1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Реализация backend и аутентификации**

по дисциплине «Безопасность интернет приложений»

1. Выполнили Берко А.С.
2. студенты гр. 4851003/90801 Кулеева А.Г.

Реденко И.Д.



1. Руководитель
2. ассистент Климшин И.И.
4. Санкт-Петербург
5. 2023

# Задание

1. Регистрация в Интернет-приложении. Двухфакторная аутентификация в Интернет-приложении. Второй фактор в аутентификации выбирается самостоятельно, главное требование исходит из определения — второй фактор подтверждение не должен доставляться по тому же каналу, что и первый.
2. Бэкэнд с готовым API, оформленный в Swagger. API должен покрывать функционал, который будет доступен пользователю приложения. Для этого необходимо изначально прикинуть, какие функции будут в вашем Интернет-приложении, и на основе функций спроектировать API.
3. Интерфейс не обязателен, даже в аутентификациях.

# Ход работы

## Создание и начальная настройка проекта

Было принято решение разрабатывать интернет-приложение с помощью фреймворка Python Django. Для этого был использован интерпретатор Python версии 3.10.5, а также сам фреймворк Django версии 4.1.7 (на данный момент является крайней версией). После установки фреймворка в терминале была введена команда python -m django startproject bip, после чего был создан проект с некоторыми начальными параметрами. Разберем созданные файлы:

**manage.py** — является ссылкой на скрипт django-admin, но с уже предустановленными переменными окружения, указывающими на проект, как для чтения настроек оттуда, так и для управления им при необходимости;

**settings.py** — здесь находятся настройки проекта. Файл уже содержит несколько разумных настроек:

* Секретный ключ;
* Режим отладки (DEBUG = True);
* Установленные приложения;
* Некоторые ссылки;
* Параметры работы с базами данных. В качестве СУБД была выбрана SQLite, установленная по умолчанию с фреймворком;
* Поддерживаемый язык (был изменен на русский);
* Часовой пояс (был оставлен по умолчанию);

**urls.py** — содержит URL'ы для отображения представлений;

**wsgi.py** — это WSGI обёртка для приложения. Этот файл используется сервером разработки Django и возможно другими контейнерами, такими как mod\_wsgi, uwsgi и др. на «боевом» сервере;

**models.py** будет содержать Django ORM-модели для приложения;

**views.py** будет содержать код представлений;

**tests.py** будет содержать написанные разработчиками модульные и интеграционные тесты.

Проверим, что проект работает корректно. Введем в терминале команду python manage.py runserver и получим ссылку на локальный сервер (рисунок 1).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 — Запуск локального сервера

Перейдем по полученной ссылке (рисунок 2).

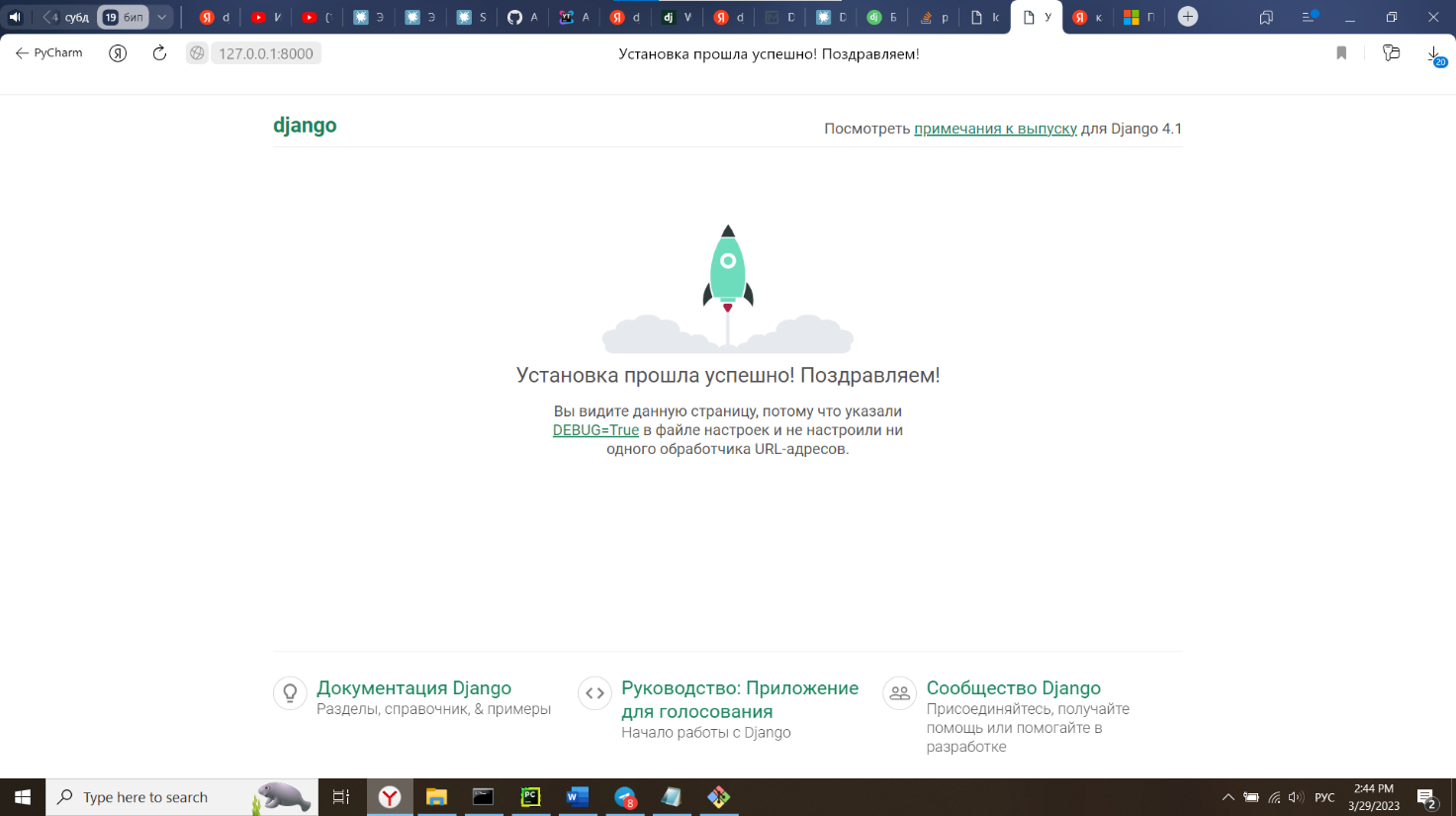


Рисунок 2 — Вид полученного сайта

Была создана учетная запись администратора с помощью команды python manage.py createsuperuser (рисунок 3).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 — Вход в панель администратора

## Основные приложения

Всего в проекте 4 приложения (страницы сайта). Они были зарегистрированы в файле settings.py, а ссылки на них были прописаны в файле urls.py.

Все ссылки перенаправляют на одноименный файл, но внутри данного приложения. В локальном файле ссылок вызывается метод из файла views.py, который отправляет HTTP-response. Ответ осуществляется с помощью метода render, аргументом для которого служит файл HTML. В данном файле прописана минимальная разметка страницы, чтобы можно было тестировать привязку к базе данных.

### Главная страница

На данной странице располагается шапка, содержащая основную информацию об Интернет-приложении и список объявлений от благотворительных организаций. Также есть кнопка «Войти», которая перенаправляет пользователя на страницу входа. Если пользователь уже аутентифицировался, то на этом месте будет кнопка «Мой профиль», которая перенаправит пользователя на соответствующую страницу.

Была создана модель списка объявлений (рисунок 4).

class Desk(models.Model):  
 title = models.CharField('Краткое описание', max\_length=255)  
  
 def \_\_str\_\_(self):  
 return self.title

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 — Созданная модель

### Мой профиль

Данная страница содержит информацию об организации:

* Форма организации
* Руководитель
* Название
* ОГРНИП/ОГРН
* ИНН/КПП
* Адрес регистрации
* Фактический адрес
* Дата регистрации
* Контактные данные

Также каждой организации присвоен тип: ресторан или благотворительная организация. В случае последней будут отображаться все открытые заявки и кнопка для добавления новой заявки. Для обоих типов будет история заявок, либо откликов (в зависимости от типа), а также оценка и отзывы.

Можно ли редактировать профиль, если в теории мы выгружаем данные с госуслуг? Зачем вручную изменять всякие ИНН и тд? С дургой стороны, а если изменилось юр лицо или ещё что-то? Или как-то описать, что будет подключена автообнова с госуслуг?

### Страница входа

Данная страница должна открывать в отдельном окне в целях безопасности. Пользователю предлагается два варианта: войти и зарегистрироваться. При входе запрашивается только логин, пароль, а также второй фактор аутентификации. Если пользователь хочет создать профиль, то он должен придумать логин и пароль, а также заполнить все поля с информацией об организации. Предполагается, что информация будет проверяться через госуслуги, однако в целях тестирования дополнительные проверки были отключены.

### Страница объявления

При выборе из списка объявлений на главной странице пользователь будет перенаправлен на отдельную страницу. Там содержится более подробное описание требуемой помощи, информация об организации авторе объявления и кнопка откликнуться для пользователей-ресторанов. Также автор объявления может его редактировать.