```
public string Email { get; set; }
public string Phone { get; set; }
public string Address { get; set; }
public virtual ICollection<WorkingTime> WorkingTime { get; set; }
public virtual ICollection<ApplicationUser> Users { get; set; }
public virtual ICollection<Employee> Employees { get; set; }
public virtual ICollection<Service> Services { get; set; }
}
```

Остальные модели реализованы аналогично.

В папке Services же находятся все сервисы, в которых и заключена вся логика работы приложения. Данные являются связующим звеном между серверным слоем и слоем доступа к данным. В данной работе представлены следующие сервисы:

– AccountService. Необходим для работы с администраторами;

– CompanyService. Отвечает за обработку данных компании;

– EmployeeService. Данный сервис выполняет все действия, связанные с сотрудниками компании;

– ProfileService. Данный сервис необходим для предоставления некоторых данных в теле токена авторизации;

– ReservationService. Данный сервис работает непосредственно с бронированиями;

– ResourceService. Сервис, отвечающий за ресурсы;

– ServiceService. Сервис, в котором заключена вся логика по работе с услугами;

– TimeService. Основной сервис, который предоставляет все свободные временные зоны для бронирований.

Данный подход к обработке и хранению данных может быть предназначен для любой экологической организации, которая располагает некими услугами, для более эффективной работы в своей сфере, а также для надежности хранения ценной информации. К тому же позволяет экономить человеческие усилия, которые можно направить в другое русло, и, само собой, бумажные и другие ресурсы.

ЛИТЕРАТУРА

1. Архитектура межсервисного взаимодействия, https://secretmag.ru/business/methods/7-servisov-dlya-onlajn-zapisi.htm, [Электронный ресурс].

2. Модель взаимодействия клиент-сервера, https://ru.wikipedia.org/wiki/, [Электронный ресурс].

3. Семь сервисов для онлайн записи, https://secretmag.ru/business/methods/7-servisov-dlya-onlajn-zapisi.htm, [Электронный ресурс].

4. .Net Framework, https://www.guru99.com/net-framework.html, [Электронный ресурс].

5. .Net Core, https://metanit.com/sharp/aspnet5/, [Электронный ресурс].

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ ДЛЯ ХРАНЕНИЯ ИНФОРМАЦИИ О ДИПЛОМНЫХ, АСПИРАНТСКИХ И МАГИСТЕРСКИХ РАБОТАХ

DEVELOPMENT OF A WEB-APPLICATION FOR STORING INFORMATION ABOUT GRADUATE, POST-GRADUATE AND MASTER'S WORKS

А. Л. Карпей, Д. А. Кормужанин

A. L. Karpei, D. A. Kormuzhanin

Белорусский государственный университет, МГЭИ им. А. Д. Сахарова БГУ
г. Минск, Республика Беларусь

E-mail: kar_an@tut.by
Belarusian State University, ISEI BSU
Minsk, Republic of Belarus

С использованием языка программирования Python с фреймворком Django разработано web-приложение, хранящее информацию о дипломных работах студентов, аспирантских и магистерских диссертациях. Web-приложение связано с базой данных SQLite и предоставляет возможности добавления, изменения и удаления информации из базы данных. Для создания приложения были использованы среда разработки PyCharm и программа для просмотра информации о базе данных DB Browser for SQLite.

Using the Python programming language with the Django framework, a web-application has been developed that stores information about students' theses, postgraduate and master's theses. The web application is linked to a SQLite database and provides the ability to add, modify, and remove information from the database. To develop the

application, the PyCharm development environment and the program for viewing information about the DB Browser for SQLite database were used.

Ключевые слова: хранение информации, дипломные и диссертационные работы, Python, фреймворк Django, база данных SQLite.

Keywords: information storage, theses and dissertations, Python, Django framework, SQLite database.

https://doi.org/10.46646/SAKH-2021-2-411-414

Основной целью работы является создание web-приложения, которое дает возможность хранить информацию о дипломных работах студентов, аспирантских и магистерских диссертациях. Его разработка обусловлена потребностью в ней МГЭИ им. А.Д. Сахарова БГУ. Основными задачами web-приложения являются: хранение информации, работа с ней, возможность добавления, изменения и удаления информации, возможность расширения функционала web-приложения в будущем. Web-приложение должно быть связано с нормализованной базой данных.

Для разработки web-приложения был выбран язык программирования Python с фреймворком Django.

Язык программирования 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 нем программ. Python является мультипарадигмальным языком программирования, поддерживающим императивное, процедурное, структурное и объектно-ориентированное программирование.

Для того чтобы написать web-приложение с помощью Python, существует множество web-фреймворков. Web-фреймворк обрабатывает как минимум запросы клиента и ответы сервера. Он может предоставлять следующие возможности: маршруты – интерпретирует URL и находит соответствующие файлы на сервере или серверный код Python, шаблоны – объединяет серверные данные в страницы HTML, аутентификация и авторизация – обрабатывает имена пользователей, пароли, разрешения, сессии – обслуживает временное хранилище данных во время посещения пользователем [1].

Django – один из самых популярных фреймворков для создания web-приложений, написанный на языке программирования Python. Web-приложение на Django строится из одного или нескольких приложений, которые рекомендуется делать независимыми друг от друга и подключаемыми. Это является существенным отличием этого фреймворка от других. Один из основных принципов – не повторять себя. Каждый раз при разработке web-приложения требуются похожие компоненты: способ аутентификации пользователей, панель администратора, формы, менеджер загрузки файлов и так далее. Данный фреймворк позволяет использовать готовые шаблоны для ускорения и упрощения разработки.

Для удобства работы с фреймворком была выбрана среда разработки PyCharm. Выбор среды разработки был обусловлен возможностями среды разработки, такими как виртуальная среда для каждого проекта, удобная навигация между файлами и папками проекта и поддержкой работы с фреймворком Django.

Система управления базами данных – совокупность программных средств, позволяющих управлять и создавать базы данных.

Фреймворк Django поддерживает работу со следующими системами управления базами данных: SQLite, MySQL, PostgreSQL. Для разработки web-приложения была выбрана система управления базами данных SQLite. SQLite является легко встраиваемой в приложения базой данных. Так как эта система базируется на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми системами управления базами данных. При работе с SQLite обращения происходят напрямую к файлам (в эти файлах хранятся данные), вместо портов и сокетов в сетевых системах управления базами данных. Именно поэтому SQLite очень быстрая, а также мощная благодаря технологиям обслуживающих библиотек [2].

При проектировании базы данных было выделено три таблицы для хранения (рис.1).

Таблица “Обучающиеся” хранит в себе информацию о студентах. Первичным ключом в ней является ID обучающегося, который позволяет однозначно идентифицировать студента. Также важным полем является категория студента, с помощью которого можно отдельно выделить работы студентов, магистрантов и аспирантов. Таблица “Обучающиеся” связана с таблицей “Исследования связью один к одному, так как один обучающийся проводит одно исследование.

Таблица с информацией о руководителях связана с таблицей исследования связью один ко многим, так как один руководитель может участвовать в нескольких исследованиях.

В таблице “исследования” с помощью внешнего ключа прямо указываются студент и преподаватель, которые занимались исследованием.

Скриншот созданной базы данных показан на рисунке 2.

Фреймворк Django имеет строгую структуру файлов в web-приложении. На рисунке 3 показана схема обработки HTTP запроса.

Рис. 1 – Инфологическая модель базы данных

thesis_research		CREATE TABLE "thesis_research" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "topic" varchar(500) NOT NULL, "pur
id	integer	"id" integer NOT NULL
topic	varchar(500)	"topic" varchar(500) NOT NULL
purpose	text	"purpose" text NOT NULL
objectives	text	"objectives" text NOT NULL
results	text	"results" text NOT NULL
scope	varchar(100)	"scope" varchar(100) NOT NULL
organization	varchar(100)	"organization" varchar(100) NOT NULL
explanation	text	"explanation" text
stage	varchar(50)	"stage" varchar(50)
date	date	"date" date NOT NULL
student_id	integer	"student_id" integer UNIQUE
teacher_id	integer	"teacher_id" integer
thesis_student		CREATE TABLE "thesis_student" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "full_name" varchar(200) NOT NULL,
id	integer	"id" integer NOT NULL
full_name	varchar(200)	"full_name" varchar(200) NOT NULL
specialty	varchar(50)	"specialty" varchar(50) NOT NULL
form	varchar(70)	"form" varchar(70) NOT NULL
phone_number	varchar(18)	"phone_number" varchar(18)
email	varchar(30)	"email" varchar(30)
date_start	date	"date_start" date NOT NULL
date_end	date	"date_end" date NOT NULL
category	varchar(1)	"category" varchar(1) NOT NULL
thesis_teacher		CREATE TABLE "thesis_teacher" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "full_name" varchar(200) NOT NULL,
id	integer	"id" integer NOT NULL
full_name	varchar(200)	"full_name" varchar(200) NOT NULL
degree	varchar(50)	"degree" varchar(50) NOT NULL
email	varchar(30)	"email" varchar(30)
phone_number	varchar(18)	"phone_number" varchar(18)

Рис. 2 – Созданная база данных

Файл urls.py содержит в себе функции для обработки каждого ресурса приложения. Он используется для перенаправления HTTP-запроса в соответствующее представление. Так же может передавать данные из URL-адреса в функцию в файле views.py в качестве аргумента.

Файл views.py является обработчиком запросов, вызываемым из urls.py. Функции, описанные в этом файле, имеют доступ к данным, описанным в models.py, и могут передавать эти данные на HTML-шаблон.

Файл models.py представляет собой модели, описывающие структуру данных приложения, и предоставляет возможность для управления, добавления и удаления, а также выполнения запросов в базе данных.

Каталог Templates содержит в себе структуру или разметку HTML-страниц. Views.py может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели (models.py). Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

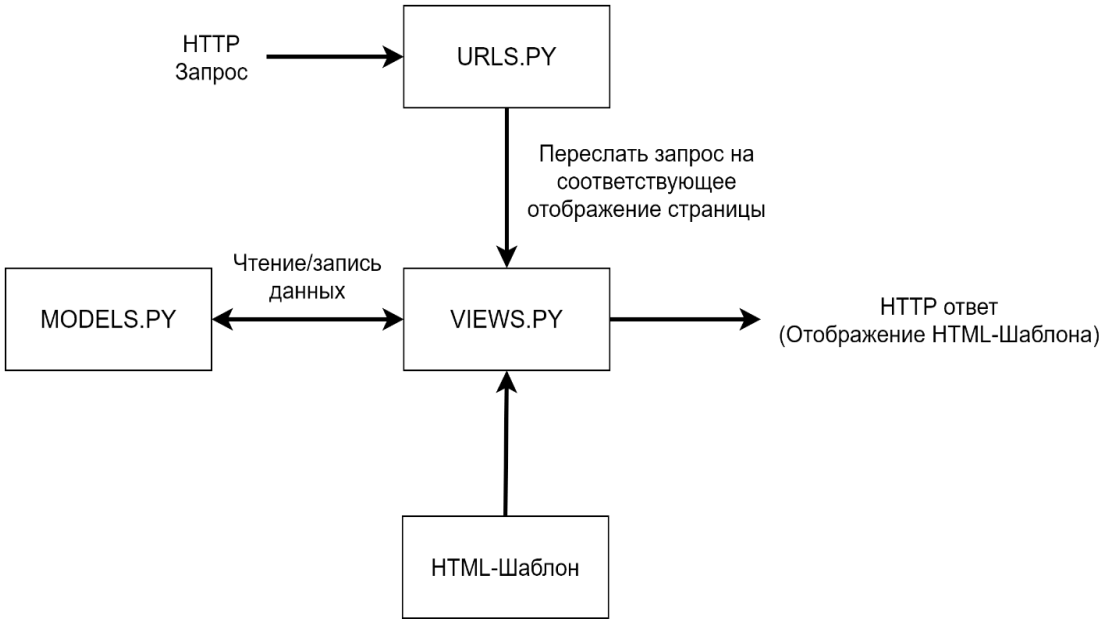

Рис.3. Схема обработки HTTP запроса

Для передачи информации из базы данных используются функции. Функции отвечают за отображение страницы и передачу информации из базы данных. Для отображения информации из базы данных на HTML-странице следует обращаться к нужной таблице, как к объекту описанного класса. Так же с помощью функций можно управлять существующими записями в базе данных, либо добавлять новые (рис.4).

```
127  def EditTeacher(request, pk):
128      try:
129          teacher = Teacher.objects.get(pk=pk)
130          if request.POST:
131              teacher.full_name = request.POST.get("full_name")
132              teacher.degree = request.POST.get("degree")
133              teacher.phone_number = request.POST.get("phone_number")
134              teacher.email = request.POST.get("email")
135              teacher.save()
136              return HttpResponseRedirect(reverse('teachers'))
137          else:
138              return render(request, "thesis/edit_teacher.html", {"teacher": teacher})
139      except Teacher.DoesNotExist:
140          return HttpResponseNotFound("<h2>Преподаватель не найден</h2>")
141
```

Рис.4. Пример функции в файле views.py

Фреймворк Django работает с базами данных с помощью моделей, обращаясь к ним как к объектам. Для создания простейшей модели в папке с приложением и создается файл models.py, в котором и объявляются классы, которые будут записаны в базу данных. После создания нужных классов в Django следует сделать миграции. Миграции в Django являются системой контроля версий базы данных. После создания миграций следует их применить. Если система контроля не выдала ошибок, в базе данных будут созданы соответствующие таблицы [3].

Разработанный пользовательский интерфейс web-приложения позволяет использовать функции добавления аккаунтов пользователей с разграничением полномочий по работе с информацией в базе данных.

ЛИТЕРАТУРА

1. *Любанович Б.* Простой Python. Современный стиль программирования/ Б. Любанович – Санкт-Петербург: Изд-во Питер, 2016. - 480 с.

2. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных [Электронный ресурс]. – Режим доступа https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql – Дата доступа 15.03.2021.

3. Документация Django 1.9 [Электронный ресурс]. – Режим доступа https://djbook.ru/rel1.9/topics/migrations.html – Дата доступа 25.02.2021.