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»

Web-разработка

Вариант №22

Выполнил:

студент группы ИВТАСбд-21

Старостин К. А.

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 – История столовой: №, наименование блюда, количество в граммах, время подачи, время замены (во сколько закончилось), остаток. Использовать ORM peewee для доступа к данным. Для реализации web-части - cerrypy. Также сделать возможность добавления и изменения записи.

# Описание реализации

Для реализации web-интерфейса и просмотра объектов базы данных была использована ORM peewee и фреймворк CherryPy. Требовалось выполнить следующие шаги:

1. Установить peewee и CherryPy с помощью pip.
2. Создать модели данных с использованием peewee, отражающие структуру базы данных.

|  |
| --- |
| from peewee import \*  db = SqliteDatabase('database.db')  class User(Model):  name = CharField()  # Дополнительные поля  class Meta:  database = db  class Product(Model):  name = CharField()  user = ForeignKeyField(User, backref='products')  # Дополнительные поля  class Meta:  database = db  # Создание таблиц в базе данных  db.create\_tables([User, Product]) |

1. Создать веб-интерфейс с использованием CherryPy. Определить необходимые маршруты и методы для отображения и обработки данных:

|  |
| --- |
| import cherrypy  # Класс-контроллер CherryPy  class WebInterface(object):  @cherrypy.expose  def index(self):  # Получение всех объектов из базы данных  users = User.select()  # Генерация HTML-страницы для отображения пользователей  html = '<h1>Пользователи</h1>'  for user in users:  html += f'<p>{user.name}</p>'  return html  # Запуск CherryPy  cherrypy.quickstart(WebInterface()) |

1. Запустить веб-сервер, выполнив скрипт и перейдя по адресу, который отобразится в консоли. Можно увидеть созданную страницу с подключенной базой данных.

Описание методов, которые используется в этой лабораторной работе.

## Метод index

Данный метод предназначен для обработки главной страницы веб-интерфейса. Метод работает следующим образом: получает все записи из базы данных. Генерирует HTML-страницу для отображения списка истории столовой в виде таблицы, выводит информацию о каждой столовой (включая ID, наименование блюда, количество в граммах, время подачи, время замены (во сколько закончилось), остаток), добавляет ссылку "Изменить" для каждой столовой (ссылка указывает на метод edit\_visit с передачей ID столовой в качестве параметра запроса), выводит форму для добавления новых посещений, возвращает сгенерированную HTML-страницу.

## Метод add\_history

Данный метод предназначен для добавления новой истории столовой.

Метод работает следующим образом: получает данные из формы отправки, создаёт новый объект с переданными данными, сохраняет объект в базе данных, возвращает сообщение об успешном добавлении записи.

## Метод edit\_canteen

Данный метод предназначен для редактирования существующих историй столовых.

Метод работает следующим образом: получает ID записи из параметра запроса, извлекает соответствующий объект из базы данных, генерирует HTML-страницу с предварительно заполненной формой, отправляет форму в метод update\_history, возвращает сгенерированную HTML-страницу.

## Метод update\_visit

Данный метод предназначен для обновления существующих историй столовой. Метод работает следующим образом: получает данные из формы отправки, находит соответствующие объекты в базе данных, обновляет значения объекта с переданными данными, сохраняет изменения в базе данных, возвращает сообщение об успешном обновлении записи.

# Описание возникших затруднений

В ходе лабораторной работы затруднений не возникло.

# Листинг кода

import cherrypy

from peewee import \*

db = SqliteDatabase('data\_base.db')

# модель для Истории столовой

class DiningRoom(Model):

id = AutoField()

Name\_dish = CharField()

Amount\_grams = IntegerField()

Serving\_time = TimeField()

Rep\_time = TimeField()

Rem\_dish = IntegerField()

class Meta:

database = db # модель будет использовать базу данных data\_dabe.db

db.create\_tables([DiningRoom]) # создание таблицы в базе данных

class WebInterface(object):

@cherrypy.expose

def index(self):

canteens = DiningRoom.select() # получение объектов из базы данных

# генерация HTML-страницы для отображения истории столовой

html = '<head><link rel="stylesheet" href="style.css"></head>'

html += '<body><h1>История столовой</h1>'

html += '<table>'

html += '<tr><th>№</th>' \

'<th>Название блюда</th>' \

'<th>Кол-во граммов</th>' \

'<th>Время подачи</th>' \

'<th>Время замены</th>' \

'<th>Остаток</th>' \

'<th>Изменить</th></tr>'

for canteen in canteens:

html += f'<tr><td>{canteen.id}</td><td>{canteen.Name\_dish}</td><td>{canteen.Amount\_grams}</td><td>{canteen.Serving\_time}</td><td>{canteen.Rep\_time}</td><td>{canteen.Rem\_dish}</td>'

html += f'<td><a href="/edit\_canteen?id={canteen.id}">Изменить</a></td></tr>'

html += '</table>'

# форма для добавления записей

html += '''

<h2>Добавить истрию:</h2>

<form method="post" action="add\_history">

<input type="text" name="name\_dish" placeholder="Название блюда" required><br>

<input type="number" name="amount\_grams" placeholder="Кол-во граммов" required><br>

<input type="time" name="serving\_time" placeholder="Время подачи" required><br>

<input type="time" name="rep\_time" placeholder="Время замены" required><br>

<input type="number" name="rem\_dish" placeholder="Остаток" required><br>

<input type="submit" value="Добавить">

</form>

'''

html += '</body>'

return html

@cherrypy.expose

def add\_history(self, name\_dish, amount\_grams, serving\_time, rep\_time, rem\_dish):

# создание новой истории столовой

canteen = DiningRoom(Name\_dish=name\_dish,

Amount\_grams=amount\_grams,

Serving\_time=serving\_time,

Rep\_time=rep\_time,

Rem\_dish=rem\_dish)

canteen.save()

return 'История столовой добавлена успешно.'

@cherrypy.expose

def edit\_canteen(self, id):

canteen = DiningRoom.get(DiningRoom.id == id)

html = '<head><link rel="stylesheet" href="style.css"></head>'

html += '<body>'

html += f'<h2>Изменение истории столовой (ID: {canteen.id}):</h2>'

html += f'<form method="post" action="update\_visit?id={canteen.id}">'

html += f'<input type="text" name="name\_dish" placeholder="Название блюда" value="{canteen.Name\_dish}" required><br>'

html += f'<input type="text" name="amount\_grams" placeholder="Кол-во в граммах" value="{canteen.Amount\_grams}" required><br>'

html += f'<input type="time" name="serving\_time" placeholder="Время подачи" value="{canteen.Serving\_time}" required><br>'

html += f'<input type="time" name="rep\_time" placeholder="Время замены" value="{canteen.Rep\_time}" required><br>'

html += f'<input type="тгьиук" name="rem\_dish" placeholder="Остаток" value="{canteen.Rem\_dish}" required><br>'

html += '<input type="submit" value="Обновить">'

html += '</form>'

html += '</body>'

return html

@cherrypy.expose

def update\_visit(self, id, name\_dish, amount\_grams, serving\_time, rep\_time, rem\_dish):

canteen = DiningRoom.get(DiningRoom.id == id)

canteen.Name\_dish = name\_dish

canteen.Amount\_grams = amount\_grams

canteen.Serving\_time = serving\_time

canteen.Rep\_time = rep\_time

canteen.Rem\_dish = rem\_dish

canteen.save()

return f'Посещение (ID: {canteen.id}) успешно обновлено!'

# Запуск CherryPy

cherrypy.quickstart(WebInterface())

# Вывод

Был разработан web-интерфейс для просмотра объектов, хранящихся в базе данных «История столовой». В работе использовались ORM peewee и фреймворк CherryPy. Реализация включала следующие шаги: установку peewee и CherryPy с помощью pip, создание моделей данных с использованием peewee, создание веб-интерфейса с использованием CherryPy и определение необходимых маршрутов и методов, а также запуск веб-сервера для проверки работоспособности программы через браузер.