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 №8

Выполнил:

студент группы ИВТАСбд-21

Зюзин Г.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 посещений поликлиники: №, ФИО пациента, ФИО врача, причина обращения, длительность. Использовать ORM peewee. 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:

|  |
| --- |
| pip install peewee  pip install cherrypy |

1. Создать модели данных с использованием peewee, отражающие структуру вашей 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  # Конфигурация CherryPy  cherrypy.config.update({'server.socket\_host': '0.0.0.0',  'server.socket\_port': 8080})  # Запуск CherryPy  cherrypy.quickstart(WebInterface()) |

1. Запустить веб-сервер, выполнив скрипт и перейдя по адресу http://localhost:8080 в своем браузере. Можно увидеть список пользователей, полученных из базы данных.

Описание методов, которые используется в этой лабораторной работе:

1. Метод index:
   1. Описание: данный метод предназначен для обработки главной страницы веб-интерфейса.
   2. Работа метода:
      * Получить все записи из базы данных.
      * Сгенерировать HTML-страницу для отображения списка посещений в виде таблицы.
      * Вывести информацию о каждом посещении, включая ID, ФИО пациента, ФИО врача, причину обращения и длительность.
      * Добавить ссылку "Изменить" для каждого посещения, которая будет указывать на метод edit\_visit с передачей ID посещения в качестве параметра запроса. Вывести форму для добавления новых посещений.
      * Вернуть сгенерированную HTML-страницу.
2. Метод add\_visit:
   1. Описание: данный метод предназначен для добавления новых записей о посещениях.
   2. Работа метода:
      * Получить данные из формы отправки.
      * Создать новый объект VisitHistory с переданными данными.
      * Сохранить объект в базе данных.
      * Вернуть сообщение о успешном добавлении записи.
3. Метод edit\_visit:
   1. Описание: данный метод предназначен для редактирования существующих записей о посещениях.
   2. Работа метода:
      * Получить ID посещения из параметра запроса.
      * Извлечь соответствующий объект VisitHistory из базы данных.
      * Сгенерировать HTML-страницу с формой, предварительно заполненной текущими значениями посещения.
      * Отправить форму на метод update\_visit.
      * Вернуть сгенерированную HTML-страницу для редактирования посещения.
4. Метод update\_visit:
   1. Описание: данный метод предназначен для обновления существующих записей о посещениях.
   2. Работа метода:
      * Получить данные из формы отправки.
      * Найти соответствующий объект VisitHistory в базе данных.
      * Обновить значения объекта с переданными данными.
      * Сохранить изменения в базе данных.
      * Вернуть сообщение об успешном обновлении записи.

**Описание возникших затруднений**

Изначально была попытка выполнить данную лабораторную работу через WSL. Но возникла проблема с установкой ORM peewee и фреймворка CherryPy. Для решения данной проблемы был создан новый проект в PyCharm без WSL.

**Описание альтернативных способов решения.**

Альтернативных способов решения задачи состоит в использовании сервера Flask.

**Листинг кода.**

from peewee import \*  
import cherrypy  
  
# Подключение к базе данных  
db = SqliteDatabase('database.db')  
CP\_CFG = {  
 '/': {  
 'tools.staticdir.on': True,  
 'tools.staticdir.dir': 'C:/Users/georg/PycharmProjects/RPP/lab6'  
 }  
}  
  
  
# Модель для таблицы История посещений  
class VisitHistory(Model):  
 id = AutoField()  
 patient\_name = CharField()  
 doctor\_name = CharField()  
 reason = TextField()  
 duration = IntegerField()  
  
 class Meta:  
 database = db  
  
  
# Создание таблиц в базе данных  
db.create\_tables([VisitHistory])  
  
  
# Класс-контроллер CherryPy  
class WebInterface(object):  
 @cherrypy.expose  
 def index(self):  
 # Получение всех объектов из базы данных  
 visits = VisitHistory.select()  
  
 # Генерация HTML-страницы для отображения посещений  
 html = '<head><link rel="stylesheet" href="style.css"></head>'  
 html += '<body><h1>История посещений поликлиники</h1>'  
 html += '<table>'  
 html += '<tr><th>№</th><th>ФИО пациента</th><th>ФИО врача</th><th>Причина обращения</th><th>Длительность</th><th>Действия</th></tr>'  
  
 for visit in visits:  
 html += f'<tr><td>{visit.id}</td><td>{visit.patient\_name}</td><td>{visit.doctor\_name}</td><td>{visit.reason}</td><td>{visit.duration}</td>'  
 html += f'<td><a href="/edit\_visit?id={visit.id}">Изменить</a></td></tr>'  
  
 html += '</table>'  
  
 # Форма для добавления записей  
 html += '''  
 <h2>Добавить посещение:</h2>  
 <form method="post" action="add\_visit">  
 <input type="text" name="patient\_name" placeholder="ФИО пациента" required><br>  
 <input type="text" name="doctor\_name" placeholder="ФИО врача" required><br>  
 <textarea name="reason" placeholder="Причина обращения" required></textarea><br>  
 <input type="number" name="duration" placeholder="Длительность" required><br>  
 <input type="submit" value="Добавить">  
 </form>  
 '''  
  
 html += '</body>'  
  
 return html  
  
 @cherrypy.expose  
 def add\_visit(self, patient\_name, doctor\_name, reason, duration):  
 # Создание нового объекта посещения  
 visit = VisitHistory(patient\_name=patient\_name, doctor\_name=doctor\_name, reason=reason, duration=duration)  
 visit.save()  
  
 return 'Посещение успешно добавлено!'  
  
 @cherrypy.expose  
 def edit\_visit(self, id):  
 visit = VisitHistory.get(VisitHistory.id == id)  
  
 html = '<head><link rel="stylesheet" href="style.css"></head>'  
 html += '<body>'  
 html += f'<h2>Изменение посещения (ID: {visit.id}):</h2>'  
 html += f'<form method="post" action="update\_visit?id={visit.id}">'  
 html += f'<input type="text" name="patient\_name" placeholder="ФИО пациента" value="{visit.patient\_name}" required><br>'  
 html += f'<input type="text" name="doctor\_name" placeholder="ФИО врача" value="{visit.doctor\_name}" required><br>'  
 html += f'<textarea name="reason" placeholder="Причина обращения" required>{visit.reason}</textarea><br>'  
 html += f'<input type="number" name="duration" placeholder="Длительность" value="{visit.duration}" required><br>'  
 html += '<input type="submit" value="Обновить">'  
 html += '</form>'  
 html += '</body>'  
  
 return html  
  
 @cherrypy.expose  
 def update\_visit(self, id, patient\_name, doctor\_name, reason, duration):  
 visit = VisitHistory.get(VisitHistory.id == id)  
 visit.patient\_name = patient\_name  
 visit.doctor\_name = doctor\_name  
 visit.reason = reason  
 visit.duration = duration  
 visit.save()  
  
 return f'Посещение (ID: {visit.id}) успешно обновлено!'  
  
  
# Конфигурация CherryPy  
cherrypy.config.update({'server.socket\_host': '0.0.0.0',  
 'server.socket\_port': 8080})  
  
# Запуск CherryPy  
cherrypy.quickstart(WebInterface(), config=CP\_CFG)

**Вывод.**

Был разработан web-интерфейс для просмотра объектов, хранящихся в базе данных истории посещений поликлиники. 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. В работе были использованы методы index, add\_visit, edit\_visit и update\_visit для отображения и обработки данных.