Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №6**

Дисциплина: «Разработка профессиональных приложений»

Классы

Вариант №19

Выполнил:

студент группы ИВТАПбд-21

Романов Н.Ю

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

Задание по варианту

Необходимо реализовать web-интерфейс для просмотра всех объектов, хранящихся в базе данных. Предметная область - из лабораторной работы 3 (4). Для доступа к данным желательно использовать ORM peewee (возможны альтернативные библиотеки). Для реализации web-части - cerrypy. Ajax использовать необязательно, можно сделать все на статических объектах.

Добавление / изменение необходимо реализовать минимум для 1 таблицы БД с внешними ключами.

Описание реализации

В данной работе основная реализация осуществлена с помощью фреймворка CherryPy и ORM Peewee. Оно позволяет работать с базой данных SQLite, хранящей информацию о каких-то товарах: их названии, стоимости и дате.

Класс MyTable определяет модель таблицы в базе данных и реализует функции для добавления, обновления и получения данных из нее.

Класс IndexPage представляет HTML-страницу, которая отображает данные из базы данных в виде таблицы.

Функция table\_style принимает список строк из базы данных и возвращает строку, содержащую HTML-разметку таблицы.

В основной ветке программы происходят следующие действия:

1. Создается объект класса MyTable
2. Получаются названия столбцов и значения строк из таблицы.
3. Формируется строка с HTML-разметкой таблицы.
4. Создается объект класса Page, передаются ему названия столбцов и строки таблицы.
5. Запускается веб-сервер CherryPy с помощью функции quickstart, которому передается объект класса Page.
6. После окончания работы веб-сервера закрывается соединение с базой данных.

**Описание возникших затруднений**

Затруднения возникли с изучением работы фреймворков CherryPy и ORM Peewee. А именно как создать страницу и расписать разметку страницы и стили таблицы, а также как правильно прочитать базу данных и с ней работать.

Описание альтернативных способов решения

Можно добавить возможность ввода данных пользователем через HTML-форму и реализовав другие операции с таблицей, такие как удаление строк или поиск.

Тестирование

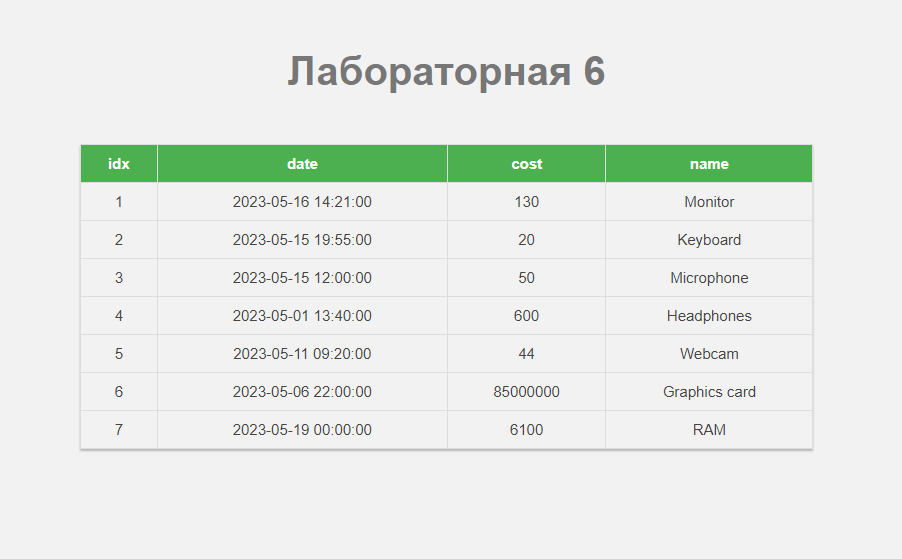


Рис. 1 Отображение таблицы  

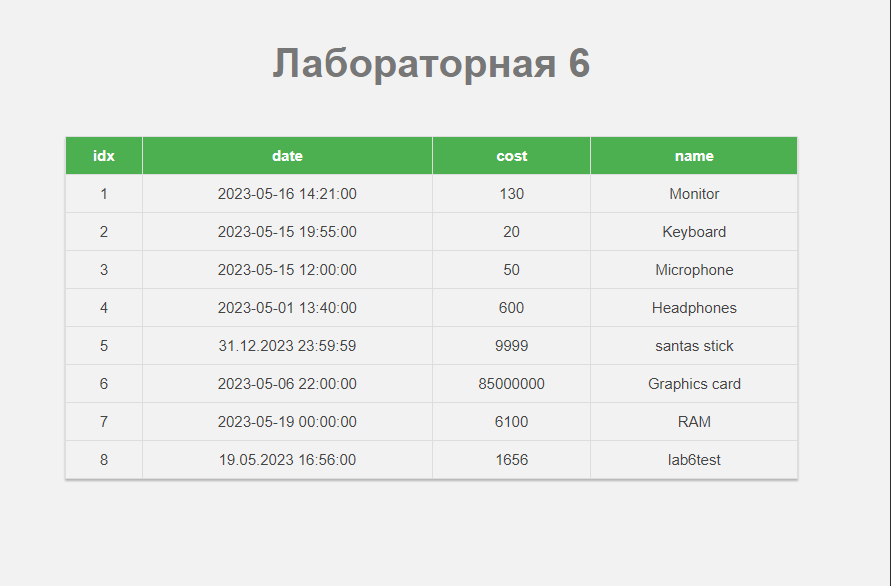



Рис 2-3. Добавление и изменение данных в таблице

Вывод

В данной лабораторной работе были приобретены навыки работы с фреймворками CherryPy и ORM Peewee. В ходе работы была сделана база данных с таблицей, содержащей индекс покупки, дату, стоимость и название продукта, а также сайт на котором отображаются данные таблицы.

Код программы

import cherrypy as cherrypy

from peewee import \*

db = SqliteDatabase('database.db')

class MyTable(Model):

class Meta:

database = db

db\_table = 'lab6'

idx = PrimaryKeyField(unique=True)

date = DateField()

cost = IntegerField()

name = TextField()

def Add(self, date, cost, name):

MyTable(date=date, cost=cost, name=name).save()

def Update(self, idx, date, cost, name):

table = MyTable.get(idx=idx)

table.date = date

table.cost = cost

table.name = name

table.save()

def getColumn(self):

cursor = db.cursor()

cursor.execute('PRAGMA table\_info("lab6")')

columns = [i[1] for i in cursor.fetchall()]

return columns

def getStrings(self):

cursor = db.cursor()

sqlite\_select\_query = """SELECT \* from lab6"""

cursor.execute(sqlite\_select\_query)

result = cursor.fetchall()

return result

class IndexPage(object):

def \_\_init\_\_(self, columns, data):

self.columns = columns

self.data = data

@cherrypy.expose

def index(self):

return f'''

<html>

<head>

<meta charset="utf-8">

<title>Лаба 6</title>

<style>

body {{

font-family: Arial, sans-serif;

background-color: #f2f2f2;

color: #444;

margin: 0;

padding: 0;

}}

h1 {{

color: #777;

text-align: center;

margin-top: 50px;

font-size: 40px;

}}

table {{

margin: auto;

max-width: 800px;

width: 80%;

box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.25);

border-collapse: collapse;

margin-top: 50px;

}}

td, th {{

border: 1px solid #ddd;

padding: 10px;

text-align: center;

font-size: 15px;

}}

th {{

background-color: #4CAF50;

color: white;

}}

tr , td{{

transition: 0.2s ease-in-out;

}}

tr:hover {{

transform: scale(1.05);

background-color: #91cf93;

cursor: pointer;

}}

td:hover {{

color: white;

transform: scale(1.05);

background-color: #b4deb5;

}}

</style>

</head>

<body>

<h1>Лабораторная 6</h1>

<table>

<tr>

{

"".join(["<th>" + i + "</th>"

for i in self.columns])

}

</tr>

{self.data}

</table>

</body>

</html>

'''

def table\_style(strings):

table\_line = ""

for tr in strings:

table\_line += "<tr>"

for td in tr:

table\_line += f"<td style='padding: 10px; text-align:center; font-size: 15px;" \

f" border: 1px solid #ddd;'> {td} </td>"

table\_line += "</tr>"

return table\_line

if \_\_name\_\_ == '\_\_main\_\_':

table = MyTable()

# table.Add("19.05.2023 16:56:00", 1656, "lab6test")

# table.Update(5, "31.12.2023 23:59:59", 9999, "santas stick")

columns = table.getColumn()

lines = table.getStrings()

table\_line = table\_style(lines)

cherrypy.quickstart(IndexPage(columns, table\_line))

db.close()