**Практическая работа 3. Создание проекта django**

**Задание:** создать проект и приложение ИС (на django). Далее в приложении составить список url маршрутов, по которым можно будет переходить пользователям ИС (что соответствует их возможным действиям на сайте).  Примеры страниц: главная страница с информацией о компании, контакты, список вакансий, товары компании, страница склада с информацией о наличии товара.

Содержание страниц:

- главная страница – html файл, содержащий заголовок -  название компании и навигация сайта – список маршрутов: ссылки для перехода на все остальные его страницы (можете конкретизировать предприятие, например, мебельная компания или компания, торгующая косметическими средствами, или стройматериалами или др.).

- на остальных страницах  - только заголовок (название страницы)

**Пример:**

**1. Создать проект python**,

который будет находится в определенной созданной для него директории (папке).

В случае, если используется pyCharm, проект может быть создан следующим образом:

В PyCharm выбрать пункты меню: File->New Project. При этом желательно создать для созданного проекта виртуальное окружение (такой вариант предлагается в проекте по умолчанию). Появится окно создания проекта следующего вида:

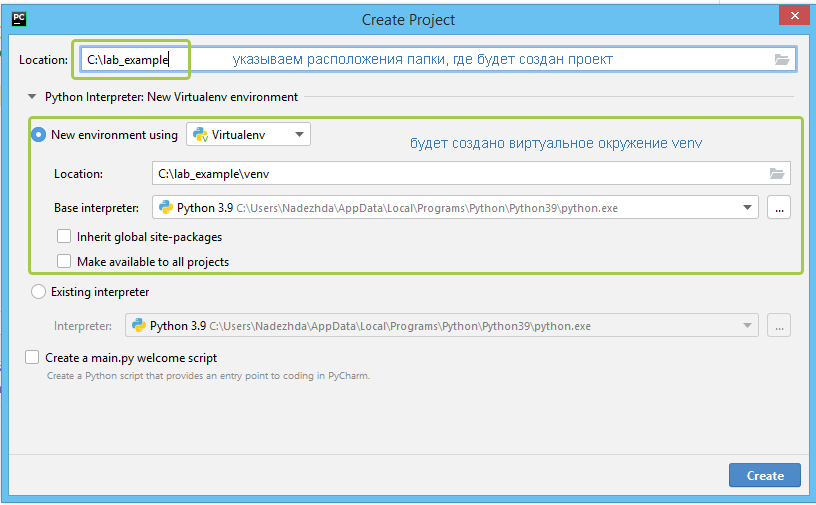


Рисунок. создание проекта с новым виртуальным окружением

2. Далее переходим в терминал проекта. Требуется **активировать виртуальное окружение** (если оно ранее не было активировано, при создании нового проекта оно активируется автоматически, то есть данный шаг пропускаем, активировать может быть необходимо при повторном открытие проекта, но не всегда это требуется)

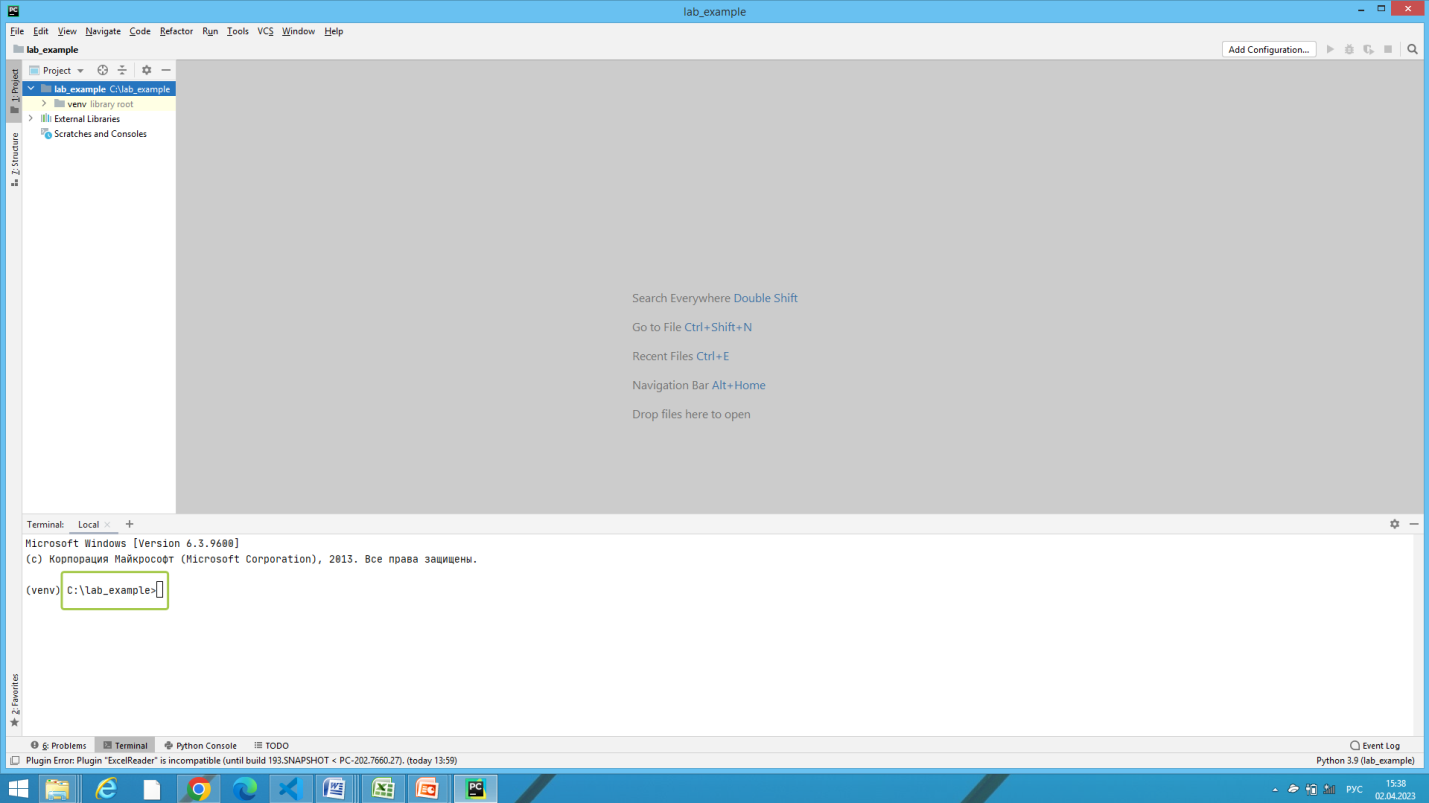


Рисунок. Пример созданного проекта с виртуальным окружением venv

Здесь нужно обратить внимание на путь в терминале (убедиться, что он совпадает с тем, в котором хотите создать проект, если отличается, с помощью команды cd в терминале перейти в нужную папку). Название виртуального окружения (venv) отображается не всегда. Также, в зависимости от библиотеки для создания виртуального окружения, в папке с проектом может появится папка venv или нет (такой вариант тоже возможен и не является ошибкой).

3. **Установить библиотеку django**

(написать в терминале команду pip install django)

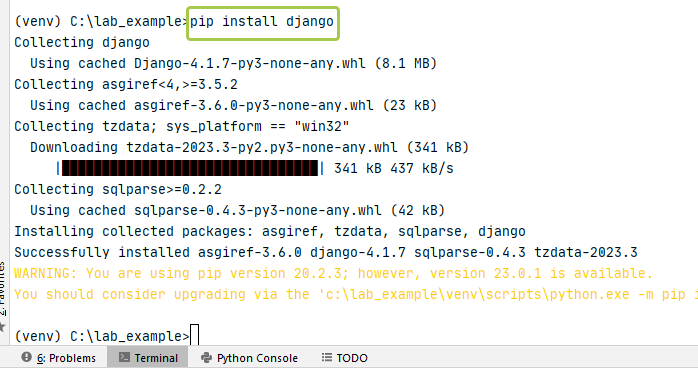


Рисунок. установка библиотеки

4. **Создание проекта.**

При установке Django в папке виртуальной среды устанавливается утилита **django-admin**. **Она** предоставляет команды для управления проектом Django. Для создания проекта применяется команда **startproject** (в качестве аргумента передается название проекта).

django-admin startproject Название\_проекта.

Например, создадим проект с именем "example\_project".

c:lab\_example>django-admin startproject example\_project

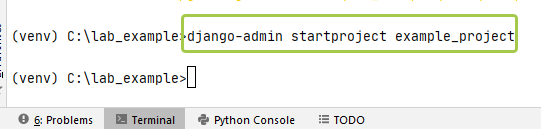


Рисунок. Создание проекта

После выполнения этой команды в текущей папке (в примере - c:\lab\_example) будет создан каталог example\_project.

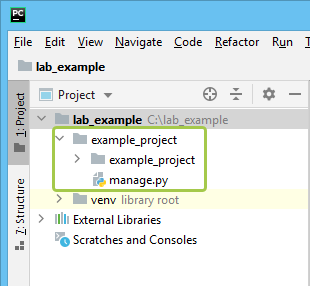


Рисунок. Папка созданного проекта django

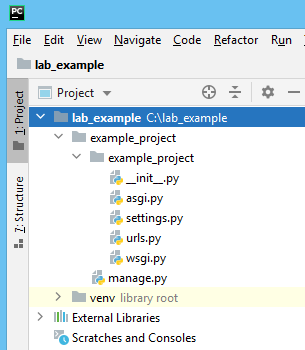


Рисунок. Структура файлов созданного проекта django

Созданный проект состоит из следующих файлов:

* **manage.py**: выполняет различные команды проекта, например, создает и запускает приложение
* **example\_app** - папка проекта, которая содержит следующие файлы:
  + **\_\_init\_\_.py**: данный файл указывает, что папка, в которой он находится, будет рассматриваться как модуль. Это стандартный файл для программы на языке Python.
  + **settings.py**: содержит настройки конфигурации проекта
  + **urls.py**: содержит шаблоны URL-адресов, по сути определяет систему маршрутизации проекта
  + **wsgi.py**: содержит свойства конфигурации WSGI (Web Server Gateway Inerface). Он используется при развертывании проекта.
  + **asgi.py**: название файла - сокращение от Asynchronous Server Gateway Interface и расширяет возможности WSGI, добавляя поддержку для взаимодействия между асинхронными веб-серверами и приложениями.

5. **Запуск проекта.**

Для этого с помощью команды cd перейдем в консоли к папке проекта (в которой находится файл manage.py). И затем для запуска проекта выполним следующую команду:

python manage.py runserver

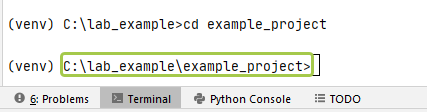


Рисунок. Переход в папку проекта

После запуска проекта в консоли будет доступен адрес, по которому запущен проект. По умолчанию проект запускается на **http://127.0.0.1:8000/**.

## 

## Рисунок. Запуск проекта

Откроем веб-браузер и введем данный адрес в адресную строку браузера. (Для остановки сервера, чтобы писать следующие команды, в терминале требуется нажать Cntrl+C)

## 

## Рисунок. Отображение сервера в браузере

**6. Создание приложения**

Веб-приложение или проект Django состоит из отдельных приложений. При создании проекта он уже содержит несколько приложений по умолчанию. Список всех приложений находится в проекте в файле **settings.py** в переменной **INSTALLED\_APPS**.

|  |  |
| --- | --- |
|  |  |

Далее требуется создать приложение, которое будет реализовать определенный функционал.

Для создания приложения в проекте применяется команда, которую нужно написать в терминале

|  |  |
| --- | --- |
|  | python manage.py startapp название\_приложения |

Например, создадим в проекте первое приложение, которое будет называться **example\_app**.

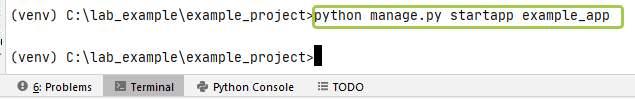
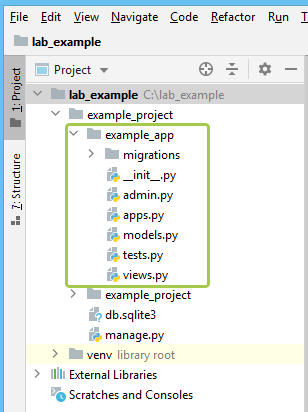


Рисунок. Создание приложения

И после выполнения этой команды мы увидим в проекте новую папку, которое будет хранить все файлы созданного приложения:



## Рисунок. Файлы приложения

Рассмотрим его структуру:

* папка **migrations**: предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с определением моделей
* **\_\_init\_\_.py**: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета
* **admin.py**: предназначен для административных функций, в частности, здесь призводится регистрация моделей, которые используются в интерфейсе администратора
* **apps.py**: определяет конфигурацию приложения
* **models.py**: хранит определение моделей, которые описывают используемые в приложении данные
* **tests.py**: хранит тесты приложения
* **views.py**: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ

**7. Подключение приложения к проекту.**

Для этого откроем файл **settings.py** и добавим в массив INSTALLED\_APPS название созданного приложения:

INSTALLED\_APPS = [  
 **'example\_app',** **'django.contrib.admin'**,  
 **'django.contrib.auth'**,  
 **'django.contrib.contenttypes'**,  
 **'django.contrib.sessions'**,  
 **'django.contrib.messages'**,  
 **'django.contrib.staticfiles'**,  
]

|  |  |
| --- | --- |
|  |  |

В проекте может быть несколько приложений, каждое из которых требуется добавлять аналогичным образом.

**8. Добавление функций представления**

Важным элементов любого веб-приложения является обработка запроса, который отправляет пользователь. В Django за обработку запроса отвечают представления или **views**. Представления - функции обработки, которые принимают данные запроса и генерируют некоторый результат, который затем отправляется пользователю. Представления размещаются в приложении в файле **views.py**.

Создадим основные функции представления для заданного проекта: магазина. В соответствии с заданием требуется создать следующие функции (каждая функция будет соответствовать обработке запроса – перехода пользователя на страницу):

* главная страница с информацией о компании,
* контакты,
* список вакансий,
* товары компании,
* страница склада с информацией о наличии товара

Содержимое функций, согласно заданию, следующее:

- главная страница – должна вернуть html файл, содержащий заголовок -  название компании и навигация сайта – список маршрутов: ссылки для перехода на все остальные его страницы

- на остальных страницах  - только заголовок (название страницы)

Перейдем в папку приложения **example\_app** к файлу **views.py**, и создадим функции следующим образом:

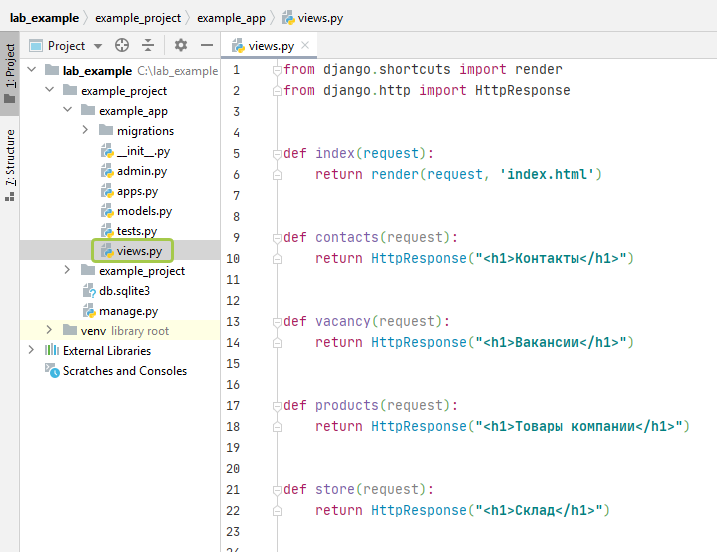


Рисунок. Функции приложения (здесь на главной странице используется файл index.html, который создадим позже)

В данном модуле мы импортируем класс **HttpResponse** из стандартного пакета *django.http*. Затем определяются функции, которые в качестве параметра получают объект запроса request. Класс HttpResponse предназначен для создания ответа, который отправляется пользователю.

Из модуля django.shortcuts импортируется функция **render**. Функция index вызывает функцию render, которой передаются объект запроса request и путь к файлу шаблона в рамках папки templates - "index.html".

Чтобы функция сопоставлялась с запросом, требуется определить для нее маршрут в проекте в файле urls.py (что сделаем позже).

**9. Добавление шаблонов (html страниц приложения)**

**9.1 Создание папки шаблонов**

Шаблоны (templates) отвечают за формирование внешнего вида приложения. Они предоставляют специальный синтаксис, который позволяет внедрять данные в код HTML.

Настройка функциональности шаблонов в проекте Django производится в файле **settings.py**. с помощью переменной **TEMPLATES**.

Добавим в папку приложения каталог **templates**.

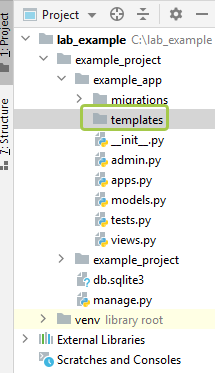


Рисунок. Добавление папки шаблонов

**9.2. Создание html файл(ов)**

В папке **templates** создадим файл **index.html**:

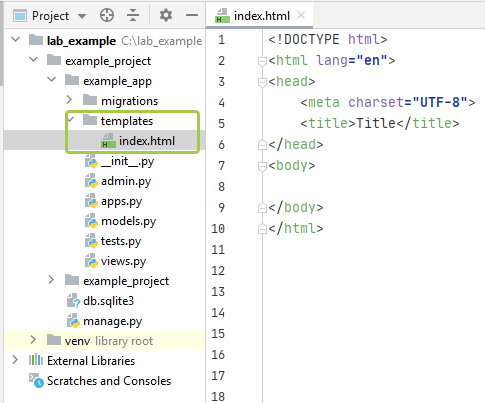


Рисунок. Добавление файла шаблонов

Далее в файле **index.html** определим следующий код: добавим заголовок страницы и навигацию по страницам (ссылки, на которые сможет переходить пользователь, которые создадим в соответствии с требованиями в задании)

|  |  |
| --- | --- |
|  |  |

<!DOCTYPE html>  
<html lang**="en"**>  
<head>  
 <meta charset**="UTF-8"**>  
 <title>Приложение для работников магазина</title>  
</head>  
<body>  
<h1>Главная страница приложения</h1>  
<nav>  
 <a href**=""**>Главная страница</a>  
 <a href**=""**>Контакты</a>  
 <a href**=""**>Список вакансий</a>  
 <a href**=""**>Товары компании</a>  
 <a href**=""**>Наличие на складе</a>  
</nav>  
</body>  
</html>

Так как маршруты пока не созданы, то ссылки a никуда не ведут (исправим это далее)

**10 Маршрутизация**

В данном случае под маршрутизацией понимается набор возможных переходов по ссылкам внутри приложения (т.е. его страницам).

**10.1 Маршрутизация приложения.**

Ранее в примере в файл **views.py** были добавлены функции. Для того, чтобы связать их с действиями пользователя, создадим файл **urls.py (в папке приложения example\_app)**

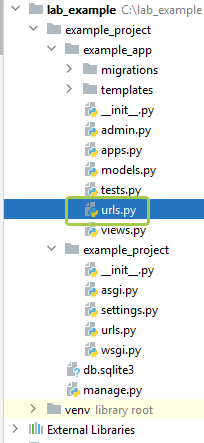
****

Рисунок. Создан файл для описания маршрутов приложения

**и опишем маршруты** для каждой функции следующим образом:

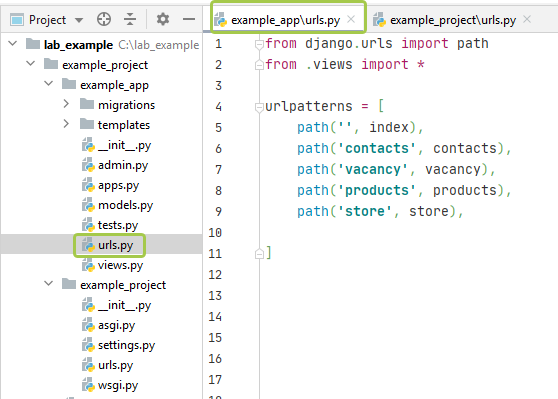
****

Рисунок. Описание маршрутов приложения

Переменная **urlpatterns** определяет набор сопоставлений функций обработки с определенными строками запроса. Например, запрос к корню веб-сайта будет обрабатываться функцией index.

**10.2 Подключение маршрутов приложения к проекту**

Маршруты в проекте Django задаются в файле **urls.py**. Этот файл находится в папке проекта, в примере, в папке example\_projects и выглядит следующим образом:

**from** django.contrib **import** admin  
**from** django.urls **import** path  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
]

Первой строкой из модуля *django.contrib* импортируется класс AdminSite, который предоставляет возможности работы с интерфейсом администратора. Второй строкой из модуля *django.urls* импортируется функция **path**. Эта функция задает сопоставление определенного маршрута с функцией обработки. Так, в данном случае маршрут "admin/" будет обрабатываться методом admin.site.urls. (который относится к приложению, устанавливаемому с django по умолчанию).

В данном проекте был добавлен маршрут в приложение, далее нужно добавить его в файл маршрутизации проекта. Перейдем в файл проекта urls.py. Добавим к списку маршрутов проекта подключение к маршрутам приложения (к файлу приложения urls.py)



Рисунок. Подключение маршрутов приложения (example\_app – название приложения)

**11. Добавление ссылок на страницу**

Вернемся к файлу index.hml и добавим ссылки следующим образом (для элементов <a> укажем маршруты, выбранные из urls.py приложения, цветом выделен добавленный код):

<!DOCTYPE html>  
<html lang**="en"**>  
<head>  
 <meta charset**="UTF-8"**>  
 <title>Приложение для работников магазина</title>  
</head>  
<body>  
<h1>Главная страница приложения</h1>  
<nav>  
 **<a href="">Главная страница</a>  
 <a href="contacts">Контакты</a>  
 <a href="vacancy">Список вакансий</a>  
 <a href="products">Товары компании</a>  
 <a href="store">Наличие на складе</a>**</nav>  
</body>  
</html>

**12. Запуск приложения**

Далее запустим приложение командой

python manage.py runserver

И перейдем в браузере по адресу *http://127.0.0.1:8000/*

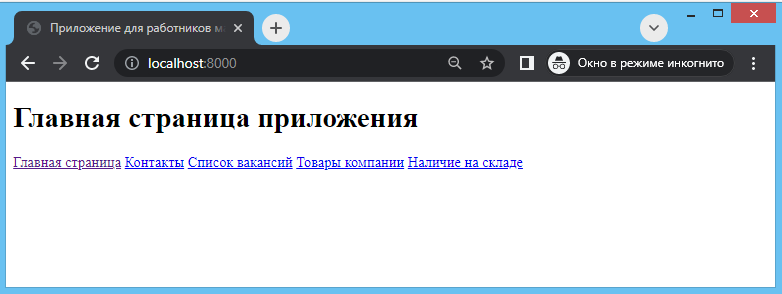


Рисунок. Пример отображения страницы приложения в браузере (главная страница)

Видно, что в браузере при переходе по адресам маршрутов будут выполняться соответствующие им функции.

## 

Рисунок. Пример отображения страницы приложения в браузере (страница контакты)

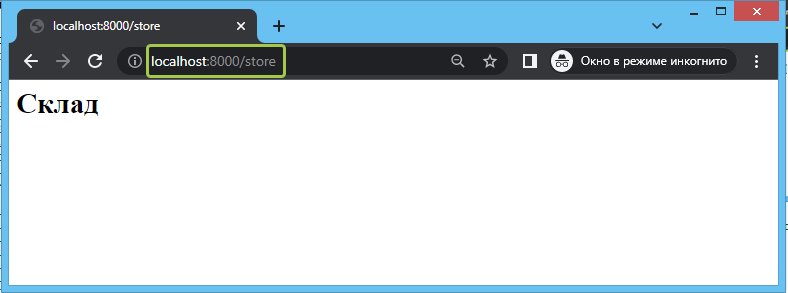


Рисунок. Пример отображения страницы приложения в браузере (страница склад)