Виникла помилка при завантаженні файлу з HTTP-сервера	Сервер ЦСК не доступний (проблеми з комунікаційними засобами) або не пройдено автентифікацію на proxy-сервері
EU_ERROR_WRITE_SETTINGS	Виникла помилка при записі параметрів у системний реєстр	
EU_ERROR_CANCELED_BY_GUI	Операція відмінена оператором	Не виникає за відсутності діалогів з оператором
EU_ERROR_OFFLINE_MODE	Доступ до сервера ЦСК не можливий (увімкнено offline-режим)	
EU_ERROR_KEY_MEDIAS_FAILED	Виникла помилка при роботі з носіями ключової інформації	Відмова бібліотеки роботи з НКІ

Продовження таблиці 2.1.

EU_ERROR_KEY_MEDIAS_ACCESS_FAILED	Виникла помилка при доступі до носія ключової інформації	Не пройдено автентифікацію на НКІ або НКІ нероботоспроможний
EU_ERROR_KEY_MEDIAS_READ_FAILED	Виникла помилка при зчитуванні особистого ключа з носія ключової інформації	Особистий ключ на НКІ відсутній або пошкоджений
EU_ERROR_KEY_MEDIAS_WRITE_FAILED	Виникла помилка при записі особистого ключа на носій ключової інформації	
EU_WARNING_KEY_MEDIAS_READ_ONLY	Носій ключової інформації не підтримує знищення даних	
EU_ERROR_KEY_MEDIAS_DELETE	Виникла помилка при видаленні особистого ключа з носія ключової інформації	
EU_ERROR_KEY_MEDIAS_CLEAR	Виникла помилка при очищенні носія ключової інформації	
EU_ERROR_BAD_PRIVATE_KEY	Виникла помилка при відкритті особистого ключа (невірний пароль чи ключ пошкоджений)	Особистий ключ зчитаний з носія або пошкоджений або вказано невірний пароль його захисту
EU_ERROR_PKI_FORMATS_FAILED	Виникла помилка при розборі даних (пошкоджені дані чи невірний формат)	В залежності від контексту операції – пошкоджений підпис чи зашифровані дані

Продовження таблиці 2.1.

EU_ERROR_CSP_FAILED	Виникла помилка при виконанні криптоперетворень	Дані пошкоджені. Рівень пошкодження не виявлений при розборі формату даних
EU_ERROR_BAD_SIGNATURE	Невірний підпис	Підписані дані модифіковано або модифіковано сам підпис
EU_ERROR_AUTH_FAILED	Виникла помилка при автентифікації (дані автентифікації пошкоджені)	Дані автентифікації пошкоджені
EU_ERROR_NOT_RECEIVER	Власник особистого ключа відсутній у списку одержувачів зашифрованих даних	Користувач-власник проточного зчитаного особистого ключа не може розшифрувати дані, тому що вони були зашифровані на іншого користувача
EU_ERROR_STORAGE_FAILED	Виникла помилка при роботі з файловим сховищем сертифікатів та CBC	Не вірно вказаний каталог файлового сховища або каталог не існує
EU_ERROR_BAD_CERT	Сертифікат пошкоджений	Сертифікат пошкоджений або має не вірний формат
EU_ERROR_CERT_NOT_FOUND	Сертифікат не знайдено	Сертифікат не знайдено жодними з доступних засобів. Послідовність пошуку – файлове сховище, протокол OCSP, LDAP-каталог

Продовження таблиці 2.1.

EU_ERROR _INVALID_CERT _TIME	Сертифікат не чинний за строком дії	Строк чинності сертифіката вже завершився або ще не наступив
EU_ERROR_CER T_IN_CRL	Сертифікат не чинний (при перевірці за допомогою СВС)	Сертифікат заблокований чи скасований
EU_ERROR_BAD _CRL	СВС пошкоджений	Один з СВС у ланцюжку пошкоджений або має не вірний формат
EU_ERROR_NO_ VALID_CRLS	Не знайдено діючих СВС	У файловому сховищі не знайдено діючих СВС
EU_ERROR_GET _TIME_STAMP	Виникла помилка при отриманні позначки часу	TSP-сервер не доступний (проблеми з комунікаційними засобами) або не пройдено автентифікацію на proxy-сервері
EU_ERROR_BAD _TSP_RESPONSE	Відповідь від TSP-сервера пошкоджена	
EU_ERROR_TSP _SERVER _CERT_NOT_FO UND	Сертифікат TSP-сервера не знайдено	Сертифікат TSP-сервера, яким підписано отриману позначку часу не знайдено у файловому сховищі
EU_ERROR_TSP_ SERVER_CERT_I NVALID	Сертифікат TSP-сервера не чинний	Сертифікат TSP-сервера, яким підписано отриману позначку часу не чинний

Продовження таблиці 2.1.

EU_ERROR_GET_OCSP_STATUS	Виникла помилка при спробі отримати статус за протоколом OCSP	OCSP-сервер не доступний (проблеми з комунікаційними засобами) або не пройдено автентифікацію на proxy-сервері
EU_ERROR_BAD_OCSP_RESPONSE	Відповідь від OCSP-сервера пошкоджена	
EU_ERROR_CERT_BAD_BY_OCSP	Сертифікат не чинний (при перевірці за протоколом OCSP)	Сертифікат, статус якого визначався за допомогою протоколу OCSP
EU_ERROR_OCSP_SERVER_CERT_NOT_FOUND	Сертифікат OCSP-сервера не знайдено	Сертифікат OCSP-сервера, яким підписано інформацію про статус сертифіката не знайдено у файловому сховищі
EU_ERROR_OCSP_SERVER_CERT_INVALID	Сертифікат OCSP-сервера не чинний	Сертифікат OCSP-сервера, яким підписано інформацію про статус сертифіката не чинний
EU_ERROR_LDAP_ERROR	Виникла помилка при роботі з LDAP-сервером	LDAP-сервер не доступний (проблеми з комунікаційними засобами)

Розглянемо константи бібліотеки. Спосіб отримання параметрів НКІ:

- EU_KEY_MEDIA_SOURCE_TYPE_OPERATOR 1 - Запитувати параметри НКІ у оператора;

- EU_KEY_MEDIA_SOURCE_TYPE_FIXED 2 - Використовувати фіксовані;
- EU_CERT_INFO_VERSION 1 - Версія структури з детальною;
- EU_CERT_INFO_EX_VERSION 5 - Версія структури з розширеною інформацією про сертифікат;
- EU_CRL_DETAILED_INFO_VERSION 1 - Версія структури з детальною;
- EU_CR_INFO_VERSION 3 - Версія структури з інформацією про запит на отримання сертифікат;
- EU_USER_INFO_VERSION 3 - Версія структури з інформацією про користувача;
- EU_TIME_INFO_VERSION 1 - Версія структури з інформацією про час;
- EU_KEYS_TYPE_RSA_WITH_SHA 2 - Ключ для використання в алгоритмі ЕЦП RSA з функцією гешування SHA. Довжини ключів за алгоритмом ЕЦП ДСТУ-4145-2002;
- EU_KEYS_LENGTH_DS_UA_191 1 - Довжина ключа 191 біт;
- EU_KEYS_LENGTH_DS_UA_257 2 - Довжина ключа 257 біт;
- EU_KEYS_LENGTH_DS_UA_307 3 - Довжина ключа 307 біт;
- EU_KEYS_LENGTH_DS_UA_FILE 4 - Довжина ключа обирається з файлу параметрів;
- EU_KEYS_LENGTH_DS_UA_CERT 5 - Довжина ключа обирається з - сертифікату діючого ОС.

Всі ключі підтримується лише функцією перегенерації ОС – ключа. Довжини ключів протоколу розподілу ключів за алгоритмом Діффі-Гелмана групі точок ЕК:

- EU_KEYS_LENGTH_KEP_UA_257 1 - Довжина ключа 257 біт;
- EU_KEYS_LENGTH_KEP_UA_431 2 - Довжина ключа 431 біт;
- EU_KEYS_LENGTH_KEP_UA_571 3 - Довжина ключа 571 біт;

- EU_KEYS_LENGTH_KEP_UA_FILE 4 - Довжина ключа обирається
  - з файлу параметрів;
- EU_KEYS_LENGTH_KEP_UA_CERT 5 - Довжина ключа обирається з

- сертифікату діючого ос.

Ключі підтримується лише функцією пере генерації ОС – ключа довжини ключів за алгоритмом ЕЦП RSA:

- EU_KEYS_LENGTH_DS_RSA_1024 1 - Довжина ключа 1024 біта;
- EU_KEYS_LENGTH_DS_RSA_2048 2 - Довжина ключа 2048 біта;
- EU_KEYS_LENGTH_DS_RSA_3072 3 - Довжина ключа 3072 біта;
- EU_KEYS_LENGTH_DS_RSA_4096 4 - Довжина ключа 4096 біта;
- EU_KEYS_LENGTH_DS_RSA_FILE 5 - Довжина ключа обирається
  - з файлу параметрів;
- EU_KEYS_LENGTH_DS_RSA_FILE 6 - Довжина ключа обирається з
  - сертифікату діючого ос.

Ключі підтримується лише функцією перегенерації ос. – ключа значення алгоритмів захисту даних з використанням алгоритму направленого шифрування RSA:

- EU_CONTENT_ENC_ALGO_TDES_CBC 4 - Алгоритм TDES-CBC;
- EU_CONTENT_ENC_ALGO_AES_128_CBC 5 - Алгоритм AES-128-CBC;
- EU_CONTENT_ENC_ALGO_AES_192_CBC 6 - Алгоритм AES-192-CBC;
- EU_CONTENT_ENC_ALGO_AES_256_CBC 7 - Алгоритм AES-256-CBC.

Мови, які підтримуються для локалізованих повідомлень та помилок бібліотеки.

- EU_DEFAULT_LANG 0 - Мова за замовчуванням
- EU_UA_LANG 1 - Українська мова
- EU_RU_LANG 2 - Російська мова
- EU_EN_LANG 3 - Англійська мова

Параметри конфігурації роботи криптографічної бібліотеки, управління функціями, що повертають інформацію про сертифікат:

- EU_RESOLVE_OIDS_PARAMETER "ResolveOIDs" – Визначає необхідність розшифровувати OID, за замовчуванням TRUE;
- EU_RESOLVE_OIDS_PARAMETER_LENGTH 4 - Довжина параметру управління збереженням налаштувань до системного реєстру (або файлу). Не впливає на графічну функцію EUSetSettings. За замовчуванням встановлений;
- EU_SETTINGS_ID_ALL. Функції EUInitialize, EUFinalize та EUSetSettingsPath - встановлюють його в значення EU_SETTINGS_ID_ALL.
- EU_SAVE_SETTINGS_PARAMETER "SaveSettings"
- EU_SAVE_SETTINGS_PARAMETER_LENGTH 4 - Управління генерацією особистого ключі. Якщо TRUE особистий ключ генерується у вигляді PKCS#12 контейнеру;
- EU_MAKE_PKEY_PFX_CONTAINER_PARAMETER "MakePKeyPFXContainer";
- EU_MAKE_PKEY_PFX_CONTAINER_LENGTH 4 - Ідентифікатори налаштувань бібліотеки;
- EU_SETTINGS_ID_NONE 0x000 – Жодний;
- EU_SETTINGS_ID_MANDATORY 0x01F - Обов’язкові;
- EU_SETTINGS_ID_ALL 0x7FF – Всі;
- EU_SETTINGS_ID_FSTORE 0x001 - Файлове сховище;
- EU_SETTINGS_ID_PROXY 0x002 - Proxy – сервер;
- EU_SETTINGS_ID_TSP 0x004 - TSP – сервер;
- EU_SETTINGS_ID_OCSP 0x008 - OCSP – сервер;
- EU_SETTINGS_ID_LDAP 0x010 - LDAP – сервер;
- EU_SETTINGS_ID_MODE 0x020 - Взаємодії з серверами ЦСК;

- EU_SETTINGS_ID_CMP 0x040 - CMP – сервер;
- EU_SETTINGS_ID_KM 0x080 - Носія особистого ключа;
- EU_SETTINGS_ID_OCSP_ACCESS_INFO_MODE 0x100 - Параметри точок доступу.

До OCSP – серверів - EU_SETTINGS_ID_OCSP_ACCESS_INFO 0x200 Точки доступу до OCSP – серверів. Максимальний розмір параметру, що може бути збережено до сховища: EU_STORAGE_VALUE_MAX_LENGTH 0x7FFF

Інформація про доступність серверу OCSP:

- EU_OCSP_SERVER_STATE_UNKNOWN 0 - Не визначено;
- EU_OCSP_SERVER_STATE_AVAILABLE 1 – Доступний;
- EU_OCSP_SERVER_STATE_UNAVAILABLE 2 - Не доступний;
- EU_RESOLVE_OIDS_ CONTEXT_PARAMETER_LENGTH 4- Довжина параметру.

Управління функціями, що експортують особистий ключ. Можливі значення TRUE = 1, або FALSE = 0. EU_ EXPORATABLE _ CONTEXT _ CONTEXT _PARAMETER "ExportableContext". Визначає призначення контексту для експорту ключів EU_EXPORATABLE_CONTEXT_CONTEXT_PARAMETER_LENGTH 4-Довжина параметру. Типи розділів реєстру:

- EU_REG_KEY_ROOT_PATH_DEFAULT 0 - За замовчанням;
- EU_REG_KEY_ROOT_PATH_HKLM 1 - Поточного комп’ютера;
- EU_REG_KEY_ROOT_PATH_HKCU 2 - Поточного користувача;
- EU_REG_KEY_ROOT_PATH_CURRENT 3 – Поточні граничні значення тегів для запису даних на пристрій;
- EU_DEV_CTX_MIN_PUBLIC_DATA_ID 0x10 - Мінімальне значення тега відкритих даних;
- EU_DEV_CTX_MAX_PUBLIC_DATA_ID 0x4F - Максимальне значення тега відкритих даних;

- EU_DEV_CTX_MIN_CONST_PUBLIC_DATA_ID 0x50 Мінімальне значення тегах незмінних відкритих даних;
- EU_DEV_CTX_MAX_CONST_PUBLIC_DATA_ID 0x6F - Максимальне значення тег незмінних відкритих даних;
- EU_DEV_CTX_MIN_CONST_PRIVATE_DATA_ID 0x70 - Мінімальне значення тегах незмінних особистих даних.

2.5. База даних розробки

База даних програми для підпису документів складається з трьох таблиць, а саме:

- Admin;
- Users;
- DocumentPodpis.

Схема всіх таблиць зображена на рисунку 2.16.

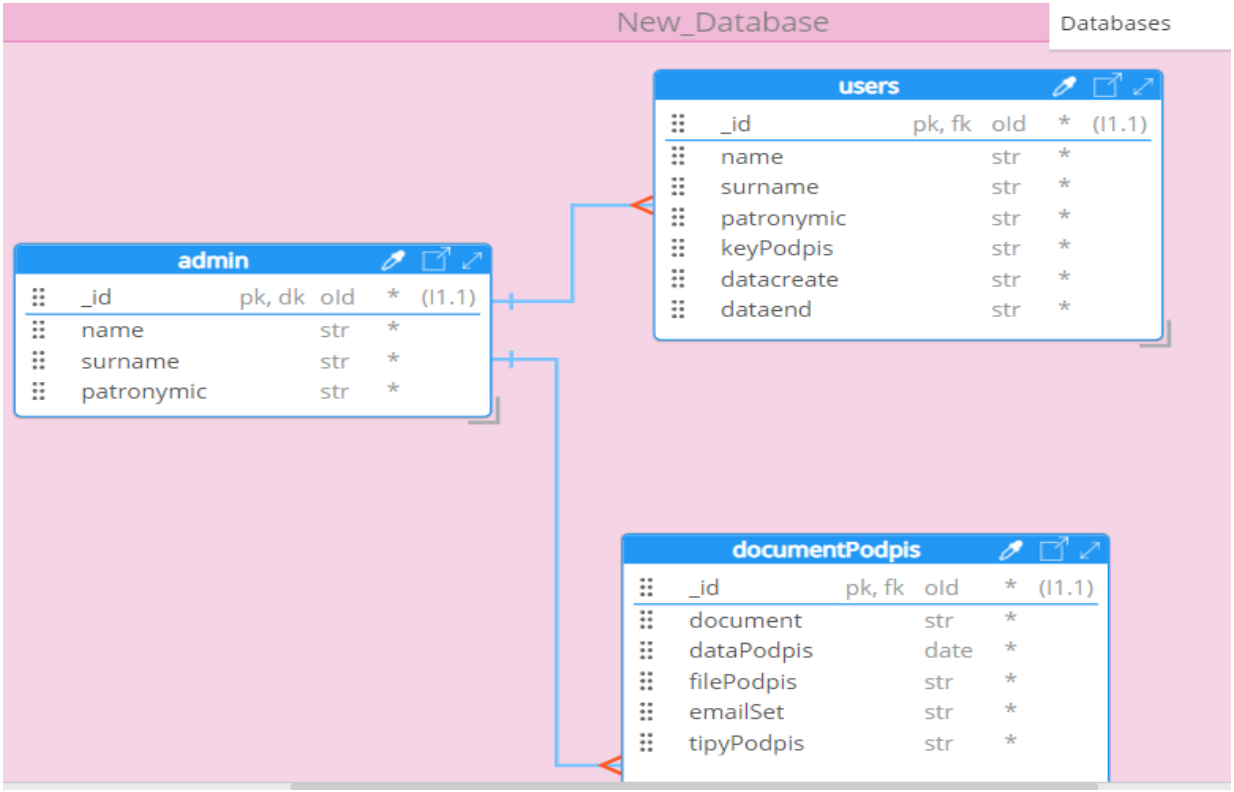


Рис. 2.16 – Вид схеми бази даних розробки

Перша таблиця admin складається з чотирьох документів. Документ – це поля як в раціонних таблицях, а таблиці – це колекції в МонгоДБ.

Перша таблиця під назвою «Admin» складеться з:

- "_id"- це унікальний ключ документа, який має тип об’єкту всі інші поля мають тип string;
- "name" - ім’я адміністратора програми;.
- "surname" - прізвище адміністратора програми;
- "patronymic" - по батькові адміністратора програм.

Друга таблиця за назвою «Users» складається з:

- "_id", " - унікальний ключ документа, який має тип об’єкт всі інші поля мають тип string;
- "name" - ім’я користувача програми;
- "surname" - призвище користувача програми;
- "patronymic" - побатькові користувача програм;

- "keyPodpis", - унікальне шістнадцятирічне значення яке записано в самому ключу підпису і видається безпекою АТ «МетаБанк»;
- "datacreate", - дата видачи ключа;
- "dataend" -дата завершення ключа.

Третя таблиця під назвою «DocumentPodpis» складається з:

- "_id", - унікальний ключ документа, який має тип об’єкт;
- "document", - поле в якому зберігається не підписаний документ має тип string;
- "dataPodpis", - поле відповідає коли було підписано документ має тип data;
- "filePodpis", - файл який вже підписано и має тип string;
- "emailSet" - пошта на яку було надіслано підписаний документ має тип string;
- "tipyPodpis" – це тип ключа за допомоги якого було підписано документ має тип string.

В другому розділі диплому було не все описано, так як є банківська таємниця про не розголошення банківської інформації яка веде за собою адміністративне право порушення. В Цьому розділі було не вказано детальний опис роботи з базою даних та його код і не було вказано про детальне використання бібліотеки ІТТ користувача Python.

РОЗДІЛ 3
ПРОГРАМНИЙ ПРОДУКТ ДОКУМЕНТІВ АТ «МЕТАБАНК»

3.1 Інтерфейс програми та діалогові вікна

Розроблений програмний продукт для документообігу зі складання договорів клієнтів АТ «МетаБанк» - складається з трьох діалогових вікон, а саме:

- Головне вікно – Qt Designer, рисунок 3.1;
- Діалогове вікно, рисунок 3.2;
- Допоміжне вікно, рисунок 3.3.

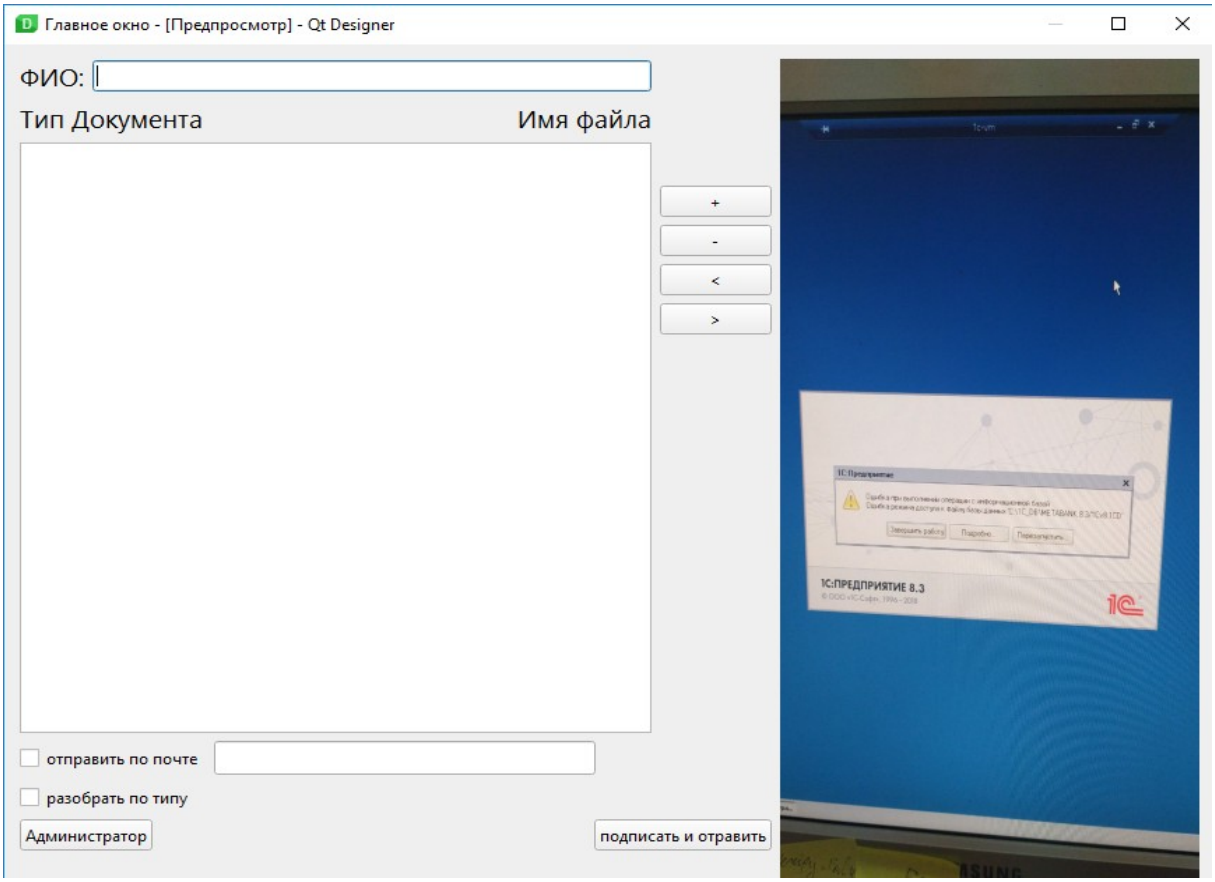


Рис. 3.1 – Головне вікно програми

Головне вікно складається з:

- ПІБ;
- Додати документ;
- Видалити документ;
- Перенести документ вгору;
- Перенести документ вниз;
- Поле для відображення документу;
- Підписання документа;
- Відправка підписаних документів поштою;
- Поле для відображення списку документів.

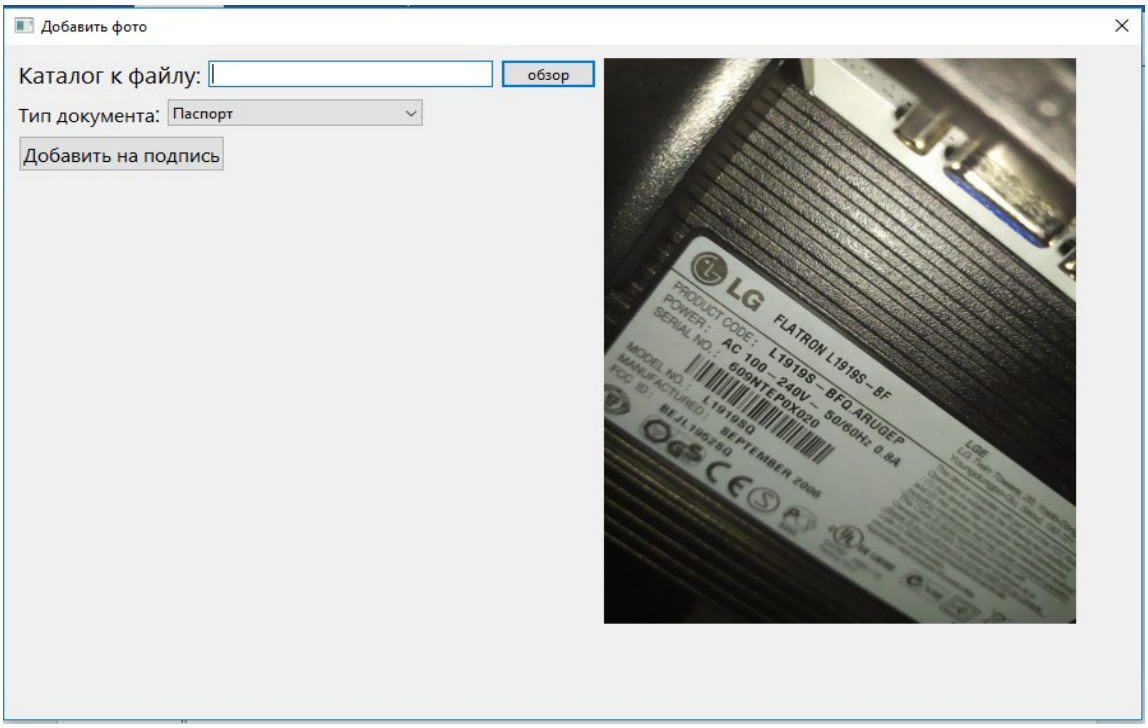


Рис. 3.2 – Діалогове вікно програми з додавання відповідного фото наприклад: номеру договору

Діалогове вікно складається з:

- Поле документа;
- Поле для вводу шляху до документа;
- Кнопка «Обзір», тип документа, кнопка додати до підпису.

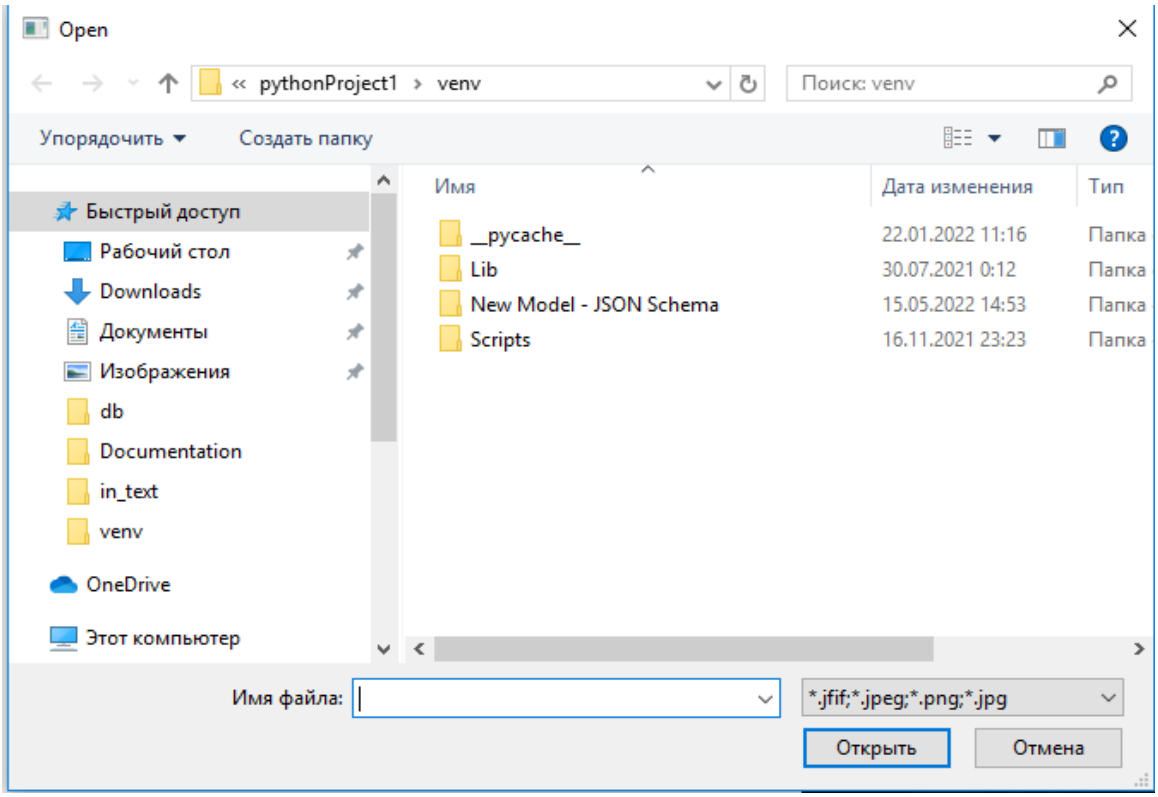


Рис. 3.3 – Допоміжне вікно програми

Розглянемо більш детально деякі елементи. Головне вікно є адаптивним до різних платформах операційної системи комп’ютера завдяки бібліотеки PyQt яка допомагає це реалізувати у вигляді коду:

- Form.resize(883, 477);
- self.gridLayout = QtWidgets.QGridLayout(Form);
- self.gridLayout.setObjectName("gridLayout");
- self.verticalLayout_4 = QtWidgets.QVBoxLayout();
- self.verticalLayout_4.setObjectName("verticalLayout_4");
- self.gridLayout_2 = QtWidgets.QGridLayout();
- self.gridLayout_2.setObjectName("gridLayout_2");
- self.verticalLayout_7 = QtWidgets.QVBoxLayout();
- self.verticalLayout_7.setObjectName("verticalLayout_7");
- self.horizontalLayout_2 = QtWidgets.QHBoxLayout();

- self.horizontalLayout_2.setObjectName("horizontalLayout_2");
- self.verticalLayout_2 = QtWidgets.QVBoxLayout();
- self.verticalLayout_2.setObjectName("verticalLayout_2");
- self.horizontalLayout_fio_2 = QtWidgets.QHBoxLayout();
- self.horizontalLayout_fio_2.setObjectName("horizontalLayout_fio_2");
- self.label_fio = QtWidgets.QLabel(Form);
- self.label_fio.setObjectName("label_fio").

Кнопки для управління документами в списку виконано за допомогою коду, рисунок 3.4.

```
def __up(self):
    row = self.listWidget.currentRow()
    if row >= 1:
        # Remove the currently selected item.
        item = self.listWidget.takeItem(row)
        self.listWidget.insertItem(row - 1, item)
        self.listWidget.setCurrentItem(item)
        self.listWidget.repaint()
def __Down(self):
    row = self.listWidget.currentRow()
    if row <= self.listWidget.count():
        item = self.listWidget.takeItem(row)
        self.listWidget.insertItem(row + 1, item)
        self.listWidget.setCurrentItem(item)
        self.listWidget.repaint()
def remov(self):
    item = self.listWidget.takeItem(self.listWidget.currentRow())
    item = None
def listW(self):
    if self.listWidget.currentItem().text().find(":") != -1:
        self.label_foto.setPixmap(QtGui.QPixmap(self.listWidget.currentItem().text().split("    ")[1]))
```

Рис.3.4 – Код кнопок управління

Відправка підписаних документів виконана за допомогою коду.

Лістинг коду 3.1 – Код файлового запиту

```
if self.checkBox.checkState().value != 0:
```

```
            if self.lineEdit_fio_2.text() !="":
                self.file = []
                print(self.file)
                zip = zipfile.ZipFile(self.lineEdit_fio_2.text() + '.zip', 'w')
                for x in range(self.listWidget.count()):
                    first_path = str(self.listWidget.item(x).text().split("   ")[1])
                    zip.write(first_path, basename(first_path))
                zip.close()
                msg = MIMEMultipart()
                msg['Subject'] = 'Test'
                msg['From'] = 's.morev@mbank.com.ua'
                msg['To'] = 's.morev@mbank.com.ua'
                body = MIMEText("""FFFFeffgfgfg""")
                msg.attach(body)
                filename = self.lineEdit_fio_2.text() + '.zip'
                fp = open(filename, 'rb')
                att = email.mime.application.MIMEApplication(fp.read(),
_subtype="cfg")
                fp.close()
                att.add_header('Content-Disposition', 'attachment', filename=filename)
                msg.attach(att)
                if self.lineEdit_mail.text() != "":
                try:
                v = validate_email(self.lineEdit_mail.text())  # validate and get info
                    email1 = v["email"]  # replace with normalized form
                    email1 = "true"
                    s = smtplib.SMTP('smtp.mbank.zp.ua: 25')
```

```
                    s.sendmail('s.morev@mbank.com.ua', [self.lineEdit_mail.text()],
msg.as_string())
                    s.quit()
                except EmailNotValidError as e:
                    # email is not valid
                    msgBox = QMessageBox()
                    msgBox.setWindowTitle("Ошибка")
                    msgBox.setText("Укажите верный Email")
                    msgBox.setStyleSheet("QLabel{min-width: 200px;}")
                    msgBox.show()
                     returnValue = msgBox.exec()
            else:
                msgBox = QMessageBox()
                msgBox.setWindowTitle("Ошибка")
                msgBox.setText("Укажите Email!")
                msgBox.setStyleSheet("QLabel{min-width: 200px;}")
                msgBox.show()
                returnValue = msgBox.exec()
        else:
            msgBox = QMessageBox()
            msgBox.setWindowTitle("Ошибка")
            msgBox.setText("Укажите ФИО!")
            msgBox.setStyleSheet("QLabel{min-width: 200px;}")
            msgBox.show().
```

При кликані на кнопку адміністратор, відкривається вікно для вводу ключа адміністратора який раніше видала внутрішня служба безпеки банку, показано на рисунку 3.5.

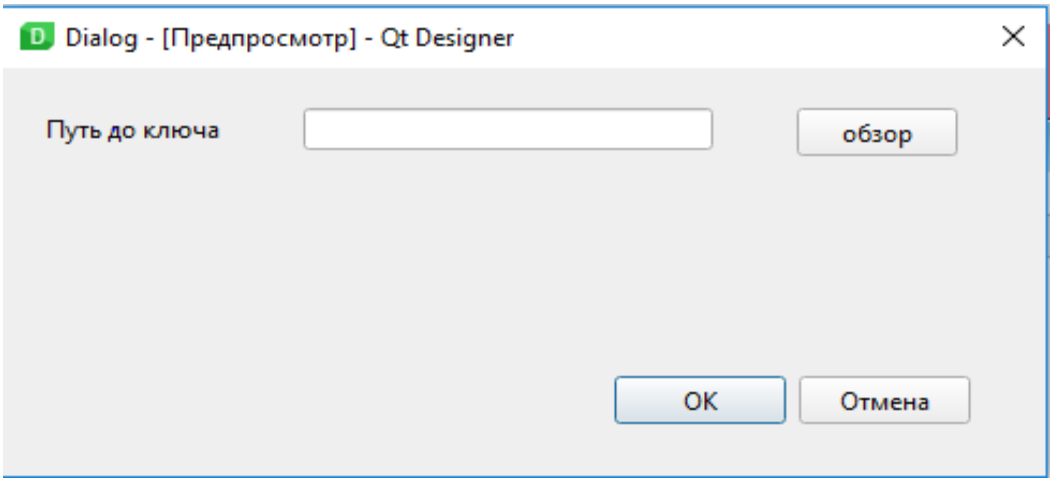


Рис. 3.5 – Ідентифікація адміністратора

Потім якщо все успішно то відображається результат запиту до бази даних у вигляді звітно-статистичної таблиці з відображенням певних даних, приклад скріну таблиці показано на рисунку 3.6.

	ФИО	Дата окончания ключа	Дата выдачи ключа	Ключ подписи	файл	Email получателя	Тип подписи
1	Морев Сергей Александровыч	28.12.2022	28.12.2021	F2EA12B46M	test.pdf	s.morev@mbank.zp.ua	p7z
2	Пак Татьяна Николаевна	01.10.2022	01.10.2021	D4E7645AAC	документ ...	s.morev@mbank.zp.ua	p7z
3	Петрусенко Сергей Анатолевыч	01.10.2022	01.10.2021	AD34H2G53J	документ ...	s.morev@mbank.zp.ua	p7z

Рис. 3.6 – Результат даних після запиту до бази даних

3.2 Корпоративна таємниця АТ «МетаБанк»

Комерційна таємниця — це приватна інформація, яку важко отримати тим, хто нею не володіє. Ця інформація має комерційну цінність і зберігається в таємниці тими, хто юридично володіє інформацію. Цією таємницею можуть бути

відомості про технічний характер, організаційного, комерційного, виробничого та іншого характеру, за винятком відомостей, які згідно із законодавством не можуть бути віднесені до комерційної таємниці (Постанова Кабінету Міністрів України від 10.07.1994 р. N 612).).

Кожен працівник будучи прийнятий на роботу підписує наказ від Голови правління про нерозголошення корпоративної та банківської таємниці. Після підписання цього наказу працівник несе адміністративну та кримінальну відповідальність. В дипломній роботі використовується єлектроні ключі та куплена бібліотека «користувача ЦСК Python» від АТ «ІІТ»для роботи з цими ключами. Так як це банківська установа яка на пряму взаємодіє с Національним Банком України теж заборонено розповсюджувати базу даних акціонерного товариства МетаБанк по постанові No 93 Національним Банком України в якому прописані всі вимоги до інформаційної безпеки банків.

ВИСНОВОК

У даній бакалаврській роботі було розроблено програму для підписання банківських документів. Викладені основні теоретичні та практичні основи програмування для операційної системи Windows з використанням сучасної мови програмування Python. Результатом практичної роботи в рамках даної бакалаврської дипломної роботи було розроблено, програму метою якої є спрощення та автоматизація процесів роботи працівників банку.

Даний програмний продукт дозволяє:

- Підписання документів електронним підписом;
- Відправку підписаного документу за допомогою електронної пошти;
- Доступ до бази даних де можна дізнатися хто і коли підписав даний документ, а також кому було відправлено;
- Має зрозумілий та зручний інтерфейс користувача.

Особливу увагу під час розробки даної програми було приділено щоб користувач міг швидко і зручно виконувати свою роботу. Програмний продукт виконаний в повному обсязі, всі вимоги враховані, тестування програми виконано у відповідності до методики тестування.

Даний програмний продукт в подальшому буде розвиватися та покращуватися, відповідно до потреб замовників інтегрувати функціонал, прокрашувати візуалізацію, впроваджувати нові технології та інструменти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Allen B. Downey - Think DSP. Цифрова обробка сигналів на Python - Видавництво "DMK Press" - 2017 - 160S.
2. Programming games and puzzles: Per (англ.) 6 Франц. - М.: Наук. Гл. ред фіз.. - приятель. 1990.- 224 П
3. Beesley D. Python. Детальне посилання. С английскою. - СПб.: Символ-Плюс, 2010. - 864 р, іл.
4. Brooks F., Mythical Man-Month, або How Software Systems are Created, M.: Symbol-Plus, 2010. (Англ.)
5. Буйначев, с.к.Основи програмування мовою Python [Електронний ресурс]: Навчальний посібник / с.к.Буйначев, Боклаг Н.Ю. - Електрон. Текстовий дан. - Видавництво Уральського університету, 2014.
6. Gaddis T. Ми починаємо програмування на Python. - 4-е видання. Пер. Від укр. - СПб.: БХЦ-Петербург, 2019. - 768 р.
7. Долінський м.с розв'язання складних та олімпійських проблем з програмування - навчальний посібник - м.: - 2006.
8. Dawson M. Programmable на Python (програмування Python для Absolute Beginner)/d. «Peter», серія Bestsells O'Reilly, 2016,- 416с.
9. Dawson M. Programmable на Python. - СПб.: Питер, 2014. - 416 р.
10. Основи програмування на мові Python належать Zlatopol D.M. Basics. - М.: ДМК Пресс, 2017. - 284 р.
11. Lutz M. Ми вивчаємо Python, 4-е видання, - Per. Від Енг. - СПб.: СимвоПлюс, 2011. - 1280 с., іл.
12. Lutz M. Programming на Python, том I, 4-е видання. С английскою. - СПб.: Символ-Плюс, 2011. - 992 р.

13. Lutz M. Programming on Python, Volume II, 4-е видання. С англійскою. - СПб.: Символ-Плюс, 2011. - 992 p.
14. Лучано Рамалло Пайтон. До вершини майстерності. - М.: ДМК Пресс, 2016. - 768 p.
15. Лубанович Б. Простий Python. Modern Programming Style, (Introducing Python: Modern Computing in Simple Packages)-d. «Пітер», Найкращий Селлерс серіалу о'Рейлі, 2016,- 480 p.
16. Майк Макграт «Python Programming for Beginners» Exmo, 2015.
17. Пілігрим Марк. Занурення в Python 3 (Dive на Python 3 російською мовою)
18. Prochorenok N., Drones V. Python 3 і PyQt 5. Розробка додатків -за. «БХК-Петербург», 2016,- 832с.