**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и технологии |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Лабораторная работа 6**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **по дисциплине** |  | Программирование под Android | | |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | **Петухов Никита Максимович** |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | **группа ДБИ-301рки** |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев Иван Валерьевич |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2024 г.**

**Лабораторная работа: Работа с базами данных**

**Цель работы Научиться использовать базы данных для хранения и обработки информации с помощью Python, используя библиотеку SQLite. Мы создадим простое приложение для учета спортсменов и их времени в соревнованиях.**

**Описание приложения Приложение будет: 1. Позволять добавлять спортсменов и их результаты. 2. Хранить данные в базе данных SQLite. 3. Позволять извлекать и отображать данные из базы данных.**

**Шаг 1: Установка необходимого пакета Вам не нужно устанавливать никаких дополнительных библиотек для работы с SQLite, так как она встроена в стандартную библиотеку Python. Однако для работы с внешними базами данных (например, PostgreSQL или MySQL) потребуется установить дополнительные модули.**

**Шаг 2: Создание базы данных Создадим модуль для работы с базой данных, который будет включать функции для добавления и извлечения данных.**

**import sqlite3**

**class Database:**

**def \_\_init\_\_(self, db\_name):**

**self.connection = sqlite3.connect(db\_name)**

**self.cursor = self.connection.cursor()**

**self.create\_table()**

**def create\_table(self):**

**self.cursor.execute('''**

**CREATE TABLE IF NOT EXISTS athletes (**

**id INTEGER PRIMARY KEY AUTOINCREMENT,**

**name TEXT NOT NULL,**

**time REAL NOT NULL**

**)**

**''')**

**self.connection.commit()**

**def add\_athlete(self, name, time):**

**self.cursor.execute('''**

**INSERT INTO athletes (name, time) VALUES (?, ?)**

**''', (name, time))**

**self.connection.commit()**

**def get\_all\_athletes(self):**

**self.cursor.execute('SELECT \* FROM athletes')**

**return self.cursor.fetchall()**

**def close(self):**

**self.connection.close()**

**Шаг 3: Интерфейс приложения**

**Теперь создадим простой интерфейс на базе Tkinter для работы с приложением.**

**import tkinter as tk**

**from tkinter import messagebox**

**class AthleteApp:**

**def \_\_init\_\_(self, root):**

**self.db = Database('athletes.db')**

**self.root = root**

**self.root.title("Учет спортсменов")**

**self.name\_label = tk.Label(root, text="Имя спортсмена:")**

**self.name\_label.pack()**

**self.name\_entry = tk.Entry(root)**

**self.name\_entry.pack()**

**self.time\_label = tk.Label(root, text="Время (в секундах):")**

**self.time\_label.pack()**

**self.time\_entry = tk.Entry(root)**

**self.time\_entry.pack()**

**self.add\_button = tk.Button(root, text="Добавить спортсмена", command=self.add\_athlete)**

**self.add\_button.pack()**

**self.show\_button = tk.Button(root, text="Показать всех спортсменов", command=self.show\_athletes)**

**self.show\_button.pack()**

**self.result\_text = tk.Text(root, height=10, width=30)**

**self.result\_text.pack()**

**def add\_athlete(self):**

**name = self.name\_entry.get()**

**time = self.time\_entry.get()**

**if name and time:**

**try:**

**time = float(time) # Преобразуем время в float**

**self.db.add\_athlete(name, time)**

**messagebox.showinfo("Успех", "Спортсмен успешно добавлен!")**

**self.name\_entry.delete(0, tk.END)**

**self.time\_entry.delete(0, tk.END)**

**except ValueError:**

**messagebox.showerror("Ошибка", "Введите корректное время!")**

**def show\_athletes(self):**

**athletes = self.db.get\_all\_athletes()**

**self.result\_text.delete(1.0, tk.END) # Очищаем текстовое поле**

**for athlete in athletes:**

**self.result\_text.insert(tk.END, f"ID: {athlete[0]}, Имя: {athlete[1]}, Время: {athlete[2]} секунд\n")**

**def close(self):**

**self.db.close()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**root = tk.Tk()**

**app = AthleteApp(root)**

**root.protocol("WM\_DELETE\_WINDOW", app.close) # Закрытие базы данных при выходе**

**root.mainloop()**

**Шаг 4: Запуск приложения 1. Скопируйте код и сохраните его в файле athlete\_app.py. 2. Убедитесь, что у вас установлен Python. 3. Запустите приложение из командной строки: python athlete\_app.py.**

**Шаг 5: Объяснение кода 1. Класс Database: - Устанавливает соединение с базой данных SQLite, создает таблицу athletes (если она не существует), добавляет спортсменов и извлекает всех спортсменов из базы данных. 2. Класс AthleteApp: - Создает графический интерфейс для работы с базой данных. Содержит методы для добавления спортсменов и отображения их списка. 3. Взаимодействие с пользователем: - Пользователь вводит имя и время спортсмена, после чего данные добавляются в базу. Также есть возможность показывать всех спортсменов.**

**Заключение В результате выполнения лабораторной работы вы создали приложение для учета спортсменов с использованием базы данных SQLite. Вы научились добавлять данные, извлекать их и взаимодействовать с графическим интерфейсом. Это прекрасная основа для более сложных приложений, работающих с базами данных.**