

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
| **Институт математики и компьютерных технологий**  **департамент математического**  **и компьютерного моделирования** |

**Курсовой проект**

по дисциплинам

 «Вычислительная математика»

«Математическое и компьютерное моделирование»

Направление подготовки  
02.03.01 «Математика и компьютерные науки»

Выполнила студентка гр. Б9119-02.03.01сцт

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Панченко Н.К.\_

      (*подпись*)                 *(Ф.И.О.)*

Проверил

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

      (*подпись*)                 *(Ф.И.О.)*

«\_7\_\_»\_\_\_июня\_\_\_\_2022г.

**г. Владивосток**

**2021**

Оглавление

**Аннотация**

В данной работе рассматривается задача реализации персональной базы данных для бёрдвотчера. Решение выполнено с помощью интегрированной среды разработки PyCharm для языка программирования Python.

Введение

Бердвотчинг — это вид активного отдыха очень популярный в Америке и Западной Европе. Заключается в наблюдении за птицами, чаще всего с использованием бинокля или подзорной трубы, или фотографировании птиц, записи их голосов. Значительная часть научного орнитологического материала в мире собирается любителями-бёрдвотчерами. В тех уголках мира, куда бёрдвотчинг пришел в последние 25 лет, например в Азии и на пост-советском пространстве, в большей степени популярен фотобёрдвотчинг, то есть фотографирование птиц. Это связано в первую очередь с тем, что в это время широкой аудитории стали доступны как зеркальные камеры с телеобъективами, так и фотоаппараты-суперзумы. Каждый владелец такого фотоаппарата рано или поздно обязательно начинает делать фотографии птиц. Причин тому несколько. Во-первых, птицы заметны и хороши собой, но люди это замечают только взяв в руки бинокль или фотоаппарат. Во-вторых, птицы разнообразны, и у фотографа появляется желание находить и делать фотографии разных птиц. В-третьих, процесс фотоохоты сопровождается азартом и страстью, то есть, позволяет проявить древние инстинкты, но без крови. И в-четвертых, результат фотобердвотчинга — это прекрасные фотографии.

Цель создать базу данных где пользователь сможешь хранить собранную им информацию по каждому виду такую как: фотографии, описание, и т.д.

Основная часть

**Требования к окружению**

Для разработки использовалась среда разработки PyCharm c использованием языка программирования Python и встроенными библиотеками, и модулями.

Системные требования для установки PyCharm:

* 64 – бит версия Windows 10, 8;
* 8ГБ оперативной памяти;
* 2.5ГБ свободного пространства на диске;
* Python 2.7, или Python 3.5 или новее.

PyCharm

Это кроссплатформенная среда разработки, которая совместима с Windows, macOS, Linux для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django(программа).

Для работы использовался Python-3.9, библиотеки Tkinter, PIL и модули SQlite3, io.

Python

Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Стандартные библиотека включает большой объём полезных функций.

SQLite3

Модуль языка Python предоставляет интерфейс SQL, совместимый со спецификацией DB-API 2.0, описанной в PEP 249(спецификация API базы данных Python), и требует SQLite 3.7.15 или новее.

SQL

Декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

SQLite

SQLite — это библиотека C, которая предоставляет легкую дисковую базу данных, не требующую отдельного серверного процесса и позволяющую обращаться к базе данных с помощью нестандартного варианта языка запросов SQL. Некоторые приложения могут использовать SQLite для внутреннего хранения данных.

PIL(Python Imaging Library)

Библиотека языка Python (версии 2), предназначенная для работы с растровой графикой.

Возможности библиотеки:

* поддержка форматов BMP, EPS, GIF, JPEG, PDF, PNG, PNM, TIFF и некоторых других на чтение и запись;
* преобразование изображений из одного формата в другой;
* правка изображений (использование различных фильтров, масштабирование, рисование, матричные операции и т. д.).

IO

Модуль IO для выполнения операций ввода-вывода, связанные с файлами (например, чтение / запись файлов).

Tkinter

Это графическая библиотека, позволяющая создавать программы с оконным интерфейсом.

**Проект**

Методы и классы

Класс DB создаёт три таблицы для хранения информации с помощью запросов:

|  |
| --- |
| **self.c.execute(**  **'''CREATE TABLE IF NOT EXISTS birds (**  **id integer primary key, species text, date\_str, date\_int integer, place text, pic blob, notes text)''')**  **self.c.execute(**  **'''CREATE TABLE IF NOT EXISTS species (**  **species\_id integer primary key, name text, name\_lat text, name\_ing text, notes text, places text, biologi text, pic blob)''')**    **self.c.execute(**  **'''CREATE TABLE IF NOT EXISTS picture (**  **id integer primary key, name text, pic blob, pic\_min blob, pic\_mid blob)''')** |

**Листинг 1**

Класс Main запускает начальное окно программы.

Метод tk.Button – добавляет четыре кнопки действий(добавить категорию, добавить вид, добавить изображения, удалить). К кнопкам добавляются иконки.

Метод ttk.Treeview – создается таблица для отображения данных.

Метод tk.Scrollbar – добавляет полусу прокрутки для таблицы.

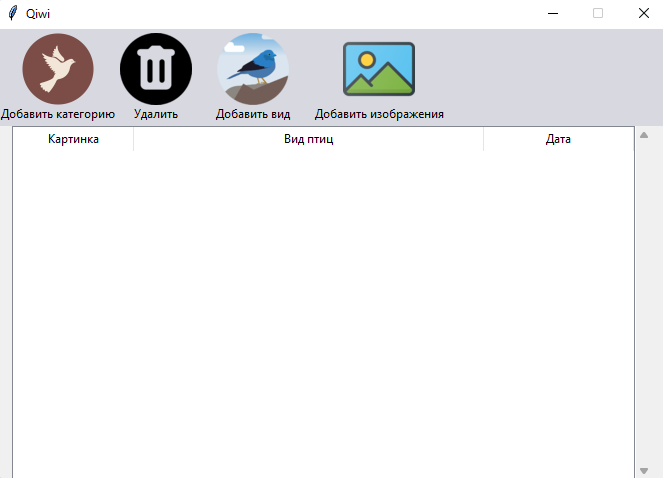
|  |
| --- |
| **tk.Button(toolbar, text='Добавить вид', bg='#d7d8e0', bd=0, image=self.species,**  **compound=tk.TOP, command=self.open\_dialog\_species)**  **ttk.Treeview(self, columns=('species', 'data'))**  **tk.Scrollbar(self, command=self.tree.yview)** |

**Листинг 2**

Метод view\_records – отображает данные в таблице.

|  |
| --- |
| **def view\_records(self):**  **self.db.c.execute('''SELECT pic, species, date\_str FROM birds''')**  **[self.tree.delete(i) for i in self.tree.get\_children()]**  **self.imglist = []**  **for record in self.db.c.fetchall():**  **img = ImageTk.PhotoImage(data=record[0])**  **self.tree.insert("", 'end', image=img, values=record[1:])**  **self.imglist.append(img)** |

**Листинг 3**



**Рисунок 1: основное окно приложения.**

Кнопка «Добавить вид» открывает окно, в котором можно создать новый вид для категорий.

Создаются поля для внесения информации методом ttk.Entry.

Для больших полей с описание используется метод tk.Text.

Для добавлений нужно нажать на кнопку «Добавить» вызовется метод который считает данные с полей и добавит данные в таблицу в которой хранится информация с видами.

|  |
| --- |
| **label\_speceis = tk.Label(self, text='Название вида на русском:')**  **label\_speceis.place(x=15, y=20)**  **self.entry\_species = ttk.Entry(self, width=25)**  **self.entry\_species.place(x=200, y=20)**  **label\_notes = tk.Label(self, text='Описание:')**  **label\_notes.place(x=15, y=130)**  **self.text\_notes = tk.Text(self,width=43, height=8)**  **self.text\_notes.place(x=13, y=160)**  **btn\_submit = tk.Button(self, text='Добавить', width=20, font=('Helvetica', 11, 'bold'))**  **btn\_submit.place(x=110, y=648)**  **btn\_submit.bind('<Button-1>', lambda event: self.view.records\_for\_speces(self.entry\_species.get(),**  **self.entry\_species\_lat.get(),**  **self.entry\_species\_en.get(),**  **self.text\_notes.get(1.0, 'end-1c'),**  **self.text\_places.get(1.0, 'end-1c'),**  **self.text\_biologi.get(1.0, 'end-1c')))** |

Листинг4

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 2: окно «Добавить вид».**

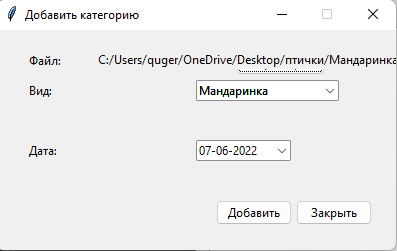
Кнопка «Добавить категорию» открывает окно, в котором вносятся данные для категории и добавляются в таблицу с категорией и отображается в таблице на главной странице.

В окне «Добавить категорию» расположены:

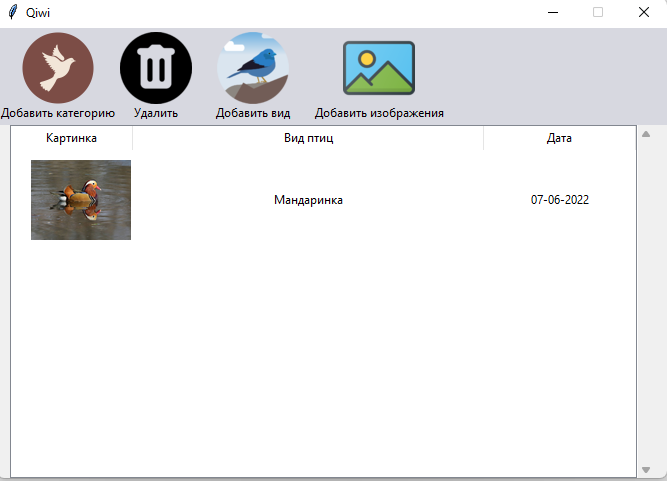
* кнопка «Выбрать файл» выбирается картинка для пред показа вида в категории;
* Выпадающий список, хранящий в себе добавленные виды;
* Поле даты для.
* Кнопка «Добавить» собирает данные и добавляет в таблицу для категорий, которые отображаются в таблице на главном окне.

|  |
| --- |
| **name\_values = self.get\_name()**  **self.entry\_species = ttk.Combobox(self, values=name\_values)**  **self.entry\_species.place(x=200, y=50)**  **btn\_file = ttk.Button(self, text='Выбрать файл', command=self.open\_file\_path)**  **btn\_file.place(x=240, y=20)**  **btn\_ok.bind('<Button-1>', lambda event: self.veiw.records(self.entry\_species.get(),**  **self.entry\_date.get(),**  **self.get\_timestamp\_from\_string(self.entry\_date.get()),**  **self.entry\_path.cget("text")))**  **def open\_file\_path(self):**  **filetypes = (('Image', '\*.jpg'), ('All files', '\*.\*'))**  **filename = fd.askopenfilename(title='Open a file', initialdir='/', filetypes=filetypes)**  **self.entry\_path = ttk.Label(self, text=filename)**  **self.entry\_path.place(x=100, y=20)** |

**Листинг 5**



**Рисунок 3: окно «Добавить категорию».**



**Рисунок 4: главное окно после добавления категории.**

Кнопка «Добавить изображение» открывает окно, в котором добавляются фотографии к виду.

В окне «Добавить категорию» расположены:

* Кнопка для выбора файла с изображением;
* Выпадающий список с выбором виду, к которому добавить изображение;
* Кнопка добавить, которая собирает выбранные данные и добавляет в таблицу с изображениями.

|  |
| --- |
| **name\_values = self.get\_name()**  **self.entry\_species = ttk.Combobox(self, values=name\_values,width=30)**  **self.entry\_species.place(x=150, y=50)**  **btn\_file = ttk.Button(self, text='Выбрать файл', command=self.open\_file\_path)**  **btn\_file.place(x=240, y=20)**  **btn\_ok = ttk.Button(self, text='Добавить')**  **btn\_ok.place(x=220, y=170)**  **btn\_ok.bind('<Button-1>', lambda event: self.veiw.records\_for\_picture(self.entry\_species.get(), self.entry\_path.cget("text")))** |

**Листинг 6.**

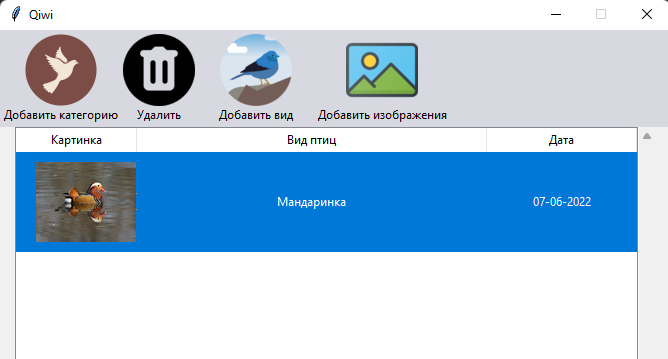
**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 5: окно «Добавить изображение».**

По двойному щелчку по выбраному элементу из таблицы на гланом экране откроется окно в котором собрана вся информация по виду и фотогорафии.

|  |
| --- |
| **def OnDoubleClick(self, event):**  **item = self.tree.identify('item', event.x, event.y)**  **self.open\_dialog\_view\_page()**  **def init\_veiw\_page(self):**  **self.title('Галерея')**  **self.geometry('1000x1000+400+1')**  **self.resizable(False, False)**  **data = self.view.get\_data\_species()**  **# print(data[0])**  **self.db.c.execute('''SELECT id, pic\_min, pic\_mid FROM picture WHERE name=?''', (data[0],))**  **e = self.get\_len\_data(data[3])**  **f = self.db.c.fetchall()**  **self.db.conn.commit()**  **self.a = self.get\_list\_pic\_mid(f)**  **self.imglist\_mid\_pic = []**  **for record in f:**  **img = ImageTk.PhotoImage(data=record[1])**  **self.imglist\_mid\_pic.append(img)**  **lbl\_name\_ru = ttk.Label(self, text=data[0],**  **font=("Arial", 28))**  **lbl\_name\_ru.place(x=40, y=10)**  **lbl\_name\_lat = ttk.Label(self, text=data[1],**  **font=("Arial", 16))**  **lbl\_name\_lat.place(x=40, y=60)**  **lbl\_name\_en = ttk.Label(self, text=data[2],**  **font=("Arial", 14))**  **lbl\_name\_en.place(x=40, y=90)**  **self.Artwork = tk.Label(self, image=self.a[0], background="black").place(x=40, y=120)**  **self.creat\_btn(f)**  **lbl\_notes = ttk.Label(self, text="Описание:", font=("Arial", 14))**  **lbl\_notes.place(x=40, y=610)**  **lbl\_notes\_text = ttk.Label(self, text=data[3], width= 130, font=("Arial", 10))**  **lbl\_notes\_text.place(x=40, y=640)**  **lbl\_notes\_text.bind('<Configure>', lambda e: lbl\_notes\_text.config(wraplength=lbl\_notes\_text.winfo\_width()))**  **lbl\_place = ttk.Label(self, text="Расположение:", font=("Arial", 14))**  **lbl\_place.place(x=40, y=730 + e)**  **lbl\_place\_text = ttk.Label(self, text=data[4], width=130, font=("Arial", 10))**  **lbl\_place\_text.place(x=40, y=760 + e)**  **lbl\_place\_text.bind('<Configure>', lambda e: lbl\_place\_text.config(wraplength=lbl\_place\_text.winfo\_width()))**  **lbl\_biologi = ttk.Label(self, text="Биология:",font=("Arial", 14))**  **lbl\_biologi.place(x=40, y=810 + e)**  **lbl\_biologi\_text = ttk.Label(self, text=data[5], width=130, font=("Arial", 10))**  **lbl\_biologi\_text.place(x=40, y=840 + e)**  **lbl\_biologi\_text.bind('<Configure>', lambda e: lbl\_biologi\_text.config(wraplength=lbl\_biologi\_text.winfo\_width()))** |



**Рисунок 6: выбранный элемент**