*Київський національний університет імені Тараса Шевченка*

*Факультет інформаційних технологій*

*Кафедра інтелектуальних технологій*

**КУРСОВА РОБОТА**

**З АНАЛІТИКИ ДАНИХ**

**спеціальність 122 «Комп’ютерні науки»**

**освітня програма «Аналітика даних»**

на тему:

«Розробка додатку для роботи з базою даних кадрового

складу організації»

Виконав студент 2-го курсу групи АнД-21

Пивовар В. В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

підпис

Керівник:

Самохвалов Ю. Я. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис

Національна шкала: \_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

підпис (прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

підпис (прізвище та ініціали)

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

підпис (прізвище та ініціали)

Київ – 2021

**Реферат**

Курсова робота містить: 34 сторінки, 16 рисунків, 4 додатки та 5 джерел

**Метою курсової роботи** є автоматизація діяльності відділу кадрів, а саме: розробка додатку для роботи з базою даних співробітників та її аналізом.

**Об’єкт дослідження** – робота відділу кадрів організації.

**Предмет дослідження** – робота з базою даних кадрового складу організації.

**Методи дослідження** – в даній курсові роботі для досягнення мети буде використана мова програмування Python та СУБД SQLite. В результаті буде розроблено додаток, який можна використовувати для перегляду бази даних, її редагування, а також для певних статистичних розрахунків та їх візуалізації. Також проведено тестування програми на пробній базі даних.

**Зміст**

1. Вступ
2. Розділ I: Використання мови програмування Python при розробці додатку з графічним інтерфейсом для роботи з базою даних
   1. Python GUI додаток для Windows
   2. Бібліотека tkinter
   3. Розширення ttk
   4. База даних SQLite
   5. СУБД SQLite та Python, модуль sqlite3
3. Розділ II: Розробка GUI додатку на мові програмування Python для роботи з базами даних SQLite
   1. Створення вікна та розробка стартового меню
   2. Підключення бази даних до додатку
   3. Візуалізація записів бази даних в додатку
   4. Редагування записів бази даних
   5. Пошук записів у базі даних
   6. Аналіз даних у базі та його візуалізація
   7. Тестування додатку
4. Висновки
5. Список використаних джерел
6. **Вступ**

В кожній великій організації є відділ кадрів, який виконує багато функцій, зокрема обробка та аналіз даних про співробітників. Як правило, створюються БД, що зберігають інформацію про кадровий склад, а потім їх використовують для певних цілей, наприклад, для статистичного аналізу з метою оптимізації підбору нових працівників чи відслідковування витрат на заробітні плати за певний період чи умов.

Досліджувана тема є *актуальною*, адже майже у всіх організаціях існують свої бази даних зі власною специфікацією, адже для різних компаній може бути важливою абсолютно різна інформація. Щоб комфортно працювати з ними створюються й додатки, які допомагають не лише контролювати та редагувати ті чи інші записи в БД, а й обробляти потрібні дані та проводити певні статистичні підрахунки. Також в таких програмах візуалізовуються записи БД для кращого сприйняття.

*Метою роботи* є автоматизація діяльності відділу кадрів, а саме: розробка додатку для роботи з базою даних співробітників та її аналізом. Звичайно, для зручності використання така програма повинна мати графічний інтерфейс, а дані – бути візуалізованими у вигляді таблиць, графіків, діаграм різного виду тощо.

*Об’єктом дослідження* є робота відділу кадрів. Для реалізації поставленої задачі буде використано мову програмування Python з вбудованими модулями та СУБД SQLite.

1. **Розділ I: Використання мови програмування Python при розробці додатку з графічним інтерфейсом для роботи з базою даних**
   1. **Python GUI додаток для Windows**

GUI (Graphical user interface, укр. Графічний інтерфейс користувача) – це форма інтерфейсу, яка дозволяє користувачу взаємодіяти з електронним пристроєм за допомогою графічних іконок та аудіо-індикаторів замість текстових інтерфейсів. GUI-фреймворки легко доступні – вони безкоштовні і мають відкритий вихідний код. В Python є широкий спектр варіантів: від крос-платформних до заточених під конкретну платформу.[1] Вперше концепцію ГІК запропонували вчені з дослідницької лабораторії Xerox PARC у 1970-х, але отримала комерційне втілення лише в продуктах корпорації Apple Computer. В операційній системі AMIGAOS ГІК з багатозадачністю був використаний у 1985 році. Нині цей інтерфейс — стандартна складова частина більшості доступних на ринку операційних систем і застосунків.[2]

Як було зазначено раніше, мова програмування Python має досить багато різних інструментів для створення додатків з графічним інтерфейсом. Ось найпопулярніші з них:

* **Kyvi** – це безкоштовна бібліотека Python з відкритим вихідним кодом. Вона використовується для розробки мобільних та інших додатків широкого призначення з інтерфейсом. Поширюється за ліцензією MIT(група ліцензій, розроблених Массачусетським технологічним інститутом для розповсюдження вільного програмного забезпечення) і може працювати на таких платформах як: Android, IOS, Linux, macOS, Windows.
* **PyQT** – одна з найпопулярніших крос-платформних бібліотек Python, що реалізує бібліотеку Qt з однойменного фреймворка для розробки додатків. Зараз PyQT доступна на Unix / Linux, Windows, macOS і Sharp Zaurus. Однією з переваг є варіативність, адже тільки розробник вирішує, як створювати програму: за допомогою коду або середовища Qt Designer для створення візуальних елементів.[1]
* **PySide** – це ще одна бібліотека з відкритим вихідним кодом, яка працює з Qt. PySide був випущений під ліцензією LGPL в серпні 2009 компанією Nokia, колишніми власниками інструментарію Qt, після того як Nokia не змогла досягти домовленості з розробниками PyQt, компанією Riverbank Computing, для приведення умов їхньої ліцензії до сумісності з LGPL як альтернативної ліцензії.[3]
* **Tkinter** – це зв'язка Python з набором GUI-інструментів Tk. Він доступний на Linux, Windows і macOS до встановлених Python. Ім'я Tkinter походить від Tk interface. Tkinter входить до стандартної бібліотеки Python, а отже не потрібно нічого скачувати чи встановлювати, лише імпортувати бібліотеку в коді програми.
  1. **Бібліотека tkinter**

Оскільки саме tkinter буде використовуватися у ході виконання роботи для реалізації графічного інтерфейсу, в даному підрозділі детальніше зупинимося на особливостях та можливостях даної бібліотеки. Як було зазначено в підрозділі 1.1, tkinter входить до стандартної бібліотеки Python, тому для його використання потрібно лише імпортувати в програму командою import tkinter as tk(для версії Python 2.7 та нижче: import Tkinter as tk). Перед початком роботи потрібно ознайомитися з базовими поняттями та об’єктами:

* **Віджет** (англ. widget, від Window gadget) — елемент керування, стандартизований компонент графічного інтерфейсу, з яким взаємодіє користувач.
* **Клас Tk** – в основний клас застосунку Tkinter. При створенні об'єкта цього класу буде завантажено інтерпретатор і створено основне вікно програми.
* **Toplevel** – це вікно верхнього рівня, яке використовують для створення багатовіконних програм, а також вікон діалогу.
* **Canvas** (полотно) — це об'єкт класу ТCanvas для виведення зображень, які можна змінювати і переміщати у процесі виконання програми.
* **Пакувальники** (pack, place, grid) – механізм розташування (пакування) елементів керування (віджетів) на вікні.
* **Button** – кнопка.
* **Label** – звичайний напис з яким користувач ніяк не може взаємодіяти.
* **Entry** – спеціальний віджет для введення інформації користувачем.
* **Frame** – рамка для організації віджетів всередині вікна.
* **Checkbutton** – елемент списку з місцем для мітки. Значення кожної мітки (прапорця) пов'язано зі своєю змінною. Значення цієї змінної змінюється властивістю onvalue (увімкнено) і offvalue (вимкнено).
* **Radiobutton** (перемикач) – елемент списку з місцем для мітки подібний до checkbutton, але може бути обраний лише один варіант(увімкнена лише одна мітка).[4]

Однією з переваг tkinter є простота використання, адже для того щоб основу для додатку з графічним інтерфейсом, потрібно лише 3 рядки коду, і в результаті отримаємо первинне вікно програми, з яким можна далі працювати(рисунок 1):

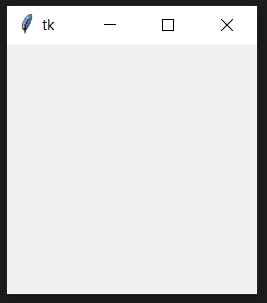


Рисунок 1 – Первинне вікно програми tkinter;

Для персоналізації вікна в tkinter існує досить багато різних параметрів, які можна змінювати під себе. Зазвичай, спочатку налаштовують розмір вікна(метод geometry() – явно задається розмір вікна у пікселях), чи можна його змінювати(resizable() – true або false для висоти та ширино вікна), назву(метод title()), колір фону(параметр ‘bg’). Для роботи з віджетами створюються об’єкти Frame() або Canvas(). До них обов’язково застосовується один з пакувальників(pack, place, grid). Виконавши всі вищеперераховані дії вже можна отримати повноцінне початкове вікно програми з потрібним розміром, кольором та іншими параметрами.

* 1. **Розширення ttk**

Розширення ttk (themed tk) – це розширення бібліотеки tkinter з новим набором віджетів. У ttk використовується новий рушій для створення віджетів. Він має підтримку тем і стилів оформлення. Завдяки цьому віджети ttk виглядають більш природньо в різних операційних системах. У ttk включені наступні віджети, які можна використовувати замість відповідних віджетів tk: Button, Checkbutton, Entry, Frame, Label, LabelFrame, Menubutton, PanedWindow, Radiobutton, Scale і Scrollbar. Крім того є кілька нових віджетів: Combobox, Notebook, Progressbar, Separator, Sizegrip і Treeview. ttk має чотири вбудованих теми: default, classic, alt, clam. Крім того додатково під Windows є теми winnative, xpnative і vista, а під Mac OS X – aqua.[4]

Окремо серед всіх особливостей ttk потрібно виділити віджети treeview та combobox, адже в ході виконання даної роботи вони будуть активно використовуватися. Додаток буде працювати з базою даних, тому необхідно візуалізувати записи БД у вигляді таблиць. Саме для такого завдання підходить treeview. Віджет combobox призначено для відображення списку значень, їх вибору або зміни користувачем. У звичайній версії tkinter є схожий віджет Listbox але різниця є. Як зазначалося раніше ttk, в першу чергу, розширює можливості програми в плані оформлення і ця ситуація не є виключенням. Різниця полягає у тому, що combobox має можливість згортатися, а Listbox буде відображено завжди відкритим, що може погіршити сприйняття інтерфейсу і зробити його менш компактним та зрозумілим.

Ще однією особливістю ttk є клас Style – спеціальний клас для роботи зі стилями й темами. Його використання відкриває нові можливості для візуального оформлення віджетів.

* 1. **База даних SQLite**

Простим словами, SQLite – це програмний пакет загального користування, який є реляційною системою управління базами даних (СУБД). Реляційні системи баз даних використовується для зберігання визначених користувачем записів у великих таблицях. Окрім зберігання та управління даними, механізм БД може обробляти різного роду запити. Популярними СУБД що реалізовуються на комерційній основі є Oracle Database, IBM DB2 та Microsoft SQL Server. Також існує велика кількість безкоштовних(з відкритим кодом) СУБД: MySQL та PostgreSQL. Частинка «Lite» в назві SQLite говорить скоріше не про обмежені можливості, а про простоту використання. Дана СУБД має ряд переваг і особливостей, які потрібно виділити:

* **Безсерверність** – це означає, що для роботи SQLite не потрібно серверів чи інших відокремлених систем. Цю особливість можна виділити як перевагу, так і недолік. З одного боку, це сильно спрощує роботу з даною СУБД адже вся інформація зберігається в одному файлі безпосередньо в системі, непотрібно нічого додатково налаштовувати. З іншого боку, обмежуються можливості роботи, тому що таку БД можна використовувати лише на локальних системах.
* **Непотрібна конфігурація/налаштування** – як було зазначено в попередньому пункті в SQLite відсутній сервер, тобто не потрібно займатися його налаштуванням, що явно спрощує роботу. Створення БД SQLite таке ж просте як і створення звичайного файлу.
* **Крос-платформа** – вся інформація зберігається лише в одному крос-платформному файлі.
* **Низька ресурсоємність** – збірка за замовчуванням містить всю необхідні дані для коректної роботи і становить менше мегабайта коду і вимагає лише кілька мегабайт пам'яті.
* **Надійність** – кажуть, що простіший механізм, то рідше він виходить з ладу. В нашому випадку ця приказка підходить ідеально. В порівнянні з іншими СУБД, SQLite набагато простіша, тому легша в обслуговуванні, оновленні, а також містить менше помилок та збоїв.

Загалом, SQLite забезпечує дуже функціональне та гнучке середовище реляційних баз даних що споживає мінімум ресурсів і створює мінімум клопоту для розробників та користувачів.[5]

SQLite пропонує кілька функцій, яких немає в багатьох інших системах баз даних. Найбільш помітна відмінність полягає в тому, що SQLite використовує систему динамічного типу для таблиць. Рушій SQLite дозволить вводити будь-яке значення майже в будь-який стовпець, незалежно від типу. Це є основним відходом від традиційних систем баз даних. Ще одна корисна функція – можливість одночасно працювати з кількома БД. Це дозволяє SQLite обробляти оператори SQL, які «перетинають» кілька баз даних. Це робить об’єднання таблиць з різних баз даних досить простим.[5]

* 1. **СУБД SQLite та Python, модуль sqlite3**

Ще однією серйозною перевагою СУБД SQLite є те, що її дуже легко зв’язати з програмою на Python, адже для роботи з БД існує спеціальний модуль – sqlite3. Оскільки він входить до стандартної бібліотеки мови Python, не потрібно нічого завантажувати, лише імпортувати модуль в програму. Свого роду, sqlite3 – це адаптер до СУБД SQLite, який переводить команди написані на Python, в команди, які розуміє SQLite і навпаки.

Модуль sqlite3 дає широкі можливості для взаємодії з БД. Щоб виконати запит потрібно застосувати метод .execute() до об’єкта cursor та ввести відповідну команду в потрійних лапках (наприклад: «cursor.execute(‘‘‘SELECT \* FROM car’’’)»). Запит SELECT обирає з таблиці БД потрібні дані. Зірочка означає, що потрібно взяти всі атрибути(стовпці) таблиці, якщо потрібно брати лише деякі стовпці, вона може бути змінена на перелік назв необхідних. Після FROM вводиться назва таблиці. Також необхідно розглянути інші основні запити:

* **Запит «CREATE TABLE»:** як можна зрозуміти, цей запит створює нову таблицю. Після нього вводиться назва і в дужках перелічуються назви атрибутів(стовпців). Наприклад: ‘‘‘CREATE TABLE car (num, brand, series, owner)’’’.
* **Запит «SELECT»:** даний запит обирає записи з БД. Вище вже було зазначено як використовується даний запит, але потрібно згадати, що його можна використовувати разом з командою WHERE після якої вказується умова. Наприклад: ‘‘‘SELECT \* FROM car WHERE brand = ‘Mazda’’’’.
* **Запит «INSERT INTO»:** даний запит вводить записи в БД. Після нього вказується назва таблиці в яку вводиться запис, дані команда VALUES і в дужках дані відповідно до атрибутів. Наприклад: ‘‘‘INSERT INTO car VALUES (‘AA 0000 AA’, ‘Mazda’, ‘3’, ‘Ivanov’)’’’.
* **Запит «UPDATE»:** даний запит оновлює записи в БД. Після нього вказується назва таблиці, в якій будуть оновлені записи. Далі команда SET і кортежі які будуть оновлені. І в кінці – WHERE після чого записується умова за якою буде відбуватися оновлення запису. Наприклад: ‘‘‘UPDARE car SET num = ‘AB 1111 BB’ WHERE owner = ‘Ivanov’’’’.
* **Запит «DELETE»:** даний запит видаляє записи в БД. Після нього вказується назва таблиці, в якій будуть видалятися записи. Далі записується команда WHERE та умови за яких будуть видалятися записи. Наприклад: ‘‘‘DETELE FROM car WHERE brand = ‘Mazda’’’’.
* **Запит «DROP TABLE »:** даний запит видаляє таблиці БД. Все що потрібно зробити – це вказати назву таблиці яку потрібно видалити. Наприклад: ‘‘‘DROP TABLE car’’’.
* **«and» та «or»:** дані команди використовуються у структурі WHERE … LIKE, при використанні кількох умов. Наприклад: ‘‘‘DETELE FROM car WHERE brand LIKE ‘Mazda’ or’ owner LIKE ‘Ivanov’’’’. Такий запит видалить всі записи де brand вказаний як ‘Mazda’ «АБО» owner як ‘Ivanov’. Аналогічно буде і з командою «and», лише «АБО» зміниться на «І».

**Висновок до розділу 1:** Як було описано у першому розділі, існує чимало інструментів для створення програм з графічним інтерфейсом для Python, такі як: Kivy, PyQT, PySide, Tkinter тощо. Саме останній буде використаний у ході реалізації завдання даної роботи. Бібліотека tkinter є досить широкою і має достатньо багато інструментів для створення інтерфейсу програми. Для того щоб зробити додаток більш високоорганізованим та зрозумілим можна використати спеціальне розширення tkinter ttk, яке заточене саме під покращення зовнішнього вигляду об’єктів, що робить їх більш природними. Наступне що було розглянуто – це SQLite. Дана СУБД є спрощеною в порівняння з іншими, наприклад MS SQL Server, але попри це має досить високий функціонал, що дозволяє реалізовувати багато завдань витрачаючи при цьому невеликі ресурси, як технічні, так і людські. Крім того, з SQLite комфортно працювати в парі з Python через модуль sqlite3, тому дана зв’язка є досить популярною і буде використана у ході виконання практичної частини даної роботи.

1. **Розділ II: Розробка GUI додатку на мові програмування Python для роботи з базами даних SQLite**
   1. **Створення вікна та розробка стартового меню**

Перед початком роботи імпортуємо бібліотеки та модулі Python, що будуть використовуватися у ході розробки програмного продукту. Вже на даному етапі зрозуміло, що будуть використані sqlite3, tkinter та розширення ttk. Інші бібліотеки імпортуємо по необхідності. Спершу, звичайно, потрібно початкове вікно додатку. Створимо об’єкт root класу tk.Tk(). Він буде кореневою частиною головного вікна програми. Задаймо йому певні параметри, а саме: через метод .title() введемо назву додатку (буде відображатися в заголовку вікна), змінимо параметр ‘bg’, що відповідає за колір, на світло-сірий, методом .geometry() визначимо розмір та відключимо можливість його змінювати за допомогою .resizable() встановивши False для висоти і ширини. В результаті отримаємо каркас початкового вікна програми з потрібними нам параметрами(рисунок 2). Крім цього, ще знадобляться фрейми. Як зазначалося в першому розділі, вони потрібні для розміщення та організації віджетів всередині вікна. Створимо два фрейми: перший – основний для візуалізації даних, а другий для кнопок меню. Назвемо їх відповідно main та toolbar. Вони будуть розташовуватися на об’єкті root. Задаймо їм параметри і розташуємо на вікні методами .place() і .pack().

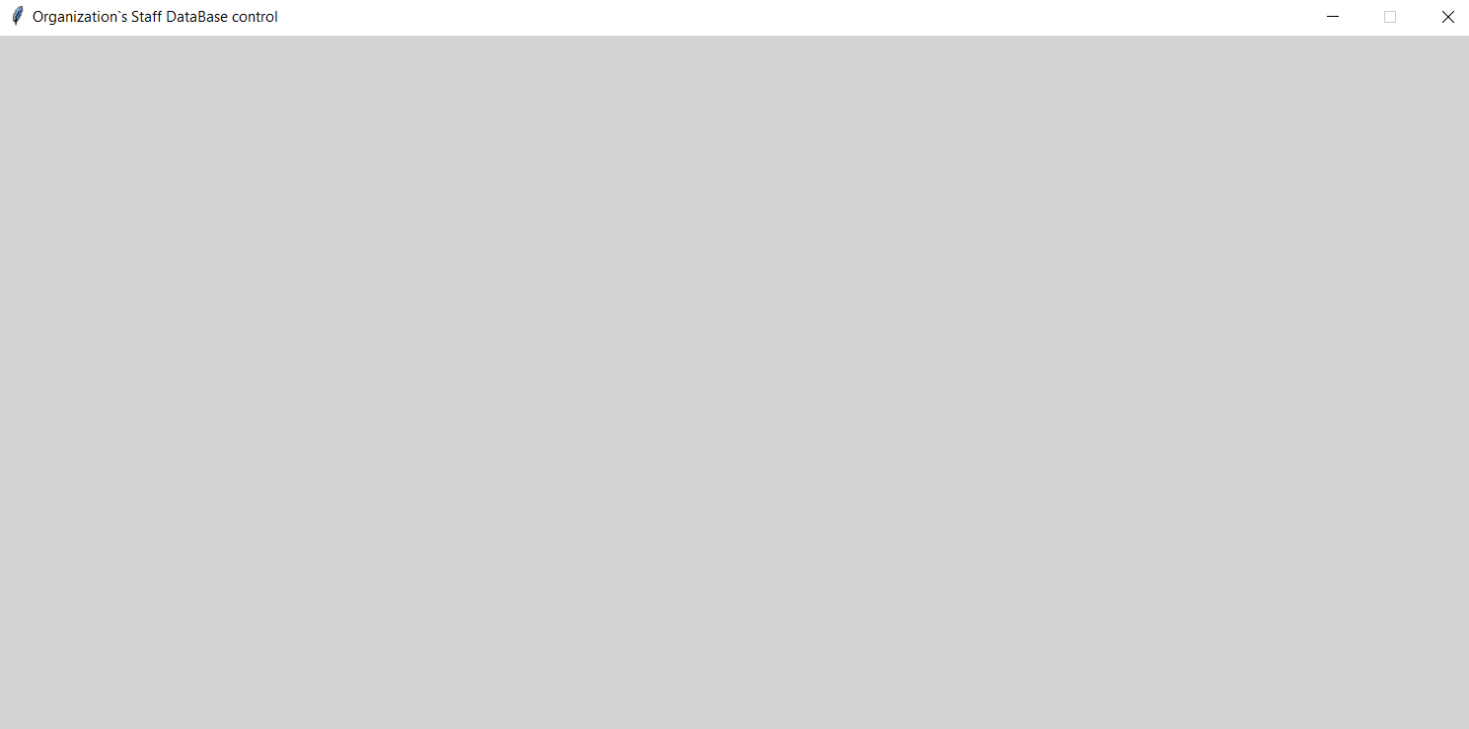


Рисунок 2 – Каркас початкового вікна програми;

Тепер, коли каркас програми готовий можна приступати до розробки стартового меню, з яким буде взаємодіяти користувач. Оскільки додаток буде побудований таким чином, що буде можливість повертатися на початковий екран, реалізуємо структуру наступним чином: Створимо об’єкт tk.Canvas, задамо йому параметри та розміситимемо за допомогою .place(). Всі об’єкти початкового меню будемо розміщувати на даному об’єкті canvas. Для стартового меню знадобиться чотири кнопки, створюються вони за допомогою віджета tk.Button, розміщуються через .place(), а також використовуючи метод .bind(), прив’язуємо до кнопки певну дію, яка буде виконуватися після натискання. Взагалі щоб викликати якусь дію чи функцію достатньо і параметра command при створення, але у даному випадку буде викликатися не тільки викликатися функція, а й передаватися в неї параметр. Створимо таку функцію, назвемо її next(), вона виконуватиме наступне: спершу застосує метод .destroy() до об’єкта canvas, щоб очистити екран, та викличе іншу функцію відповідно до того, який параметр передали (яку кнопку натиснули). Наприклад, при натисканні кнопки «Переглянути/редагувати базу даних» у next() буде передаватися 0 і викликатиметься view\_db(). Всі функції реалізуємо далі. Все вищеперераховане буде винесено в окрему функцію під назвою start\_window() і вона буде викликатися кожного разу при запуску програми та при поверненні на початковий екран. На рисунку 3 зображений зовнішній вигляд початкового екрану програми з кнопками. Путе місце в правій половині відведене для віджетів, які з’являться пізніше для однієї з функцій у next(). Також, перед переходом до наступних підпунктів, потрібно зробити макет для кнопки «Перейти на початковий екран». Вона буде створюватися на фреймі toolbar і буде повертати користувача на стартовий екран. Ця кнопка буде викликати функцію, назвемо її to\_start(). Вона буде знищювати всі непотрібні елементи методом .destroy(), а також викликати функцію start\_window().

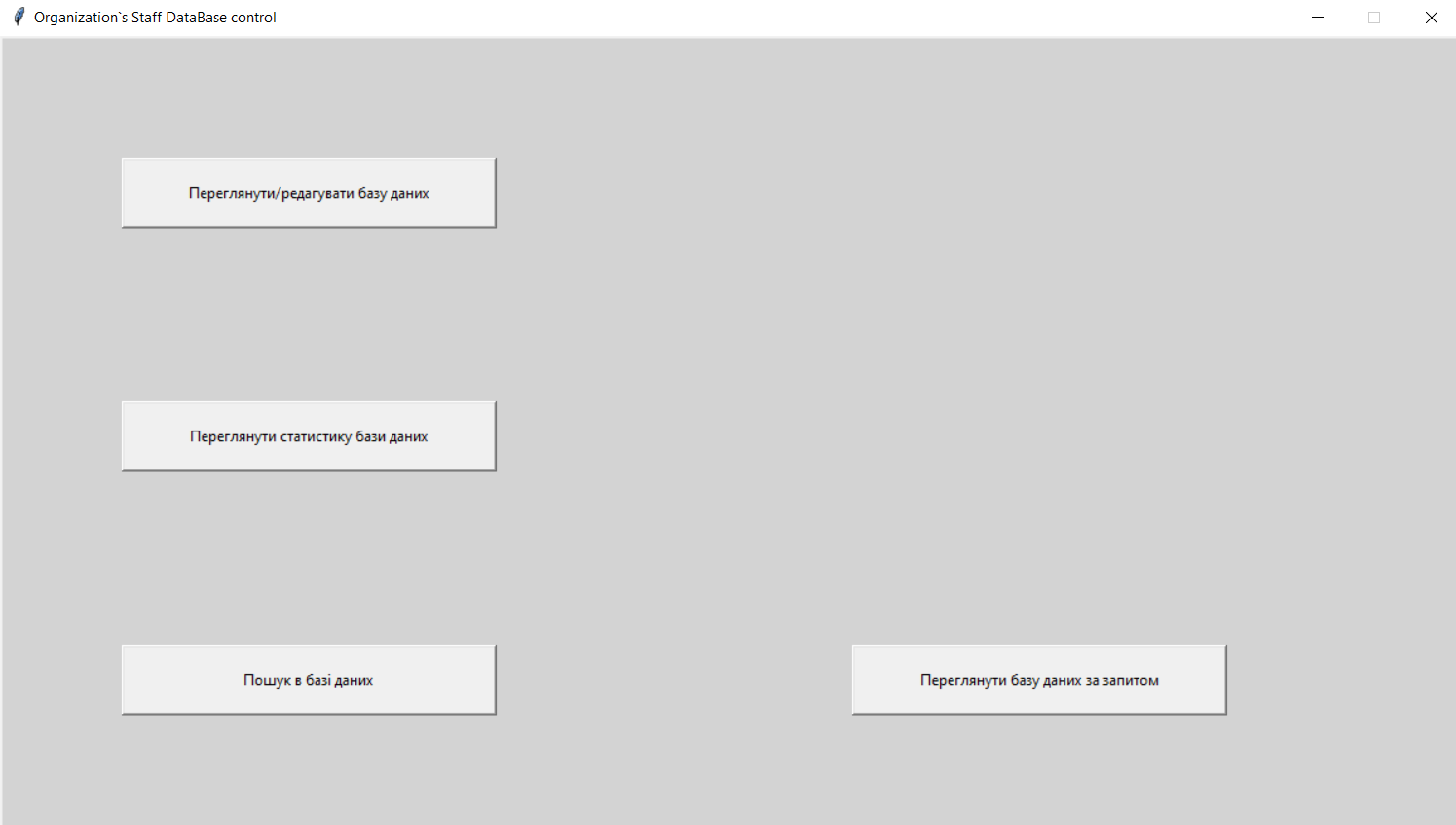


Рисунок 3 – Початковий екран з кнопками;

* 1. **Підключення бази даних до додатку**

Як було зазначено в першому розділі, БД SQLite дуже легко зв’язати з програмою на Python. Спершу потрібно імпортувати спеціальний модуль, що було зроблено в підпункті 3.1. Наступним обов’язковим кроком є підключення файлу з розширенням \*.db (єдиний файл БД). Це виконується через метод .connect() модуля sqlite3 і результат записується в змінну. Якщо файлу з такою назвою немає, то автоматично створиться новий, але він, звичайно, буде пустим, тому все одно це необхідно уважно перевіряти. Ще один об’єкт який потрібен для роботи з БД SQLite – це курсор. Для того щоб його створити потрібно застосувати метод курсор до підключеної БД. Курсор буде вказувати на БД і через нього будуть передаватися запити до неї. Як результат, за три рядки коду(імпортування модуля, підключення файлу БД через змінну і створення курсора) вже є підключена БД та об’єкт для взаємодії з нею. Дана частина коду розміщенна окремо від функцій, перед root.mainloop() (запуск інтерфейсу додатка), адже вона повинна бути виконана в будь-якому випадку для коректної роботи програми.

* 1. **Візуалізація записів бази даних в додатку**

В даному підпункті буде розглянута візуалізація записів двома способами: звичайний перегляд БД та розширений. В першому випадку користувач зможе переглядати всі дані у вигляді кількох таблиць, а в другому – самостійно обирати, яка інформація буде виведена. Почнемо зі «звичайного» перегляду БД. Створимо нову функцію з назвою view\_db(), що буде викликатися при натисканні кнопки «Переглянути/редагувати базу даних» у стартовому меню. Візуалізовуватися дані будуть за допомого об’єкта ttk.Treeview(). Створимо його на фреймі main та назвемо tree (звертатися до нього потрібно як main.tree). У БД під яку розробляється даний додаток є три таблиці: «Співробітники», «Архів зарплат» та «Посади». Створимо для кожної з них функцію (staff\_view(), salary\_archive\_view(), positions\_view() відповідно). Для всіх таблиці візуалізація буде абсолютно аналогічною, тому розглянемо в більш загальному випадку, не прив’язуючись до конкретних даних. В першу чергу застосовується команда main.tree.destroy() для очищення фрейма. Далі знову створюється main.tree як об’єкт ttk.Treeview(), але вже з потрібними стовпцями. Далі через методи .column() та .heading() для кожного стовпця окремо задаються параметри розміру, орієнтації тексту тощо. Тепер, коли каркас таблиці готовий, застосовуємо запит SELECT до відповідної таблиці через метод .execute() об’єкта cursor. Далі записуємо всі отримані дані з таблиці в змінну, використовуючи метод .fetchall(). Останнім кроком створюємо цикл, в якому методом .insert для main.tree записуємо дані в таблицю (рисунок 4).

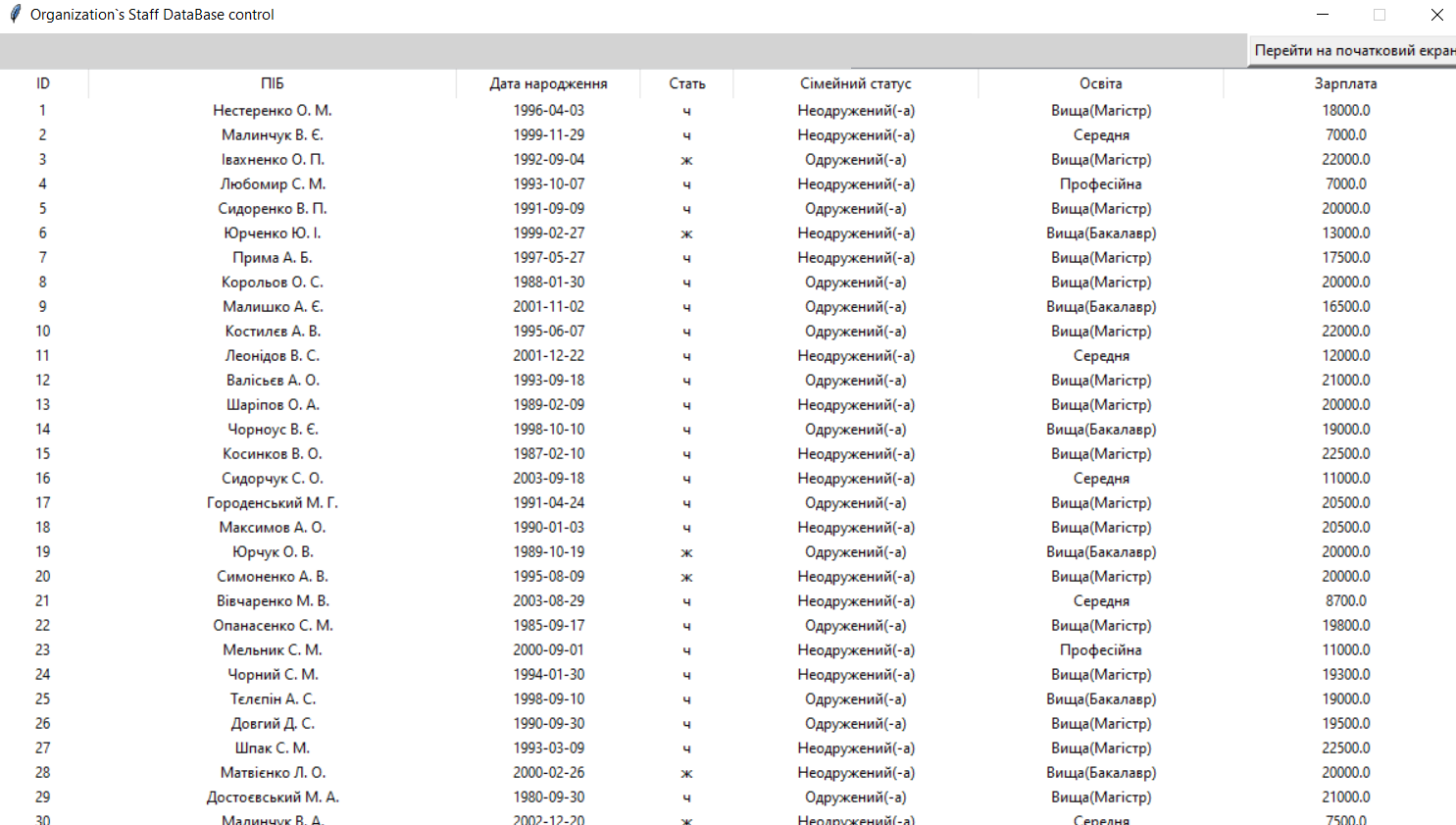


Рисунок 4 – Візуалізація даних в додатку на прикладі таблиці «Співробітники»;

Для коректної роботи потрібно щоб ширина стовпців не змінювалася, адже у ході використання користувач може випадково зробити так, що «втратить» з поля зору частину даних. Для цього зробимо спеціальну функцію і напишемо її в інші (в тому числі й для «розширеного» перегляду БД). Це зробити неважко, просто коли користувач натискає мишкою на границю між стовпчиками, виконується оператор break.

Тепер повернемося до функції view\_db(). В БД є три таблиці, і кожну з них користувач має мати можливість переглянути. Для цього реалізуємо кнопки для переходу до тих чи інших записів. Створимо об’єкт tk.Canvas, назвемо його canvas\_view\_db. На ньому і розташуємо кнопки, а після роботи з ним – видалимо при повернені на стартове меню через функцію to\_start() яку було розроблено у підпункті 3.1. Отже, на створеному об’єкті canvas розташуємо три кнопки: «Співробітники», «Архів зарплат» та «Посади». При натисканні кожної з них, виконуватиметься одна із функцій staff\_view(), salary\_archive\_view(), positions\_view() відповідно до того, що натиснув користувач(рисунок 5).

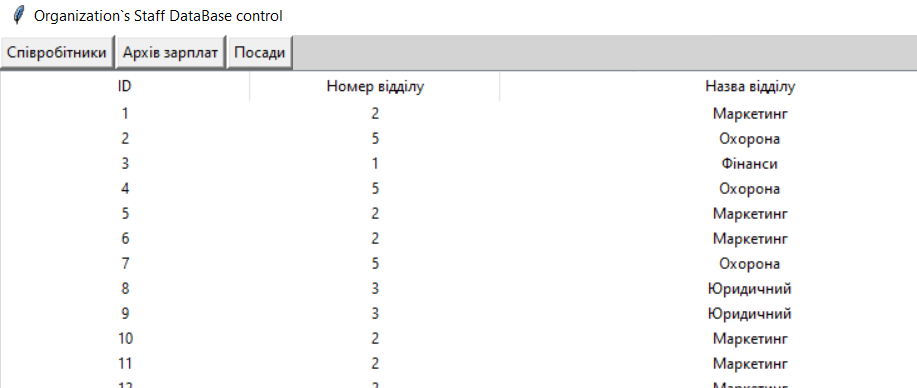


Рисунок 5 – Кнопки для переключення між таблицями;

«Звичайний» перегляд БД готовий, тепер перейдемо до «розширеного». Для цього потрібно повернутися трохи назад, до початкового вікна. За виклик функції даної візуалізації буде відповідати кнопка «Переглянути базу даних за запитом». Користувач буде сам обирати які атрибути будуть виведені на екран за допомогою «прапорців». Для цього використаємо віджет tk.Checkbox(). Створимо дев’ять таких об’єктів (для ID, ПІБ, дати народження, статі, сімейного статусу, освіти, зарплати, відділу, посаді; зображено на рисунку 6), а також, відповідно, дев’ять змінних, які будуть пов’язані з ними. Цим змінним надамо спеціальний тип даних – tk.BooleanVar(). Тепер якщо користувач відмітить прапорець змінна візьме значення true(1), в іншому випадку – false(0).

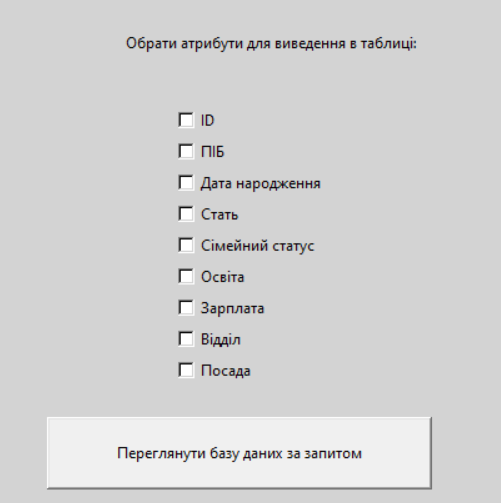


Рисунок 6 – прапорці для розширеного перегляду;

Створимо функцію для «розширеного» виводу, назвемо її expanded\_view\_db(). Сюди передамо дев’ять змінних типу tk.VarBoolean(), відповідно до того які прапорці обрав користувач в попередньому вікні. Далі створюємо два списки – result та head. Після цього для кожної змінної, що має значення true, зчитуємо з БД відповідний атрибут і записуємо його в список result, а в head – назву колонки, яка відображатиметься в таблиці ttk.Treeview(). Далі аналогічно до того як створювалися подібні таблиці на початку даного підпункту, єдине що формування колонок та надання їм певних параметрів проходить в циклі, адже завчасно невідомо скільки пунктів може обрати користувач.

Для коректної роботи програми потрібно зробити ще одну річ: обмежити, щоб користувач увімкнув хоча б один прапорець. Для цього у start\_window(), а саме у функції next() поставимо обмеження, що хоча б одна зі змінних мала значення true, в іншому випадку програма залишиться на початковому вікні.

* 1. **Редагування записів бази даних**

Крім перегляду записів БД, користувач повинен мати змогу і редагувати їх. Для цього в тій же функції view\_db() реалізуємо три кнопки: «Додати запис», «Змінити запис», «Видалити запис» та «Ввести посаду». Почнемо з додавання записів. Для цього реалізуємо функцію add\_record(), яка буде викликатися при натисканні на відповідну кнопку. Вводити інформацію в новий запис користувач буде через дочірнє вікно, що створюється за допомогою об’єкта tk.Toplevel(). Параметри до даного об’єкта задаються так само, як і для головного вікна. Це описано в підпункті 3.1. Далі для кожного поля в БД створюємо по два об’єкти: текстове поле tk.Label та віджет для вводу даних користувачем ttk.Entry. За допомогою тексту користувач буде бачити яке поле відповідає за який атрибут БД (рисунок 7). Також на новому вікні створюємо дві кнопки: «Підтвердити» та «Скасувати». Обидві з них будуть закривати дочірнє вікно для введення даних, але перша, крім того, викликатиме функцію insert\_data(). В неї передаватимуться всі дані, що ввів користувач для запису в БД. Також буде визначатися найменший вільний id під який і буде записана введена користувачем інформація, а всі пусті комірки для даних архіву зарплат будуть змінені на значення None. Далі, за допомогою запиту «INSERT INTO», в БД буде створено новий запис. Інформація в таблицю «Посади» буде введена окремо, щоб не нагромаджувати функції.

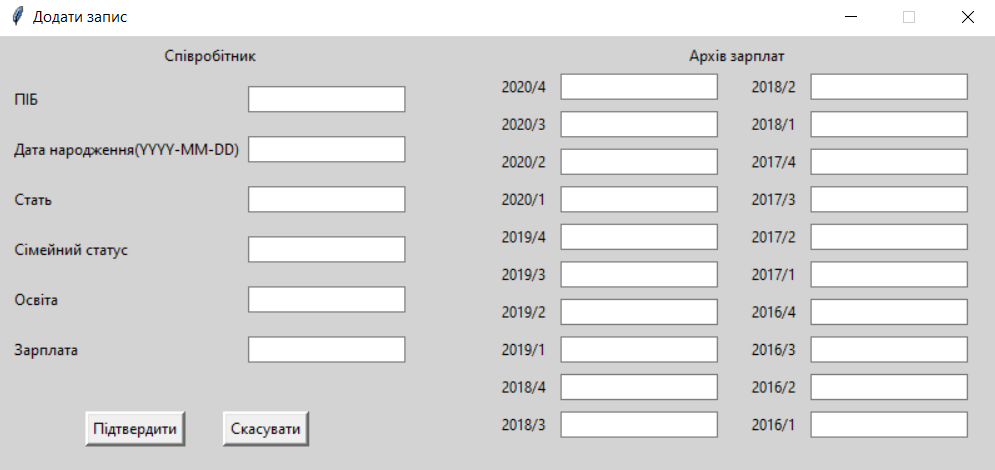


Рисунок 7 – Дочірнє вікно для додавання записів в БД, аналогічне буде і для оновлення, зміниться лише назва;

Тепер реалізуємо зміну даних в записі БД. Це буде зробити досить просто, адже вже створена функція створення запису. Оновлення буде відбуватися майже так само. Дочірнє вікно для введення користувачем інформації повністю підходить як для створення нових записів, так і для оновлення вже створених. Єдині зміни будуть у функції insert\_data(). По-перше, можна видалити частину коду де визначається найменший вільний id, адже ми оновлюємо вже створений запис зі своїм власним. По-друге, запит «INSERT INTO» та порядок параметрів, що передаються зміниться. З’явиться структура «UPDATE … SET … WHERE». Який запис змінювати, буде визначатися методом .selection() для main.tree.

Наступний крок – видалення записів БД. Для цього реалізуємо функцію delete\_record(), що викликатиметься кнопкою «Видвлити запис». Тут буде лише один цикл та виклик staff\_view(), для того щоб одразу була виведена оновлена БД. В середині даного циклу будуть запити «DELETE», що виконуватимуться по значенню id. Ці значення будуть отримуватися з методу .selection() об’єкта main.tree.

Останній крок – функція для кнопки «Ввести посаду». Реалізуємо для цього функцію set\_position(). Вона є аналогічною до update\_record(). Відмінність лише в тому до яких атрибутів БД вона застосовується. Функція для створення непотрібна, адже це реалізовано у add\_record(). Також підходить і каркас дочірнього вікна що використовувався раніше, але потрібно його зменшити у розмірах, адже залишається багато вільного місця (рисунок 8).

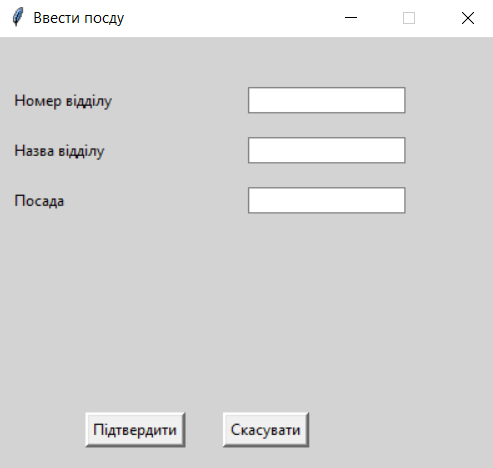


Рисунок 8 – Дочірнє вікно для введення інформації про посади;

* 1. **Пошук записів у бази даних**

Для швидшої роботи з записами в БД потрібно реалізувати пошук. Для цього створимо функцію search\_db(), що буде викликатися через кнопку «Пошук в базі даних». Спершу командою main.tree.destroy() очищаємо фрейм. Далі будуємо пусту таблицю зі значеннями які будуть доступні для пошуку (візьмемо таблицю staff). Також створимо кнопку «Пошук», яка буде викликати функцію search\_records(). Наступним кроком розташуємо текстові поля (tk.Label) та віджети для введення даних користувачем (ttk.Entry) для полів ID, ПІБ, дата народження, зарплата. Крім них, для деяких полів знадобиться додатковий віджет ttk.Combobox. Реалізуємо так, щоб для «дати народження» можна було обрати що саме шукати: день, місяць, рік чи одразу всю дату, а для зарплати – більше або менше заданого значення, точну суму що ввів користувач, або в межах (для цього використовується додатковий віджет ttk.Entry). Для полів стать, сімейний статус та освіта створимо віджети ttk.Combobox, адже в даних атрибутах досить обмежена кількість можливих значень. Далі передаємо всі дані що ввів користувач у віджети ttk.Entry до функції search\_records(). В даній функції обробляються всі дані. Наприклад, до ID та ПІБ на початок та кінець додається символ «%» для того щоб користувач міг ввести запит частково, а не точно увесь. Для інших даних формуються запити до БД відповідно до того, що обрав користувач зі списків Combobox. Далі, використовуючи структуру «SELECT … WHERE … LIKE» разом з командами «and» в таблицю виводяться лише ті записи, що відповідають запиту користувача (рисунок 9).

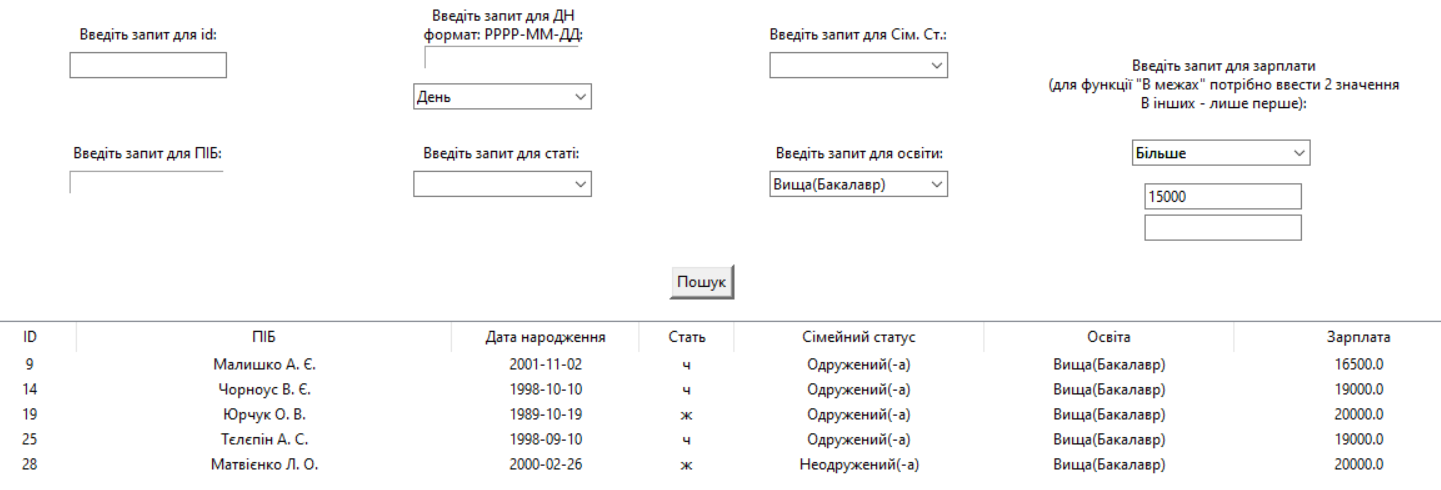


Рисунок 9 – Вікно пошуку в БД;

* 1. **Аналіз даних у базі та його візуалізація**

Залишається остання частина додатку – статистичний аналіз даних БД. Для цього реалізуємо функція statistic(), що буде викликатися через кнопку «Переглянути статистику бази даних». Візуалізацію статистичних показників зробимо за допомогою двох типів діаграм – стовпчикових та кругових. Перший тип реалізуємо наступним чином: по осі Х будуть розташовуватися квартали, а по Y – значення загальної та середньої зарплат, а також кількості співробітників за заданий період. Для цього створимо по три текстових поля та кнопки, що будуть будувати відповідні графіки. Далі – кругові діаграми. Для них створимо по шість текстових полів та кнопок відповідно для: середня, загальна зарплата та кількісний склад по освіті та статі. Наступним кроком реалізуємо функцію show\_diagrams\_window(), в неї буде передаватися один параметр, що і буде визначати, яку діаграму потрібно будувати. Але спершу буде відкриватися дочірнє вікно на основі tk.Toplevel (аналогічно до підпункту 3.4). Також створимо фрейм toolbar та розташуємо на ньому кнопку «Вийти». Далі створюємо об’єкт canvas, на ньому будуть будуватися діаграми. Для роботи даної функції в statistic() перед її викликом потрібно побудувати дерево зі структури if / elif. В залежності від того яку кнопку натисне користувач, в statistic(), а потім і в show\_diagram\_window() будуть передаватися відповідні значення (по аналогії зі стартовим меню в підпункті 3.1). Реалізовувати самі діаграми будемо двома способами: стовпчикові вручну, а кругові за допомогою спеціальної бібліотеки.

Почнемо зі стовпчикових діаграм. Спершу потрібно побудувати на об’єкті canvas координатну сітку. Для цього розташуємо дві лінії як осі X та Y (через метод .create\_line() об’єкта canvas). Вісь X буде відповідати за період та буде поділена на 20 кварталів. Створимо для кожного текстове поле і, за допомогою циклів, підпишемо кроки. З віссю Y трохи складніше, адже на даний момент навіть приблизно невідомо які будуть значення, тому потрібно зробити код універсальним. Для цього створимо функцію y\_minmax(), яка і буде визначати мінімальне та максимальне значення по осі Y. Далі визначаємо граничні показники на осі: максимальне значення округлюємо до найближчого більшого, а мінімального – до меншого. Виконавши дані дії, отримуємо верхню та нижню границю осі Y, які є трохи більшою та трохи меншою відповідно за максимальне та мінімальне значення. Далі визначаємо крок, а саме скільки пікселів відповідають одиниці по даній осі. Це потрібно для того, щоб побудувати стовпець діаграми. Останнім кроком будуються прямокутники, нижня частина яких стоїть на осі X, а верхня відповідає значенню Y в даній точці (приклад зображено на рисунку 10).

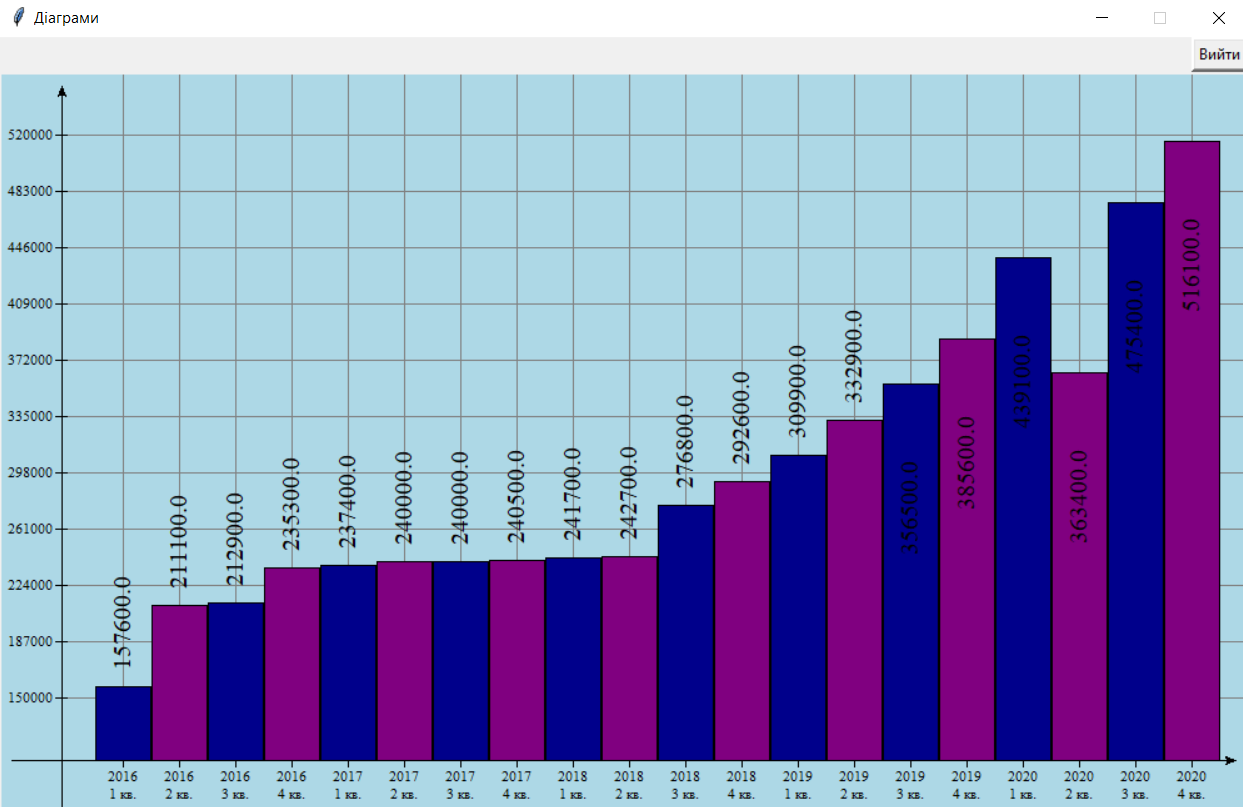


Рисунок 10 – Приклад стовпчикової діаграми;

Тепер реалізуємо кругову діаграму. Як було зазначено раніше, для даного типу діаграм застосуємо спеціальну бібліотеку, а саме matplotlib. Перед початком роботи її потрібно завантажити, адже вона не входить до стандартної бібліотеки Python. У Visual Studio це можна зробити через спеціальне середовище Python, яке створюється командою «Додати середовище…» на панелі інструментів. Після цього потрібно відкрити пакети Python та в пошуку ввести назву бажаної бібліотека (у нашому випадку matplotlib). Далі буде запропоновано виконати команду «pip install» для завантаження бібліотеки, після чого вже можна імпортувати бібліотеку та користуватися нею. Всі кругові діаграми за допомогою даної бібліотеки будуватимуться аналогічно, тому розглянемо в загальному. Спершу потрібно провести статистичні підрахунки, на основі яких будуть реалізовуватися діаграми. Наприклад, порахувати середню чи загальну зарплату для співробітників з різною освітою. Далі у дві змінні записати назви частин кругової діаграми та відповідні дані до них, які щойно були пораховані. Далі, використовуючи метод .subplots(), створити саму діаграму, і записати в неї значення використовуючи щойно створені змінні. Наступним кроком ми зберігаємо діаграму методом .savefig() та завершуємо процес за допомогою .close(). Тепер розташовуємо збережене зображення діаграми на об’єкті canvas. Щоб інформація краще сприймалася, потрібно щоб вона була не тільки на самій діаграмі, а й поруч текстом. Створюємо кілька текстових полів на яких буде записана відповідна інформація. Для кращої інформативності додаймо уточнення, наприклад діаграми зі зарплатами можуть доповнити дані про кількість співробітників, що потрапляють до тієї чи іншої категорії (приклад зображено на рисунок 11).

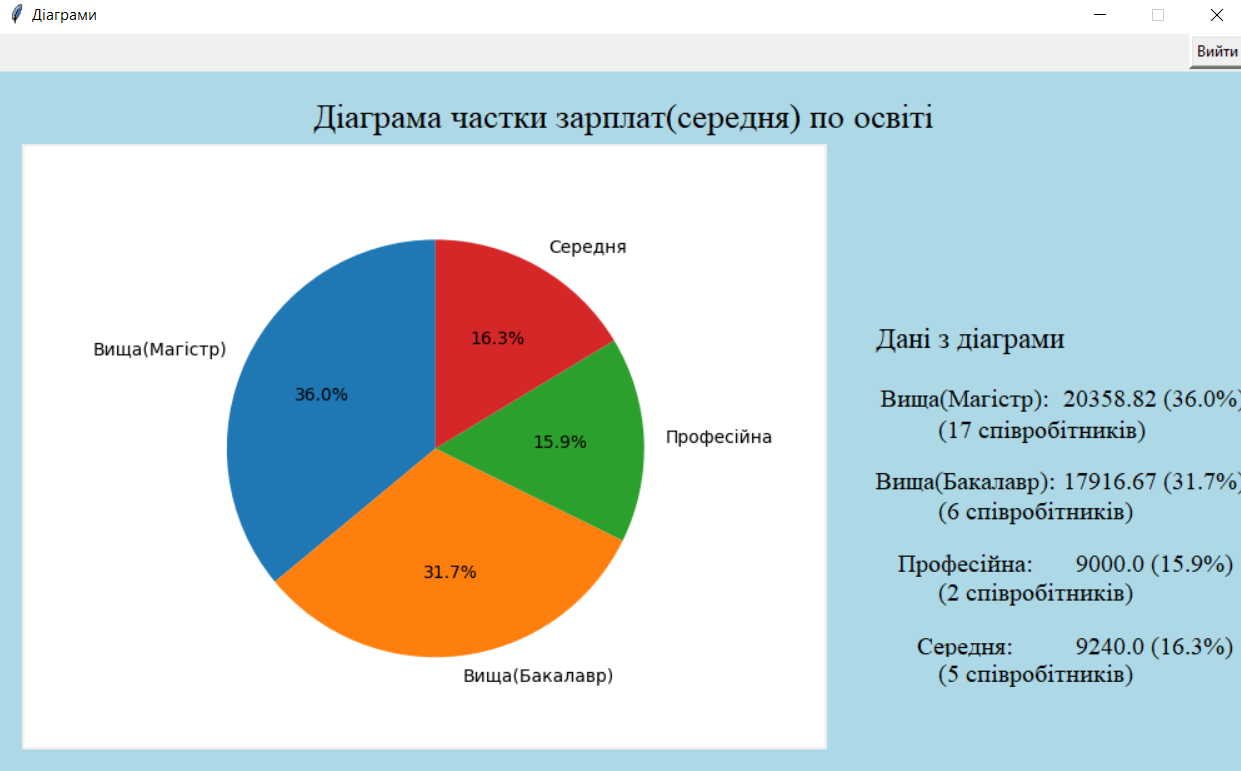


Рисунок 11 – Приклад кругової діаграми;

* 1. **Тестування додатку**

Для тестування додатку було створена база даних (додатки А, Б, В, Г), в неї увійшли 30 співробітників, що працюють в організації на даний момент та 20 колишніх співробітників. Таким чином таблиці «Співробітники» та «Посади» мають по 30 записів, а в «Архіві зарплат» відповідно 50. На рисунках 12-16 будуть показані діаграми для даної вибірки. На основі отриманих діаграм візуально можна зробити певні висновки не заглиблюючись в цифри. Наприклад, з діаграми на рисунку 14 видно, що кількість співробітників за даний період сильно збільшилась, це пояснює велику різницю між загальною зарплатою на початку і в кінці (рисунок 12). Також на рисунку 13 (середня зарплата), видно великий «провал» у другому кварталі 2020-го року, можна зробити висновок що організація сильно постраждала від пандемії якщо довелося настільки сильно скорочувати заробітну плату. На рисунку 15 видно, що більшість зарплатної відомості займають магістри, але в той же час, їх найбільше. Рисунок 16 показує що середня зарплата у чоловіків та жінок у даній організації приблизно рівна, але кількість співробітників чоловічої статі значно більша. Таким чином користувач може аналізувати дані з БД на основі діаграм.

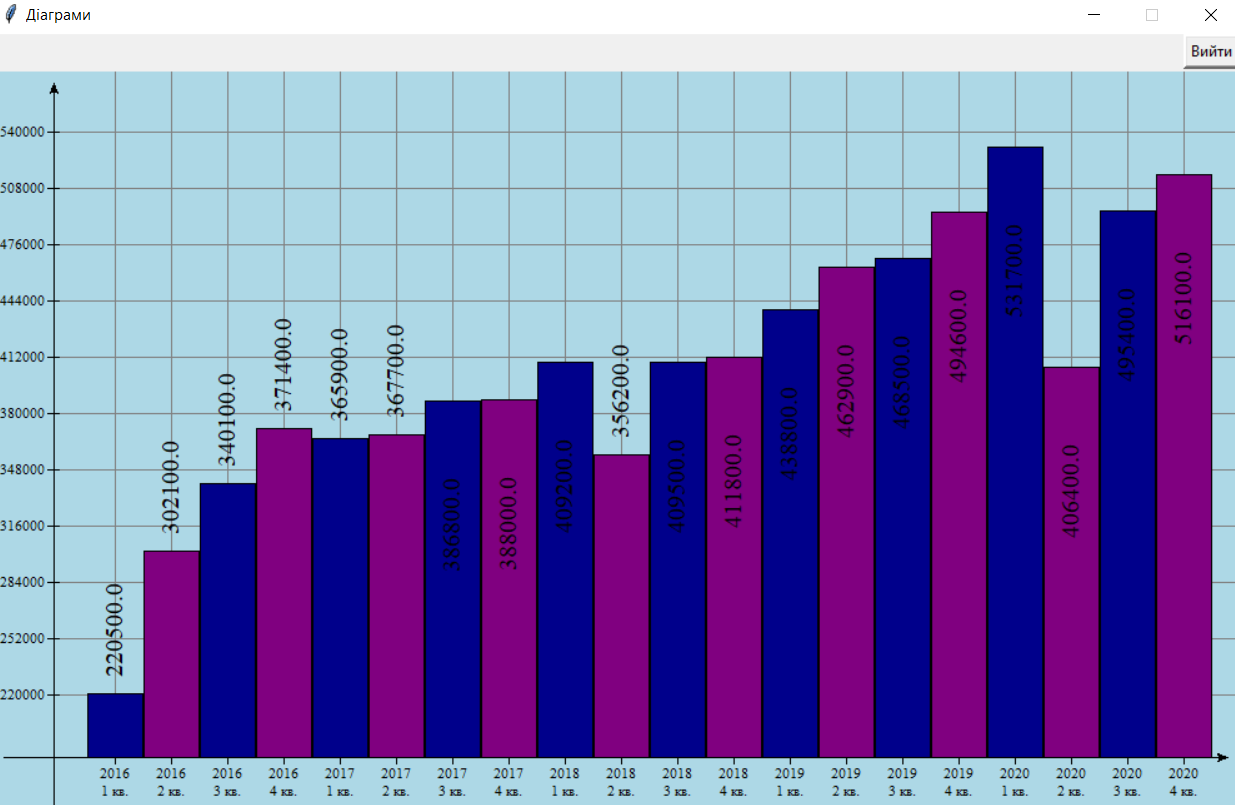


Рисунок 12 – Загальна зарплата в організації за період;

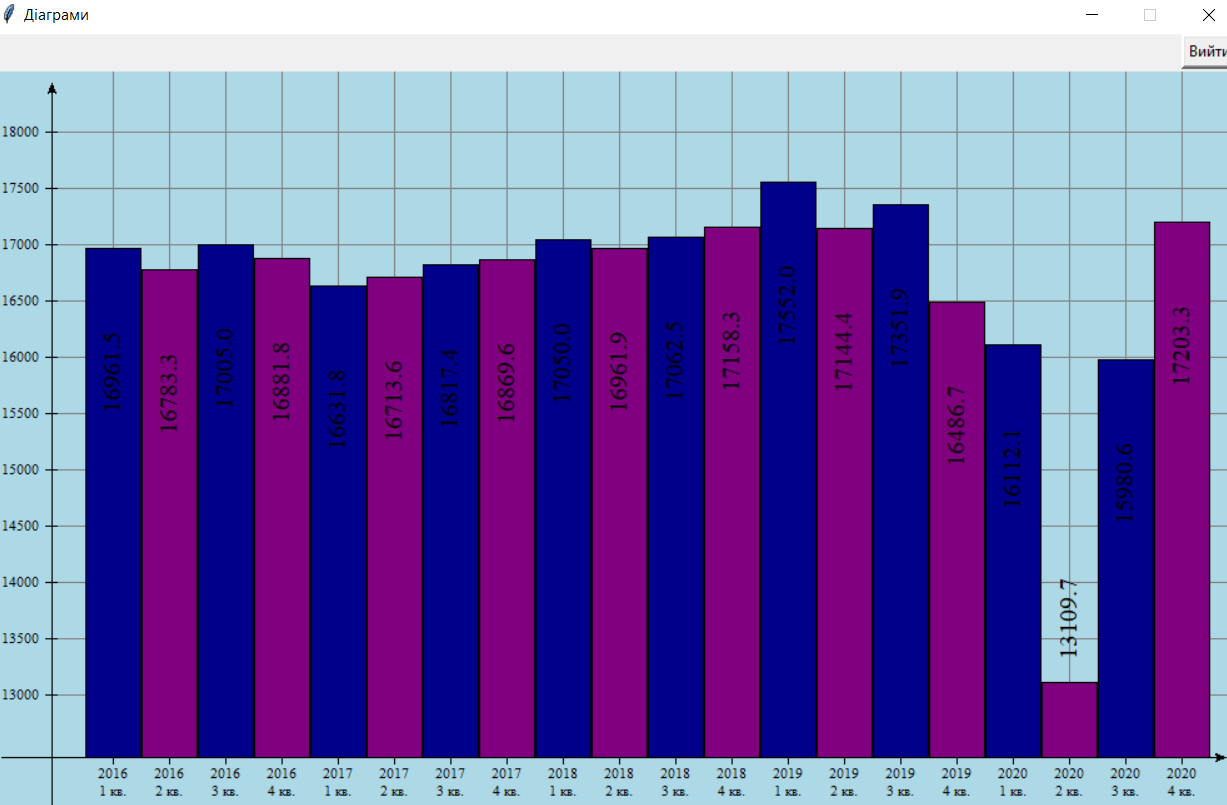


Рисунок 13 – Середня зарплата в організації за період;

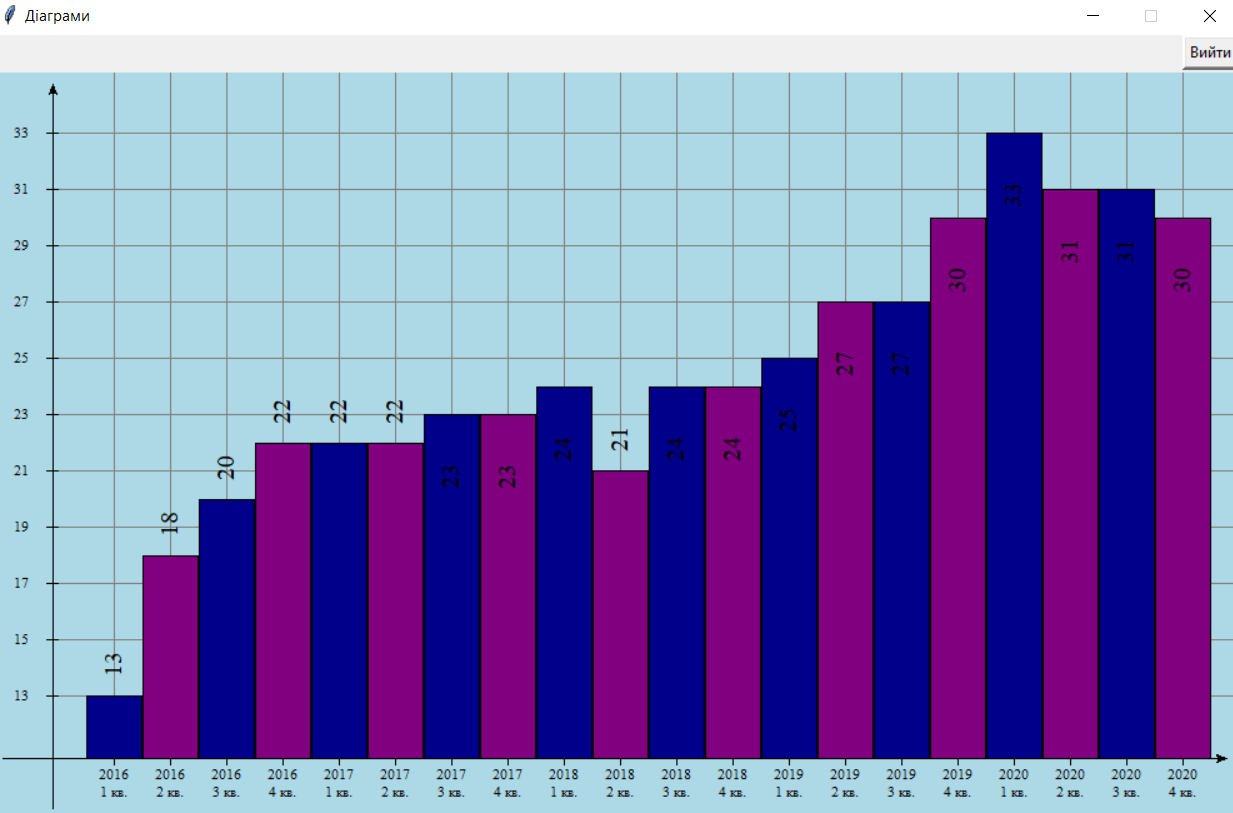


Рисунок 14 – Кількість співробітників за період;



Рисунок 15 – Діаграма частки загальної зарплати по освіті;



Рисунок 16 – Діаграма частки середньої зарплати по статі;

**Висновок до розділу 2:** У ході виконання практичної частини даної курсової роботи був розроблений додаток з графічним інтерфейсом для роботи з базою даних співробітників організації. За його допомогою можна переглядати записи БД у формі таблиці, редагувати їх (додавати нові, видаляти змінювати певні поля), виконувати пошук по атрибутах окремо чи в комбінації, а також формувати власну таблицю для перегляду даних лише в потрібних атрибутах. Також додаток може проводити певні статистичні підрахунки та візуалізовувати їх будуючи стовпчикові та кругові діаграми.

1. **Висновки**

Як було описано у першому розділі, існує чимало інструментів для створення програм з графічним інтерфейсом для Python, такі як: Kivy, PyQT, PySide, Tkinter тощо. Саме останній буде використаний у ході реалізації завдання даної роботи. Бібліотека tkinter є досить широкою і має достатньо багато інструментів для створення інтерфейсу програми. Для того щоб зробити додаток більш високоорганізованим та зрозумілим можна використати спеціальне розширення tkinter ttk, яке заточене саме під покращення зовнішнього вигляду об’єктів, що робить їх більш природними. Наступне що було розглянуто – це SQLite. Дана СУБД є спрощеною в порівняння з іншими, наприклад MS SQL Server, але попри це має досить високий функціонал, що дозволяє реалізовувати багато завдань витрачаючи при цьому невеликі ресурси, як технічні, так і людські. Крім того, з SQLite комфортно працювати в парі з Python через модуль sqlite3, тому дана зв’язка є досить популярною. У другому розділі було показано виконання практичної частини даної курсової роботи, а саме був розроблений додаток з графічним інтерфейсом для роботи з базою даних співробітників організації. За його допомогою можна переглядати записи БД у формі таблиці, редагувати їх (додавати нові, видаляти змінювати певні поля), виконувати пошук по атрибутах окремо чи в комбінації, а також формувати власну таблицю для перегляду даних лише в потрібних атрибутах. Також програма може проводити певні статистичні підрахунки та візуалізовувати їх будуючи стовпчикові та кругові діаграми. Такий додаток може бути використаний спеціалістами відділу кадрів для автоматизації частини їхньої роботи. Звичайно, на даний момент, програма не може задовільнити всі можливі потреби, адже, як було зазначено у вступі, кожна компанія може мати свої вимоги до можливостей та функціоналу подібного додатку, різну організацію бази даних, унікальні побажання щодо аналізу інформації тощо. Тому потрібно консультуватися зі спеціалістами що будуть використовувати такий додаток та продовжувати працювати над ний й доповнювати його функціонал.

1. **Список використаних джерел**
2. Стаття «Найкращі GUI бібліотеки Python»

<https://pythonru.com/baza-znanij/luchshie-gui-biblioteki-python-2020>

1. Стаття «Графічний інтерфейс користувача»

<https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0>

1. Стаття «PySide»

<https://uk.wikipedia.org/wiki/PySide>

1. Стаття «Бібліотека tkinter мови Python»

<http://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/tkinter/index.html#main>

1. Jay A. Kreibich. «Using SQLite: Small. Fast. Reliable. Choose Any Three.», Серпень 2010, 506 с.

**Додаток А**

Тестова база даних, таблиця «Співробітники»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | ПІБ | Дата народження | Стать | Сімейний статус | Освіта | Зарплата |
| 1 | Нестеренко О. М. | 1996-04-03 | ч | Неодружений(-а) | Вища(Магістр) | 18000.0 |
| 2 | Малинчук В. Є. | 1999-11-29 | ч | Неодружений(-а) | Середня | 7000.0 |
| 3 | Івахненко О. П. | 1992-09-04 | ж | Одружений(-а) | Вища(Магістр) | 22000.0 |
| 4 | Любомир С. М. | 1993-10-07 | ч | Неодружений(-а) | Професійна | 7000.0 |
| 5 | Сидоренко В. П. | 1991-09-09 | ч | Одружений(-а) | Вища(Магістр) | 20000.0 |
| 6 | Юрченко Ю. І. | 1999-02-27 | ж | Неодружений(-а) | Вища(Бакалавр) | 13000.0 |
| 7 | Прима А. Б. | 1997-05-27 | ч | Неодружений(-а) | Вища(Магістр) | 17500.0 |
| 8 | Корольов О. С. | 1988-01-30 | ч | Одружений(-а) | Вища(Магістр) | 20000.0 |
| 9 | Малишко А. Є. | 2001-11-02 | ч | Одружений(-а) | Вища(Бакалавр) | 16500.0 |
| 10 | Костилєв А. В. | 1995-06-07 | ч | Одружений(-а) | Вища(Магістр) | 22000.0 |
| 11 | Леонідов В. С. | 2001-12-22 | ч | Неодружений(-а) | Середня | 12000.0 |
| 12 | Валісьєв А. О. | 1993-09-18 | ч | Одружений(-а) | Вища(Магістр) | 21000.0 |
| 13 | Шаріпов О. А. | 1989-02-09 | ч | Неодружений(-а) | Вища(Магістр) | 20000.0 |
| 14 | Чорноус В. Є. | 1998-10-10 | ч | Одружений(-а) | Вища(Бакалавр) | 19000.0 |
| 15 | Косинков В. О. | 1987-02-10 | ч | Неодружений(-а) | Вища(Магістр) | 22500.0 |
| 16 | Сидорчук С. О. | 2003-09-18 | ч | Неодружений(-а) | Середня | 11000.0 |
| 17 | Городенський М. Г. | 1991-04-24 | ч | Одружений(-а) | Вища(Магістр) | 20500.0 |
| 18 | Максимов А. О. | 1990-01-03 | ч | Неодружений(-а) | Вища(Магістр) | 20500.0 |
| 19 | Юрчук О. В. | 1989-10-19 | ж | Одружений(-а) | Вища(Бакалавр) | 20000.0 |
| 20 | Симоненко А. В. | 1995-08-09 | ж | Неодружений(-а) | Вища(Магістр) | 20000.0 |
| 21 | Вівчаренко М. В. | 2003-08-29 | ч | Неодружений(-а) | Середня | 8700.0 |
| 22 | Опанасенко С. М. | 1985-09-17 | ч | Одружений(-а) | Вища(Магістр) | 19800.0 |
| 23 | Мельник С. М. | 2000-09-01 | ч | Неодружений(-а) | Професійна | 11000.0 |
| 24 | Чорний С. М. | 1994-01-30 | ч | Неодружений(-а) | Вища(Магістр) | 19300.0 |
| 25 | Тєлєпін А. С. | 1998-09-10 | ч | Одружений(-а) | Вища(Бакалавр) | 19000.0 |
| 26 | Довгий Д. С. | 1990-09-30 | ч | Одружений(-а) | Вища(Магістр) | 19500.0 |
| 27 | Шпак С. М. | 1993-03-09 | ч | Неодружений(-а) | Вища(Магістр) | 22500.0 |
| 28 | Матвієнко Л. О. | 2000-02-26 | ж | Неодружений(-а) | Вища(Бакалавр) | 20000.0 |
| 29 | Достоєвський М. А. | 1980-09-30 | ч | Одружений(-а) | Вища(Магістр) | 21000.0 |
| 30 | Малинчук В. А. | 2002-12-20 | ж | Неодружений(-а) | Середня | 7500.0 |

**Додаток Б**

Тестова база даних, таблиця «Посади»

|  |  |  |  |
| --- | --- | --- | --- |
| ID | № відділу | Назва відділу | Посада |
| 1 | 2 | Маркетинг | Маркетолог |
| 2 | 5 | Охорона | Охоронець |
| 3 | 1 | Фінанси | Голова відділу |
| 4 | 5 | Охорона | Охоронець |
| 5 | 2 | Маркетинг | Замісник голови відділу |
| 6 | 2 | Маркетинг | Бухгалтер |
| 7 | 5 | Охорона | Замісник голови відділу |
| 8 | 3 | Юридичний | Юрист |
| 9 | 3 | Юридичний | Юрист |
| 10 | 2 | Маркетинг | Маркетолог |
| 11 | 2 | Маркетинг | Бухгалтер |
| 12 | 2 | Маркетинг | Голова відділу |
| 13 | 5 | Охорона | Голова відділу |
| 14 | 2 | Маркетинг | Маркетолог |
| 15 | 3 | Юридичний | Голова відділу |
| 16 | 4 | Логістичний | Бухгалтер |
| 17 | 4 | Логістичний | Замісник голови відділу |
| 18 | 2 | Маркетинг | Маркетолог |
| 19 | 2 | Маркетинг | Маркетолог |
| 20 | 2 | Маркетинг | Маркетолог |
| 21 | 5 | Охорона | Охоронець |
| 22 | 2 | Маркетинг | Маркетолог |
| 23 | 5 | Охорона | Бухгалтер |
| 24 | 2 | Маркетинг | Маркетогол |
| 25 | 2 | Маркетинг | Маркетолог |
| 26 | 4 | Логістичний | Логіст |
| 27 | 4 | Логістичний | Голова відділу |
| 28 | 4 | Логістичний | Логіст |
| 29 | 3 | Юридичний | Юрист |
| 30 | 5 | Охорона | Охоронець |

**Додаток В**

Тестова база даних, таблиця «Архів зарплат» частина 1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | 2016/1 | 2016/2 | 2016/3 | 2016/4 | 2017/1 | 2017/2 | 2017/3 | 2017/4 | 2018/1 | 2018/2 |
| 1 |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 |  |
| 3 | 17000.0 | 17000.0 | 17200.0 | 18000.0 |  |  |  |  |  | 19000.0 |
| 4 |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  | 18000.0 | 18000.0 | 18000.0 | 18500.0 |  |  |
| 7 |  |  |  |  | 18000.0 | 18200.0 | 18000.0 | 18000.0 | 18000.0 |  |
| 8 | 18300.0 | 18300.0 | 18500.0 | 19000.0 | 18200.0 | 18000.0 | 18000.0 | 18000.0 | 18500.0 | 19700.0 |
| 9 |  |  |  |  |  |  |  |  | 19000.0 |  |
| 10 |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |  |
| 12 |  | 17800.0 | 18000.0 | 18000.0 | 7500.0 | 7500.0 | 7800.0 | 8000.0 | 8000.0 | 18500.0 |
| 13 | 17000.0 | 17000.0 | 17200.0 | 18000.0 |  |  |  |  |  | 18500.0 |
| 14 |  |  |  |  |  |  |  |  |  |  |
| 15 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 12000.0 | 13000.0 | 13000.0 | 13000.0 | 13000.0 | 20000.0 |
| 16 |  |  |  |  |  | 18000.0 | 19000.0 | 19000.0 | 20000.0 |  |
| 17 |  | 17000.0 | 17000.0 | 17000.0 | 20000.0 |  |  |  |  | 18000.0 |
| 18 | 17000.0 | 17500.0 | 17500.0 | 18000.0 |  |  |  |  | 18000.0 | 18000.0 |
| 19 | 18000.0 | 18000.0 | 18000.0 | 18000.0 |  |  | 17000.0 | 17000.0 | 17000.0 | 18000.0 |
| 20 |  | 17500.0 | 17500.0 | 17500.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 |
| 21 |  |  |  |  | 7800.0 | 8000.0 | 8000.0 | 8000.0 | 8000.0 |  |
| 22 | 19000.0 | 19000.0 | 19000.0 | 19000.0 |  |  |  |  |  | 19000.0 |
| 23 |  |  |  |  | 9000.0 | 9000.0 | 10000.0 | 10000.0 | 10000.0 |  |
| 24 |  |  |  |  |  |  |  |  |  |  |
| 25 |  |  |  |  | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 |  |
| 26 | 14300.0 | 15000.0 | 16000.0 | 16800.0 |  |  |  |  |  | 18000.0 |
| 27 |  |  |  | 19000.0 |  |  |  |  |  | 20000.0 |
| 28 |  |  |  |  |  |  |  |  |  |  |
| 29 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18500.0 |  | 18000.0 |
| 30 |  |  |  |  | 18000.0 | 18200.0 | 18000.0 | 18000.0 | 18000.0 |  |
| 31 |  |  |  |  | 18200.0 | 18000.0 | 18000.0 | 18000.0 | 18500.0 |  |
| 32 |  |  |  |  |  |  |  |  | 19000.0 |  |
| 33 | 18000.0 | 18000.0 | 18000.0 | 18200.0 |  |  |  |  |  |  |
| 34 |  |  | 18000.0 | 18000.0 |  |  |  |  |  |  |
| 35 |  |  | 18000.0 | 18000.0 | 7500.0 | 7500.0 | 7800.0 | 8000.0 | 8000.0 |  |
| 36 |  |  |  |  |  |  |  |  |  | 19000.0 |
| 37 |  |  |  |  |  |  |  |  |  |  |
| 38 |  |  |  |  | 12000.0 | 13000.0 | 13000.0 | 13000.0 | 13000.0 | 18000.0 |
| 39 |  |  |  | 7000.0 |  | 18000.0 | 19000.0 | 19000.0 | 20000.0 | 8000.0 |
| 40 | 16900.0 | 18000.0 | 18200.0 | 18000.0 | 20000.0 |  |  |  |  |  |
| 41 |  |  |  |  |  |  |  |  | 18000.0 |  |
| 42 | 11000.0 | 11000.0 | 11000.0 | 11000.0 |  |  | 17000.0 | 17000.0 | 17000.0 | 13000.0 |
| 43 |  |  |  |  | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 |  |
| 44 | 17000.0 | 17000.0 | 17000.0 | 18900.0 | 7800.0 | 8000.0 | 8000.0 | 8000.0 | 8000.0 |  |
| 45 |  |  |  |  |  |  |  |  |  | 18000.0 |
| 46 |  |  |  |  | 9000.0 | 9000.0 | 10000.0 | 10000.0 | 10000.0 | 18000.0 |
| 47 |  | 18000.0 | 18000.0 | 18000.0 |  |  |  |  |  |  |
| 48 |  |  |  |  | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 8000.0 |
| 49 |  |  |  |  |  |  |  |  |  |  |
| 50 |  | 9000.0 | 9000.0 | 9000.0 | 9000.0 | 9000.0 | 10000.0 | 10000.0 | 10000.0 | 11500.0 |

**Додаток Г**

Тестова база даних, таблиця «Архів зарплат» частина 2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | 2018/3 | 2018/4 | 2019/1 | 2019/2 | 2019/3 | 2019/4 | 2020/1 | 2020/2 | 2020/3 | 2020/4 |
| 1 |  |  |  |  |  |  | 16500.0 | 15000.0 | 15000.0 | 17000.0 |
| 2 |  |  |  |  |  |  | 7000.0 | 6000.0 | 7000.0 | 7000.0 |
| 3 | 19000.0 | 19000.0 | 19000.0 | 19200.0 | 19500.0 | 19500.0 | 19500.0 | 15000.0 | 20000.0 | 22000.0 |
| 4 |  |  |  |  |  | 6700.0 | 7000.0 | 6000.0 | 7000.0 | 7000.0 |
| 5 | 17800.0 | 17000.0 | 18000.0 | 18500.0 | 18500.0 | 18500.0 | 18500.0 | 15000.0 | 20000.0 | 20000.0 |
| 6 |  |  |  |  |  |  |  |  | 12100.0 | 13000.0 |
| 7 |  | 13000.0 | 13000.0 | 13000.0 | 16000.0 | 16000.0 | 16500.0 | 15000.0 | 15000.0 | 17500.0 |
| 8 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20300.0 | 16000.0 | 20000.0 | 20000.0 |
| 9 |  |  |  |  |  |  | 15000.0 | 11000.0 | 16000.0 | 16500.0 |
| 10 |  |  |  | 11000.0 | 18000.0 | 18000.0 | 18000.0 | 13800.0 | 20000.0 | 22000.0 |
| 11 |  |  |  | 10000.0 | 10000.0 | 10000.0 | 11000.0 | 7500.0 | 12000.0 | 12000.0 |
| 12 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19600.0 | 17000.0 | 20000.0 | 21000.0 |
| 13 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19600.0 | 17000.0 | 20000.0 | 20000.0 |
| 14 |  |  |  |  |  | 13500.0 | 14300.0 | 11000.0 | 17000.0 | 19000.0 |
| 15 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 15000.0 | 21000.0 | 22500.0 |
| 16 |  |  |  |  |  |  |  |  | 9800.0 | 11000.0 |
| 17 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 15000.0 | 20000.0 | 20500.0 |
| 18 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18200.0 | 18500.0 | 15000.0 | 20000.0 | 20500.0 |
| 19 | 18000.0 | 18000.0 | 18200.0 | 18000.0 | 18000.0 | 18200.0 | 18500.0 | 15000.0 | 20000.0 | 20000.0 |
| 20 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 16000.0 | 19500.0 | 20000.0 |
| 21 |  |  |  |  |  | 7000.0 | 7000.0 | 6500.0 | 8000.0 | 8700.0 |
| 22 | 19000.0 | 19500.0 | 19500.0 | 19500.0 | 19500.0 | 19500.0 | 19500.0 | 19500.0 | 19500.0 | 19800.0 |
| 23 |  |  |  |  |  |  | 7300.0 | 6500.0 | 10000.0 | 11000.0 |
| 24 | 15000.0 | 18100.0 | 19000.0 | 19500.0 | 19500.0 | 19500.0 | 19000.0 | 17300.0 | 19000.0 | 19300.0 |
| 25 |  |  | 15000.0 | 16200.0 | 16500.0 | 16500.0 | 17000.0 | 17300.0 | 12000.0 | 18300.0 |
| 26 | 18000.0 | 18000.0 | 18200.0 | 18000.0 | 18000.0 | 18000.0 | 18500.0 | 15000.0 | 16000.0 | 19500.0 |
| 27 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 20000.0 | 15000.0 | 21000.0 | 22500.0 |
| 28 |  |  |  |  | 13000.0 | 14500.0 | 17000.0 | 9000.0 | 13000.0 | 20000.0 |
| 29 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 10000.0 | 19500.0 | 21000.0 |
| 30 |  |  |  |  |  |  |  | 6000.0 | 6000.0 | 7500.0 |
| 31 |  |  | 16900.0 | 17000.0 | 17000.0 | 18000.0 |  |  |  |  |
| 32 |  |  |  |  |  | 7000.0 | 7000.0 |  |  |  |
| 33 |  |  |  |  |  |  |  |  |  |  |
| 34 |  |  |  |  |  |  |  |  |  |  |
| 35 |  |  |  |  |  |  |  |  |  |  |
| 36 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 |  |  |  |
| 37 |  | 17000.0 | 17000.0 | 18000.0 |  |  |  |  |  |  |
| 38 | 18500.0 | 18700.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 |  |  |
| 39 | 8000.0 | 8000.0 |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |
| 41 |  |  |  |  |  |  |  |  |  |  |
| 42 | 13000.0 |  |  |  |  |  |  |  |  |  |
| 43 |  |  |  |  |  |  |  |  |  |  |
| 44 |  |  |  |  |  |  |  |  |  |  |
| 45 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 | 18000.0 |  |  |  |
| 46 | 17700.0 |  |  |  |  |  |  |  |  |  |
| 47 |  |  |  |  |  |  |  |  |  |  |
| 48 | 8000.0 | 8000.0 | 9000.0 | 9000.0 | 9000.0 | 9000.0 | 10000.0 | 7000.0 |  |  |
| 49 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19000.0 | 19600.0 | 17000.0 | 20000.0 |  |
| 50 | 11500.0 | 11500.0 | 11000.0 | 11000.0 | 11000.0 |  |  |  |  |  |