**ДИПЛОМНАЯ РАБОТА**

На тему:

# Анализ и сравнение написания web-приложений с использованием разных фреймворков.

Необходимо:

## Разработать простые веб-приложения с использованием Django, Flask и FastAPI. Провести их сравнение.

### Urban University,

### курс "Разработчик Python",

### студент Федосов М.А.

### Содержание

1. Задачи дипломной работы
2. Описание дипломной работы
3. Реализация дипломной работы

## 1. Задачи дипломной работы:

Целью данной работы является сравнение возможностей web-фреймворков на базе Python: Flask, FastAPI и Django. Реализовать небольшие веб-приложения на каждом фреймворке, провести сравнения, удобство разработки и сопровождения.

## 2. Описание дипломной работы:

В качестве тем для разработки я выбрал создание баз данных для учета, контроля и соблюдения сроков исполнения входящей корреспонденции так как показывает практика, что соблюдение исполнительской дисциплины значительно влияет на продуктивность деятельности учреждения(компании). Разработанные web-приложения должны обеспечивать выполнение следующих функций:

1. Персонализацию приложения при регистрации;
2. Возможность вносить планируемые задачи c кратким описанием;
3. Устанавливать сроки исполнения;
4. Многозадачность;
5. Возможность просмотра и обновления задачи(при необходимости)

## 3. Реализация дипломной работы

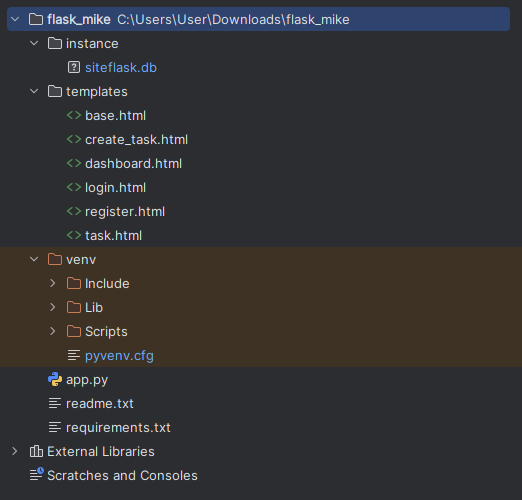
**Flask**

Flask — это микровеб-фреймворк с открытым исходным кодом, дружелюбный к новичкам, написанный на Python, который следует архитектурному шаблону MVC (модель-представление-контроллер) и поддерживает шаблоны Jinja, которые можно использовать для создания представлений.

Требования к функционалу вебприложения Список задач (To-Do List).

1. Регистрация и авторизация пользователей с сохранением в базе данных.
2. После авторизации пользователь попадает в форму Дашбод.
3. Требования к web-приложению Список задач (To-Do List).
   * Добавление, редактирование и удаление задач.
   * Отметка задач как выполненных.
   * Фильтрация задач по статусу (все, активные, выполненные).
   * Возможность сортировки по дате создания или приоритету.

Структура директории flask\_mike



## 1. Введение

Данное руководство предназначено для пользователей сайта База данных учета и исполнения входящей переписки организации, учета задач учреждения, разработанного на фреймворке Flask в рамках дипломной работы и показывает порядок работы с web-приложением.

## 2. Запуск сайта

### 2.1. Настройка окружения

Перед первым запуском необходимо создать виртуальное окружение Python. Для этого выполняем следующую команду в командной строке:

python -m venv venv

- следующей командой устанавливаем пакеты:

pip install -r requirements.txt

Дальнейший запуск сайта необходимо выполнять только работая в данном виртуальном окружении.

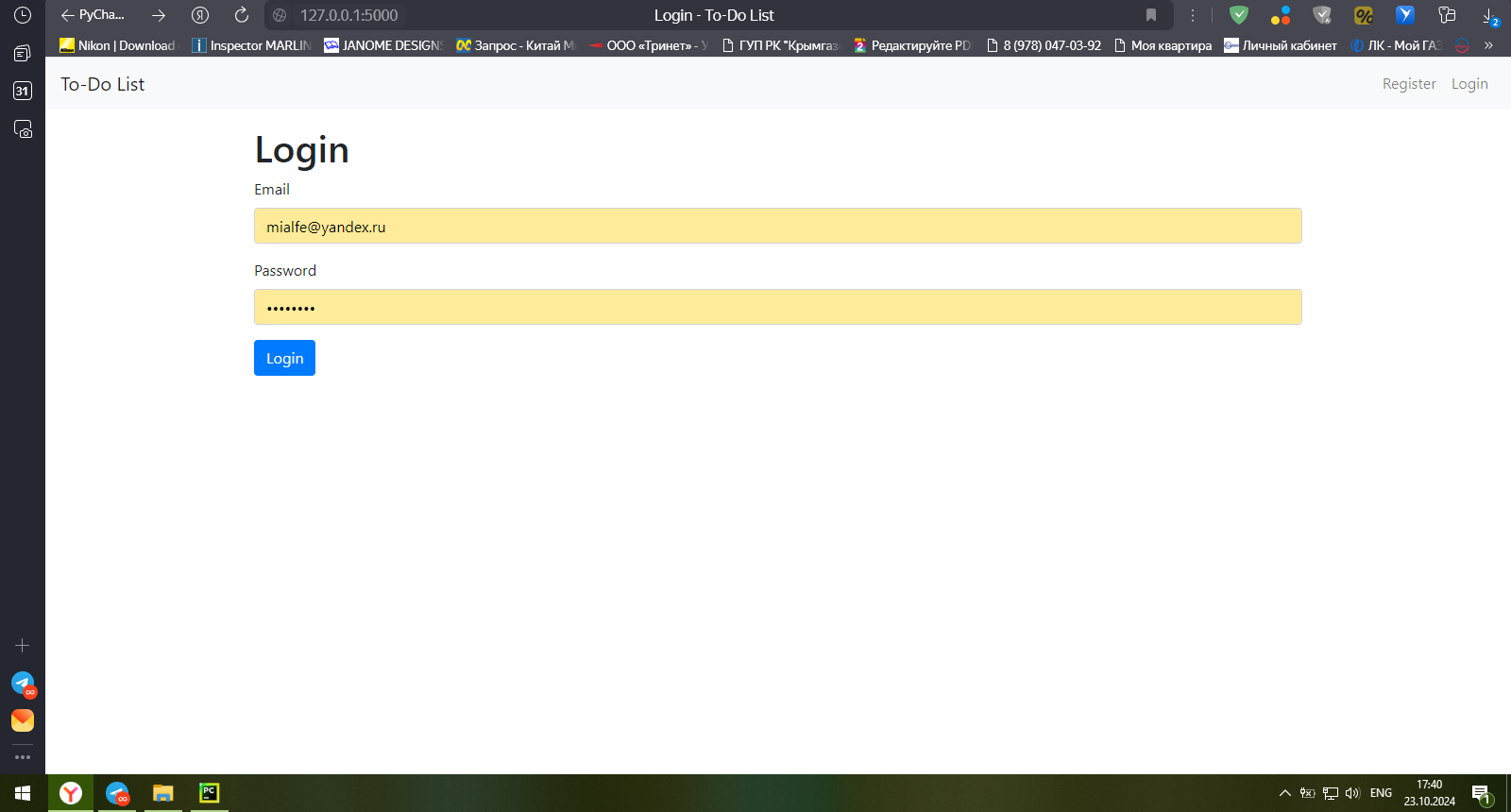
### 2.2. Запуск сайта

Для запуска сайта необходимо перейти в корневой каталог сайта Flask и запустить файл app.py

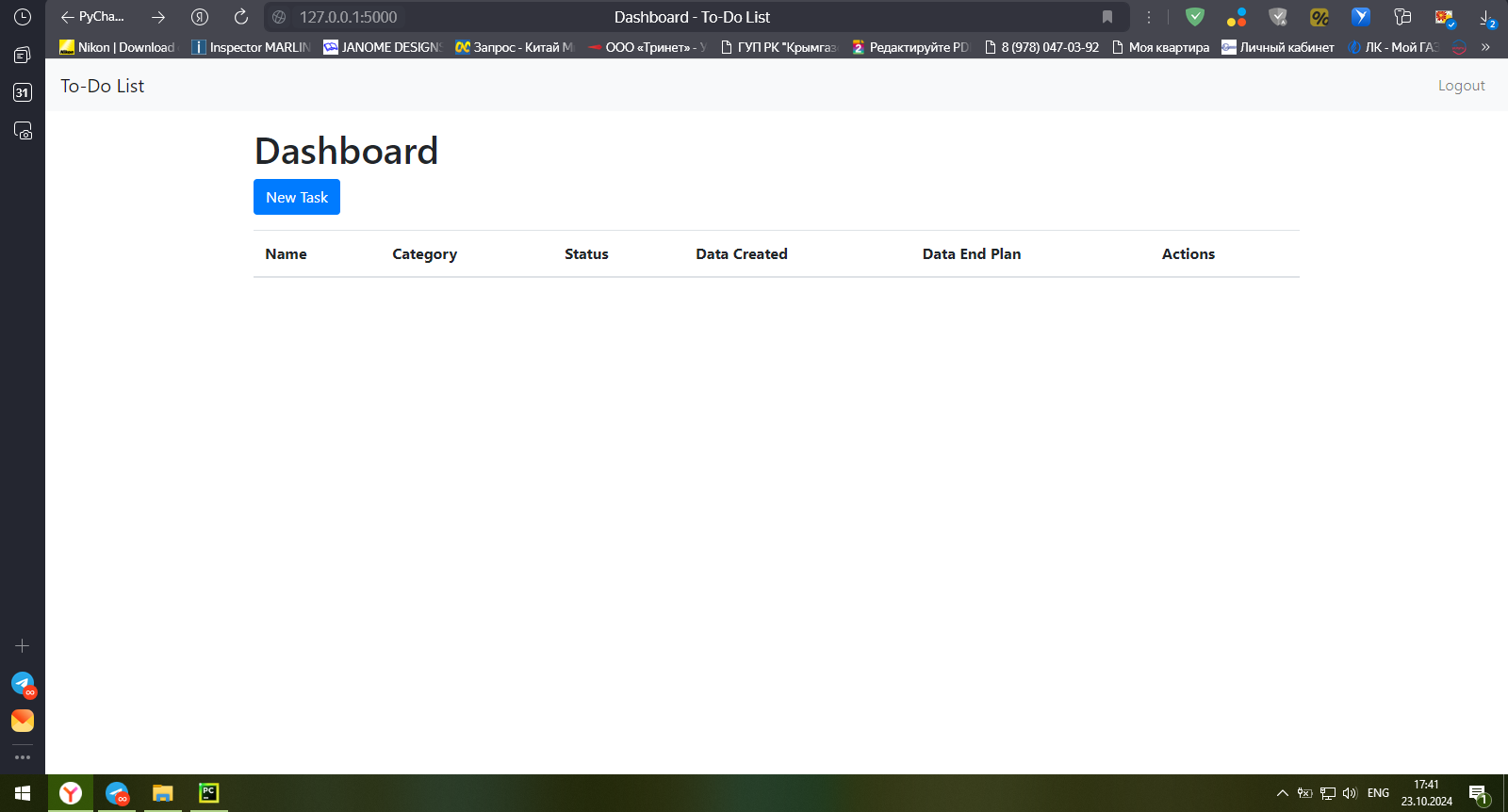
После запуска сайта переход на главную страницу осуществляется по адресу [http://127.0.1:5000](http://127.0.0.1:5000/)

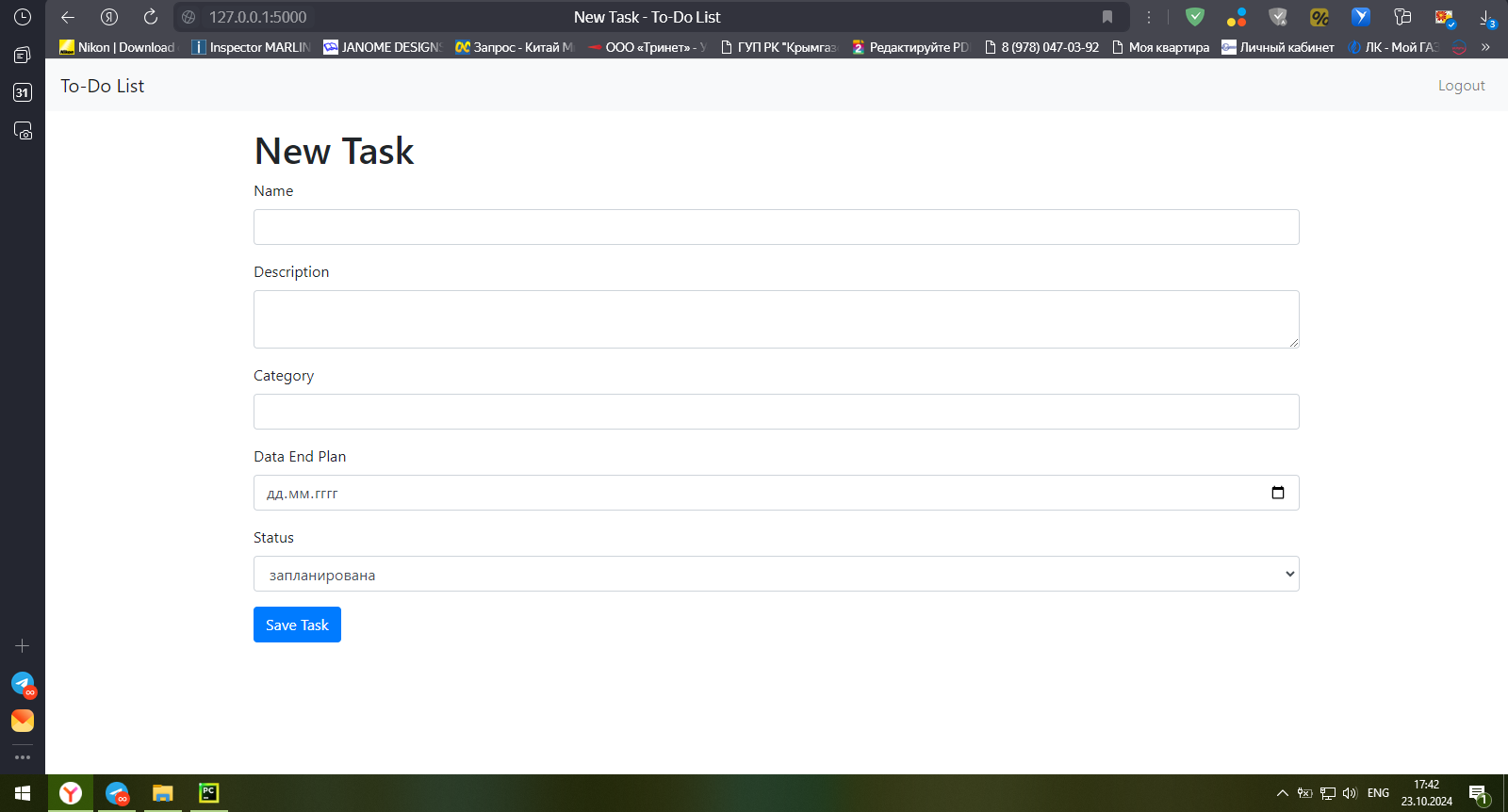
## 3.Структура сайта.

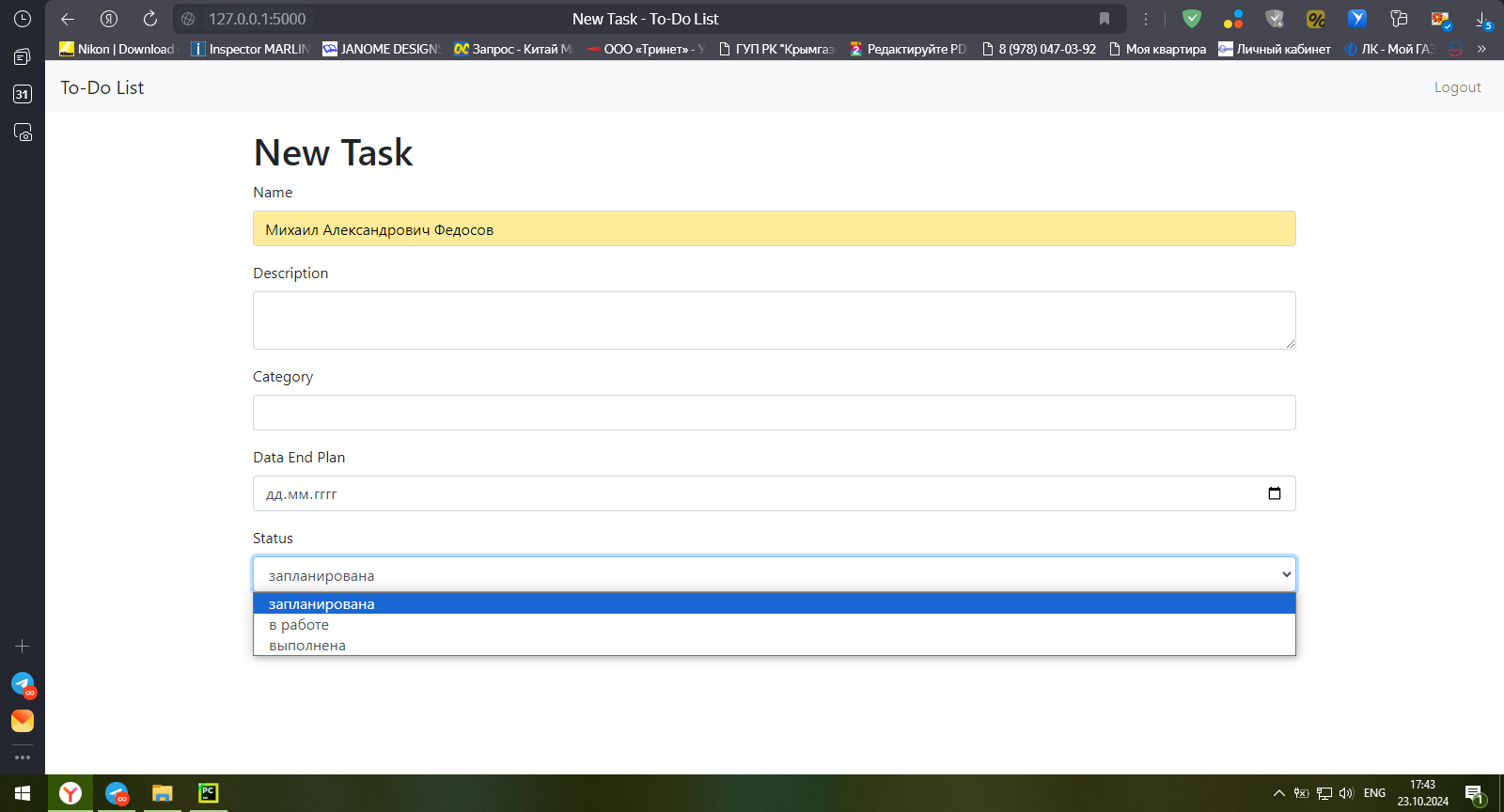
3.1.Главная страница, содержит форму авторизации и ссылку на регистрацию пользователя.

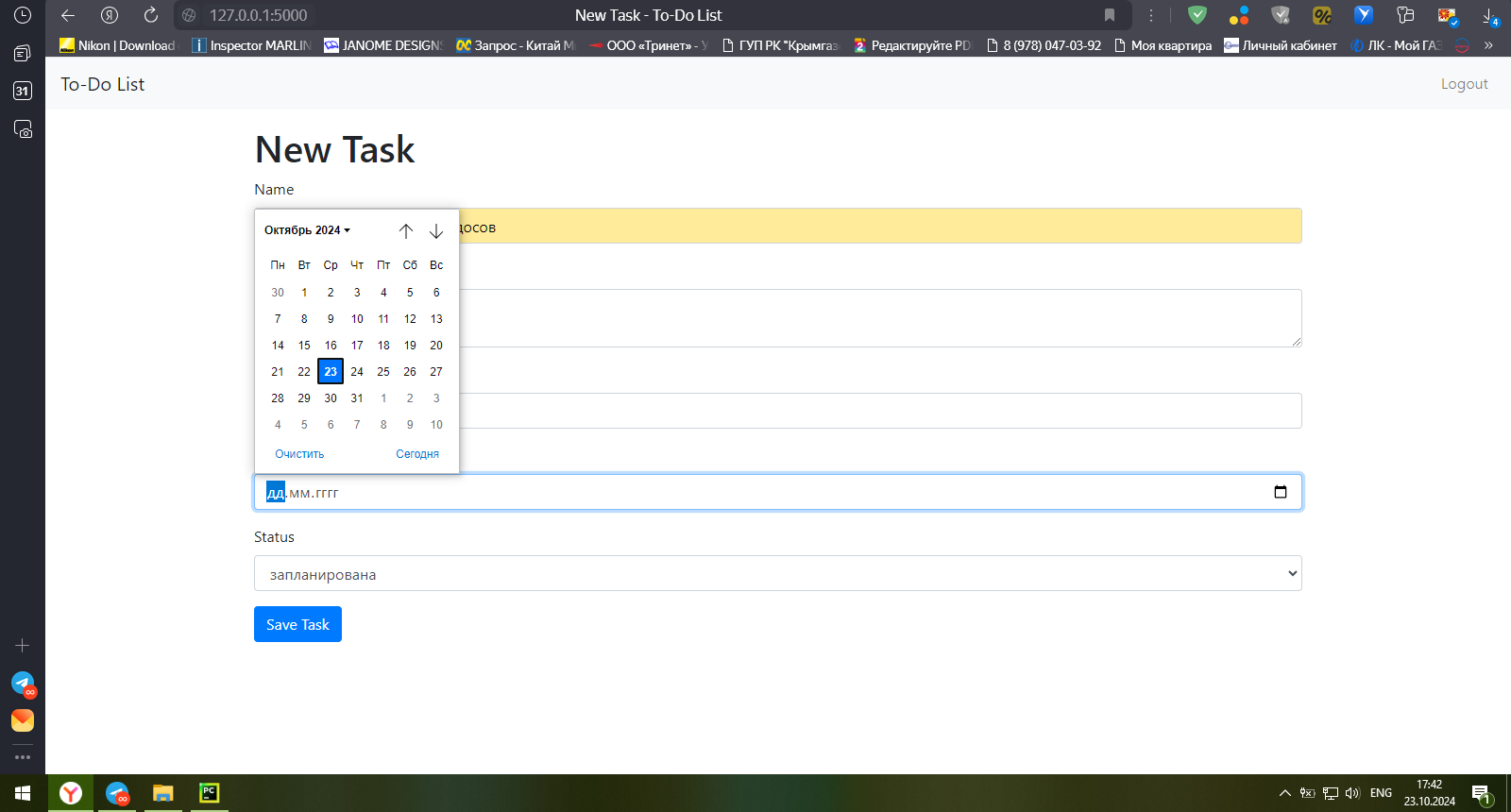


3.2. Нa дашборд попадаем после авторизации. Дaшборд - таблица со списком задач, блок фильтров (по дате, по названию, по статусу, по приоритету).

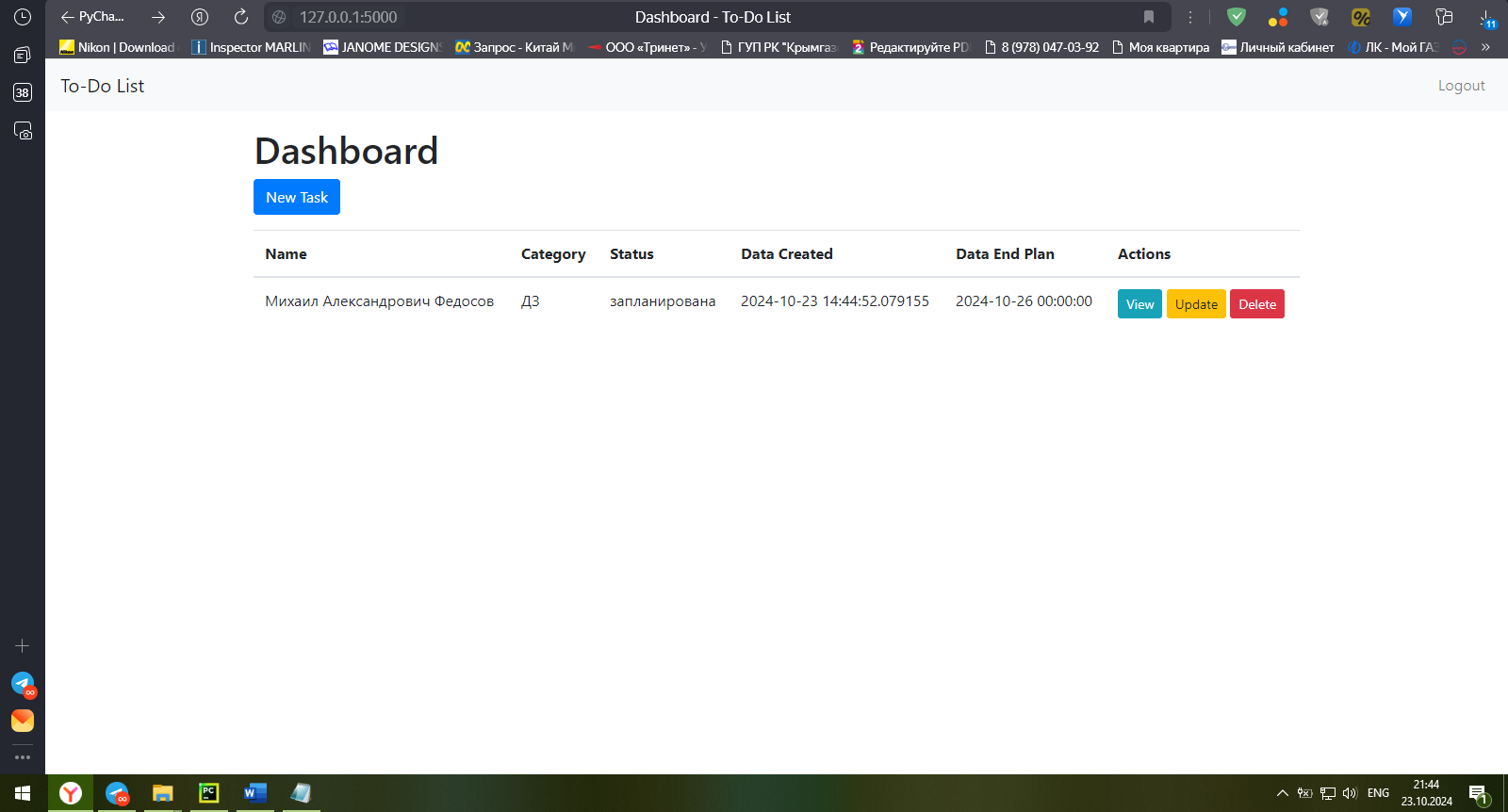




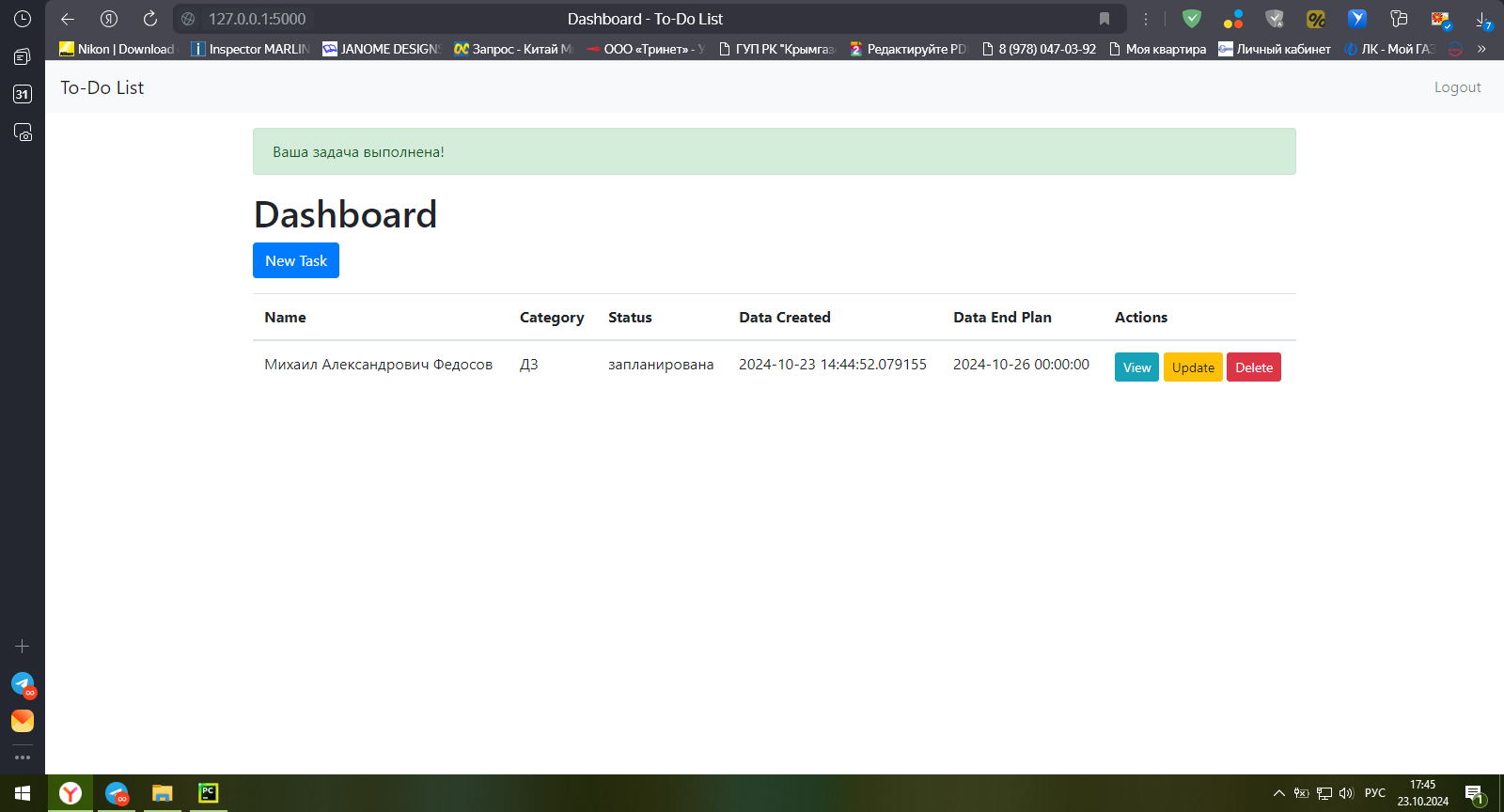




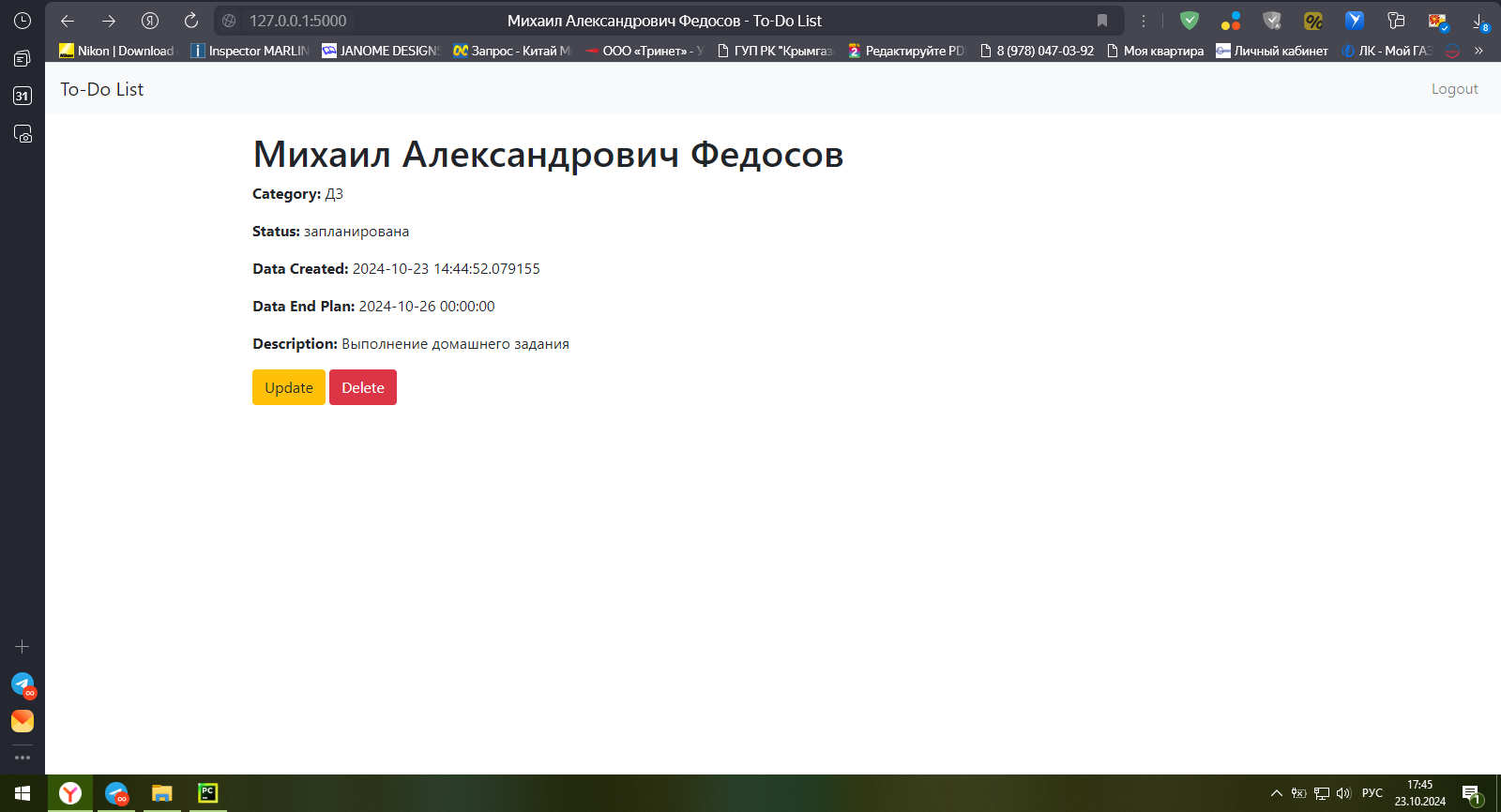
3.3 На дашборде есть кнопки Дoбавления, Редактирования, удаления и фильтра задач.



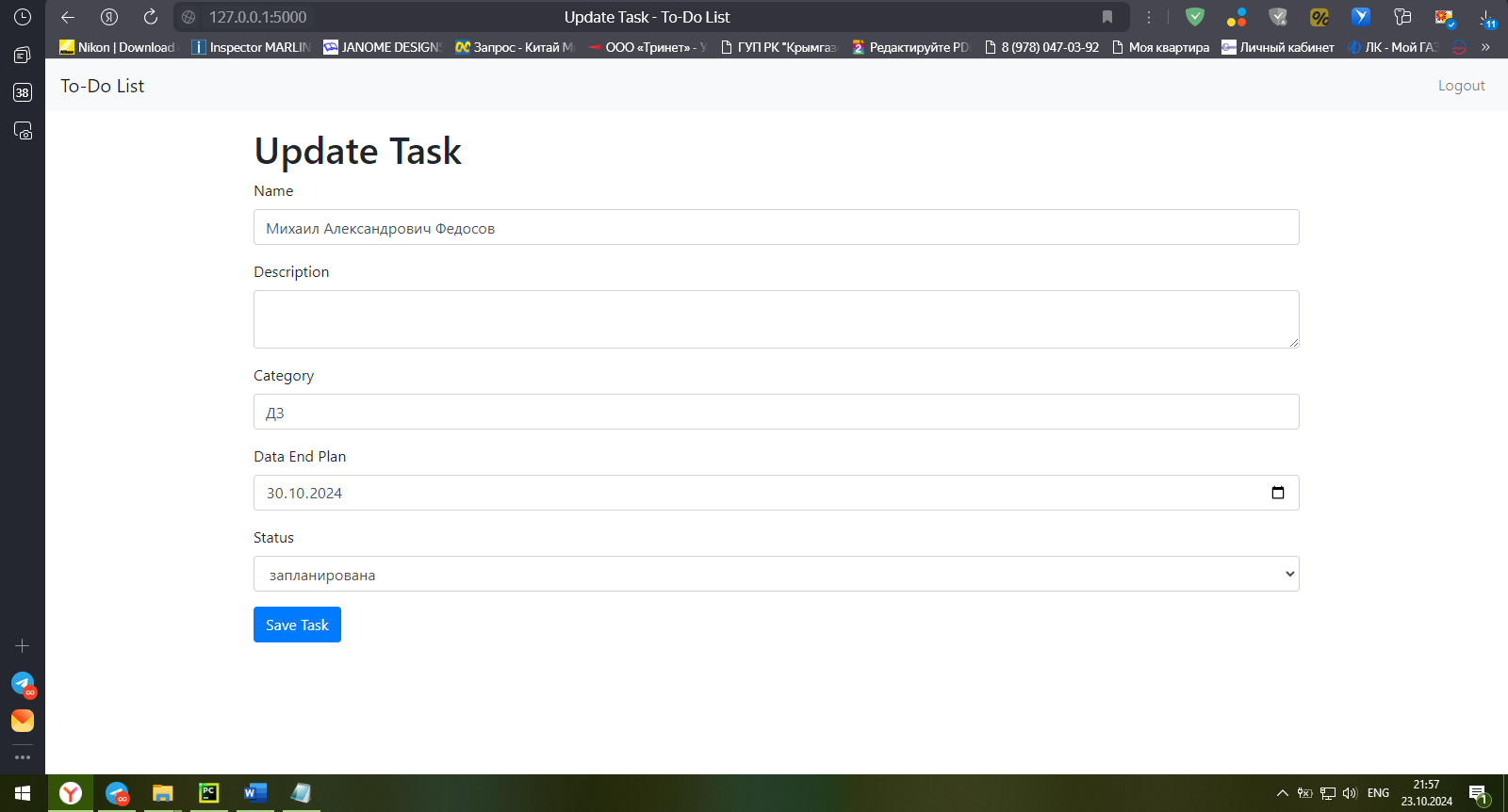
3.4 Запись сформированной задачи



3.5 Просмотр задачи



3.6 При нажатии на кнопку Редактирования открывается форма редактирования.



Личная оценка по 5-ти бальной шкале:

Скорость разработки: 4 Удобство добавления данных: 2 Скорость работы приложения: 3

Общие впечатления: удобно работать, импорты не перегружены объектами. Приятно видеть сразу декоратор и обработчик в одном файле. Интуитивно легко понять назначение объектов фреймворка по их названию.

**Fastapi**

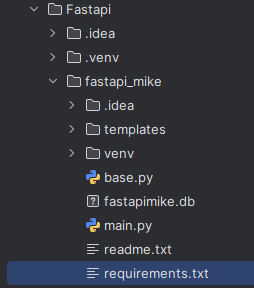
FastAPI — это также микровеб-фреймворк Starlette-ASGI (Asynchronous Server Gateway Interface) с открытым исходным кодом, написанный на Python, который используется для создания RESTful API.

Требования к функционалу веб-приложения. Список задач (To-Do List).

1. Регистрация и авторизация пользователей с сохранением в базе данных.
2. После авторизации пользователь попадает в форму Дашборд.
3. Требования к web-приложению Список задач (To-Do List).

* Регистрация и аутентификация пользователей
* Добавление, редактирование и удаление задач
* Фильтровать задачи по дате, названию, статусу и приоритету
* Статистика задач на панели управления
* Сохранение пользователей и задач в базе данных.

Структура директории Fstapi\_mike



## 1. Введение

Данное руководство предназначено для пользователей сайта База данных учета и исполнения входящей переписке организации, учета задач в учреждении разработанного на фреймворке FastAPI в рамках дипломной работы и показывает порядок работы с web-приложением.

## 2. Запуск сайта

### 2.1. Настройка окружения

Перед первым запуском необходимо создать виртуальное окружение Python. Для этого выполняем следующую команду в командной строке:

python -m venv venv

- следующей командой устанавливаем пакеты:

pip install -r requirements.txt

Дальнейший запуск сайта необходимо выполнять только работая в данном виртуальном окружении.

### 2.2. Запуск сайта

Для запуска сайта необходимо перейти в корневой каталог сайта Flask и в командной строке запустить команду :

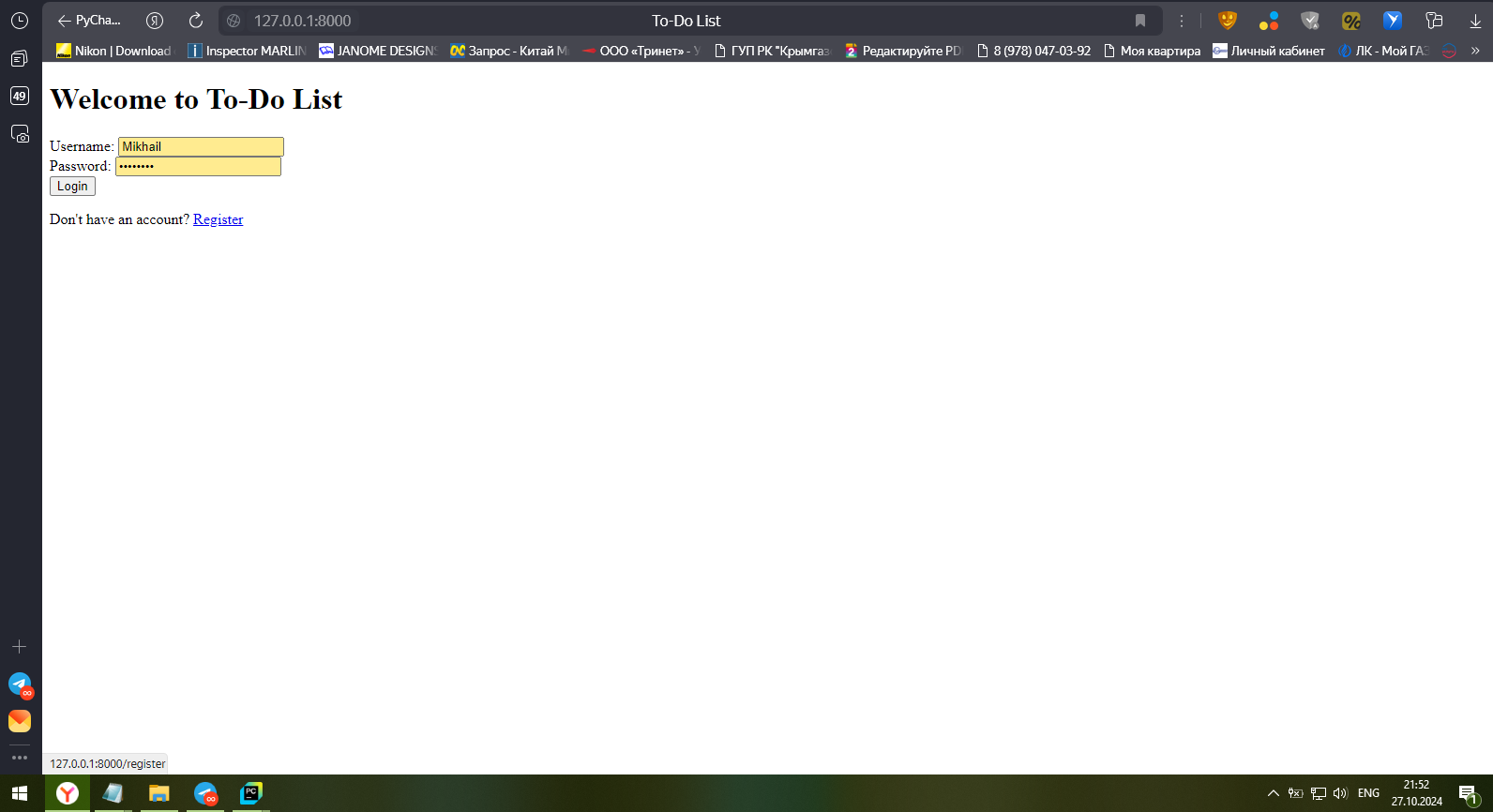
cd .\Fastapi\fastapi\_mike\

uvicorn main:app --reload

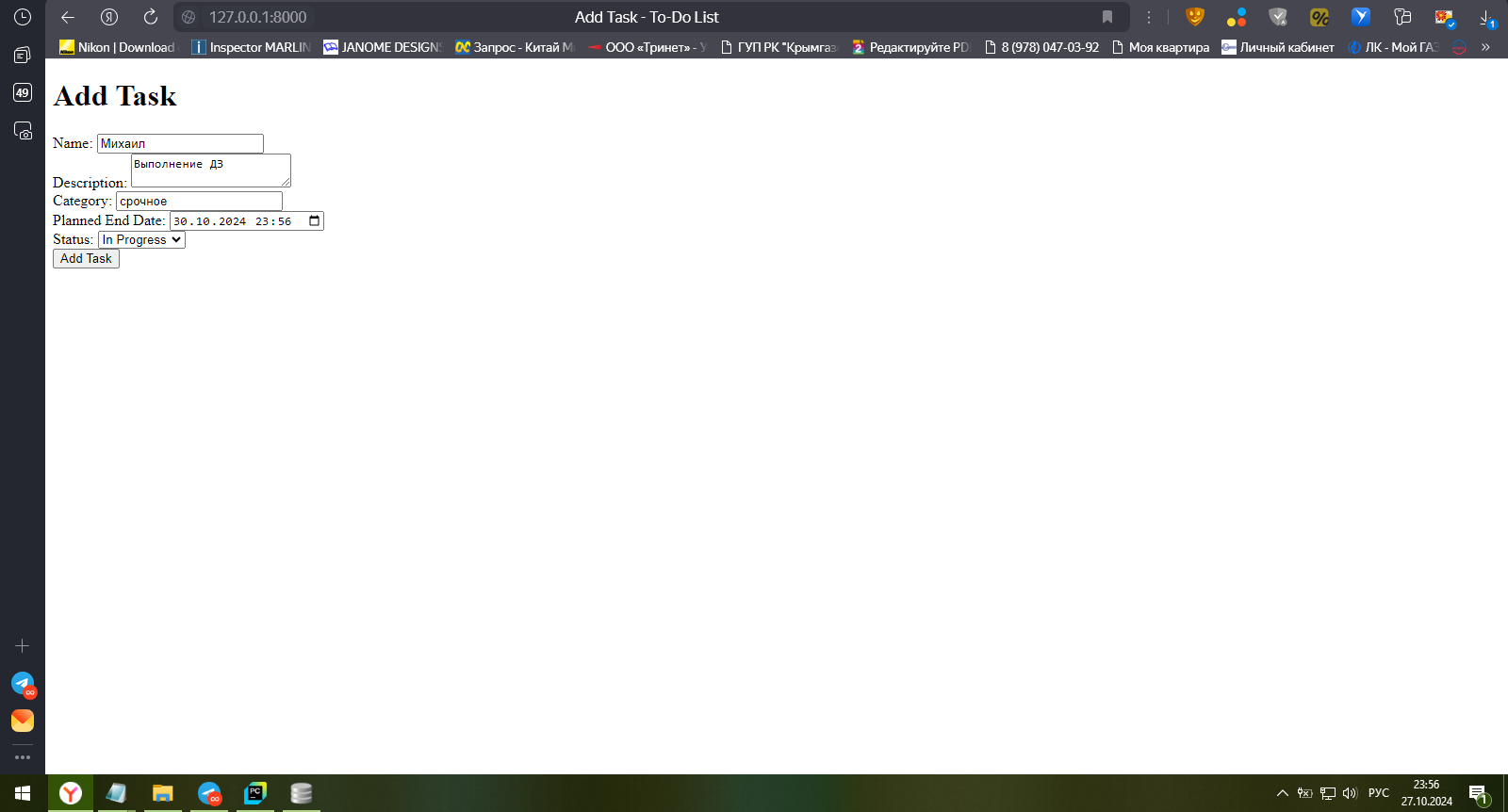
После запуска сайта переход на главную страницу осуществляется по адресу [http://127.0.1:8000](http://127.0.0.1:8000/)

## 3. Структура сайта.

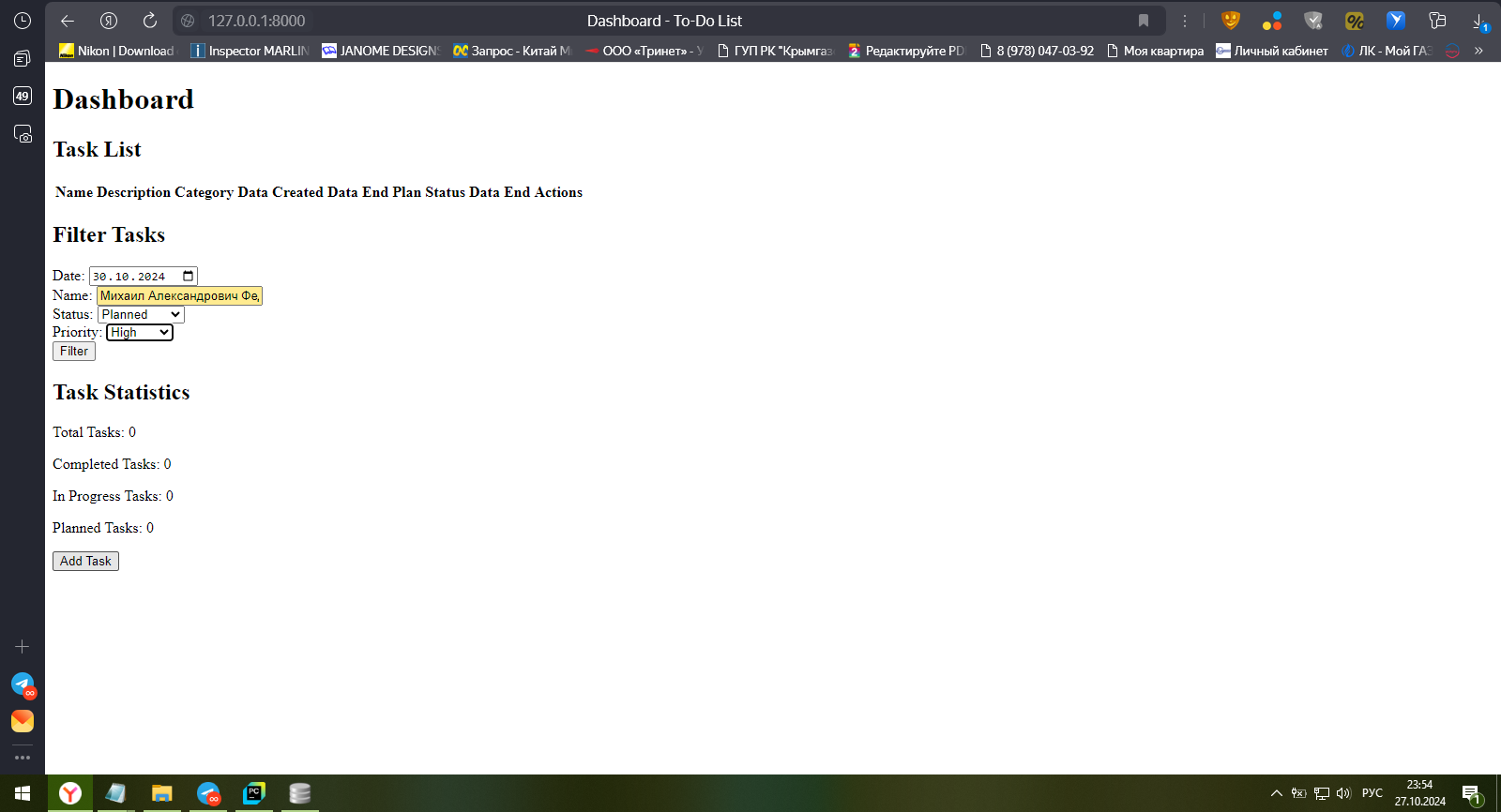
3.1.Главная страница дашборда To-Do List, содержит форму авторизации и ссылку на регистрацию пользователя.



3.2. После авторизации переходим на страницу добавления задачи, а также блоку фильтров (по дате, по названию, по статусу, по приоритету).

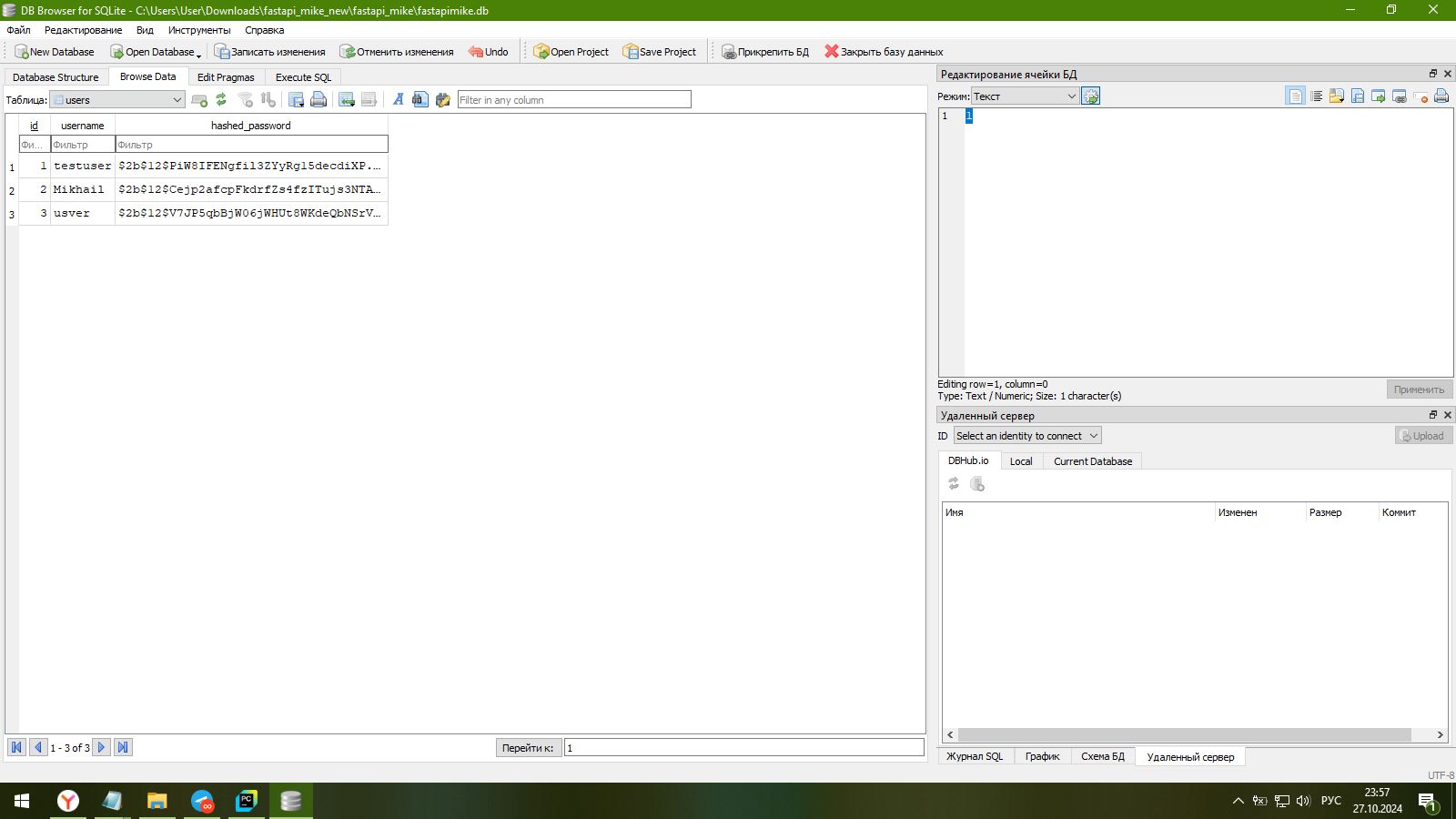


3.3 Создаем лист задачи

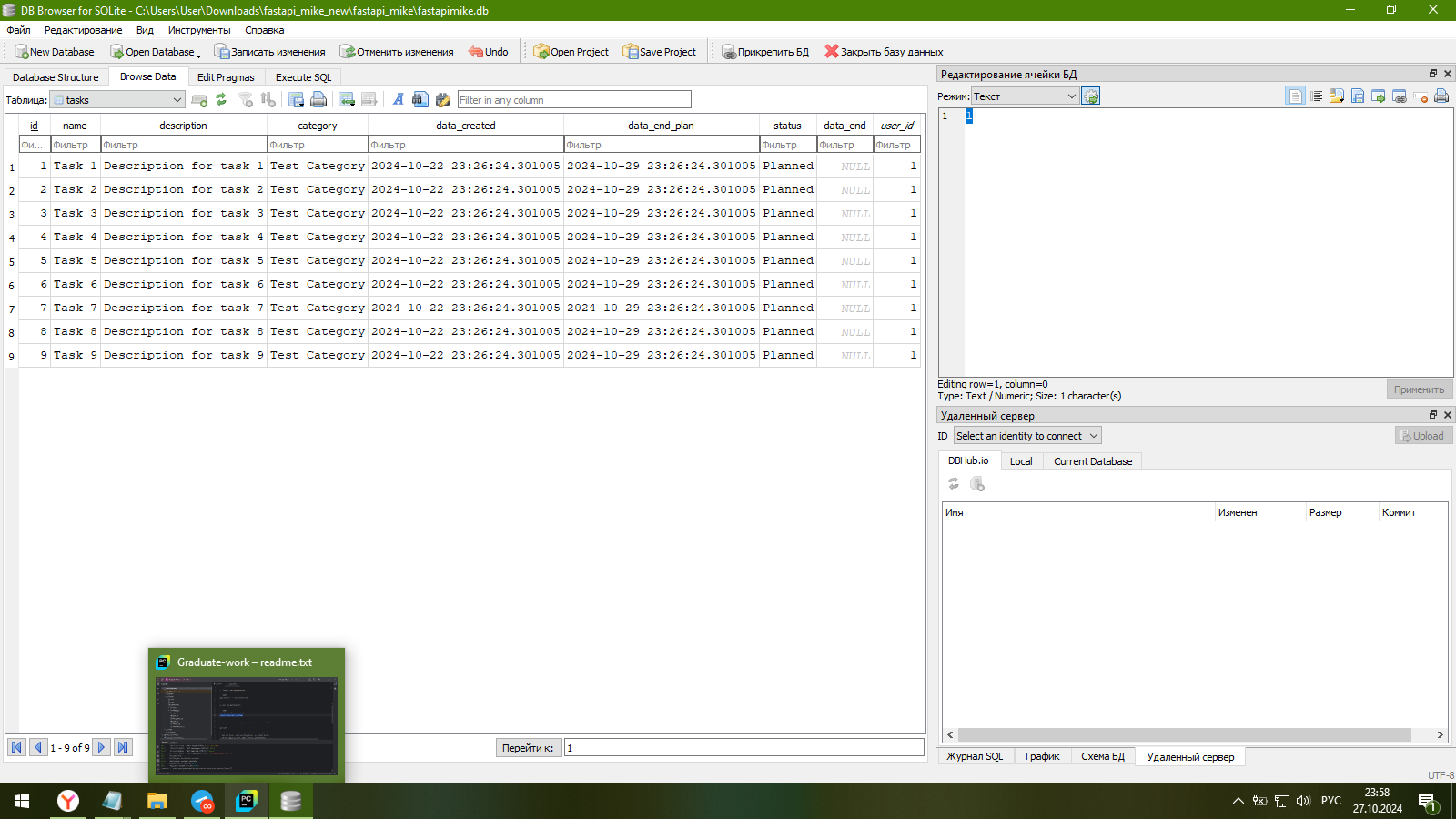


3.4 Проверяем базу данных

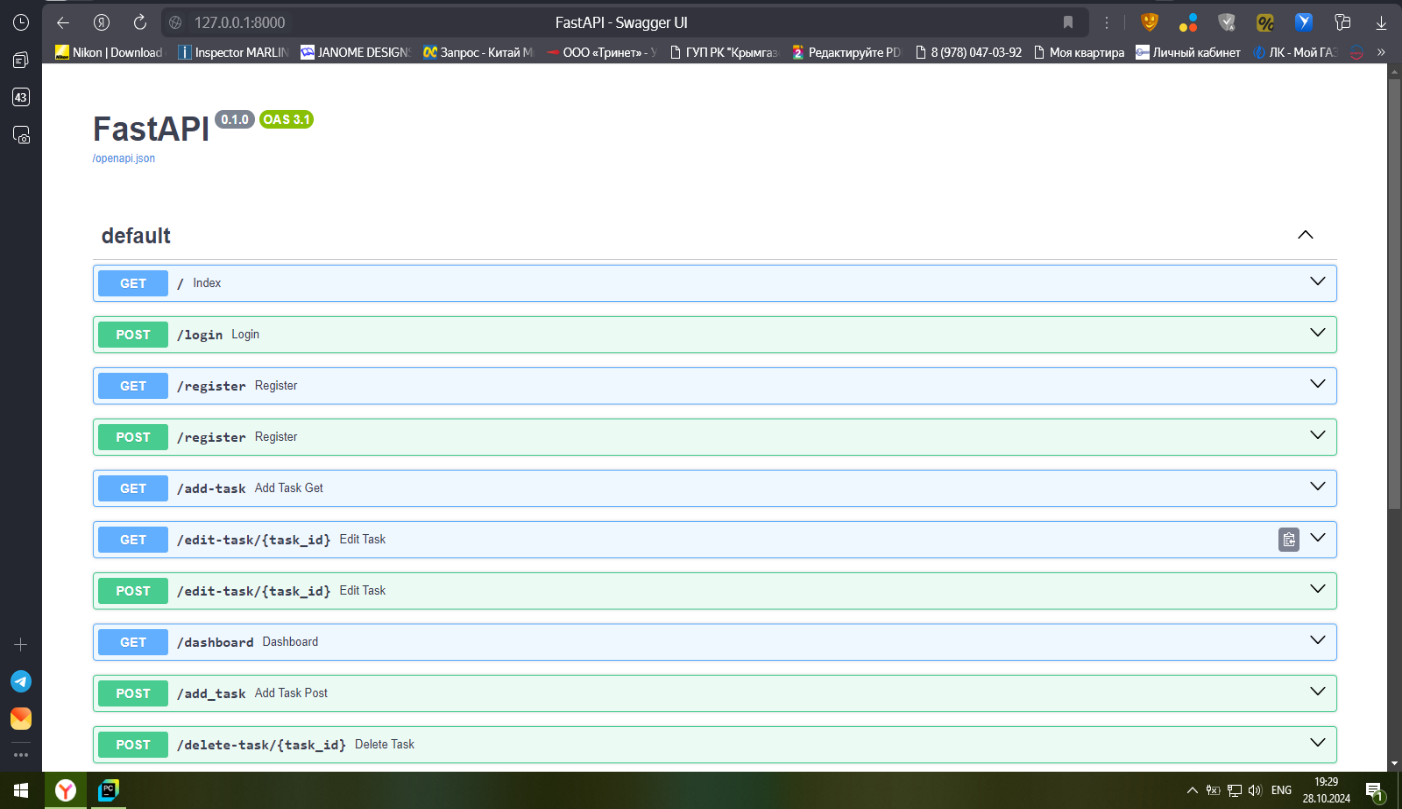
3.4.1 Пользователи

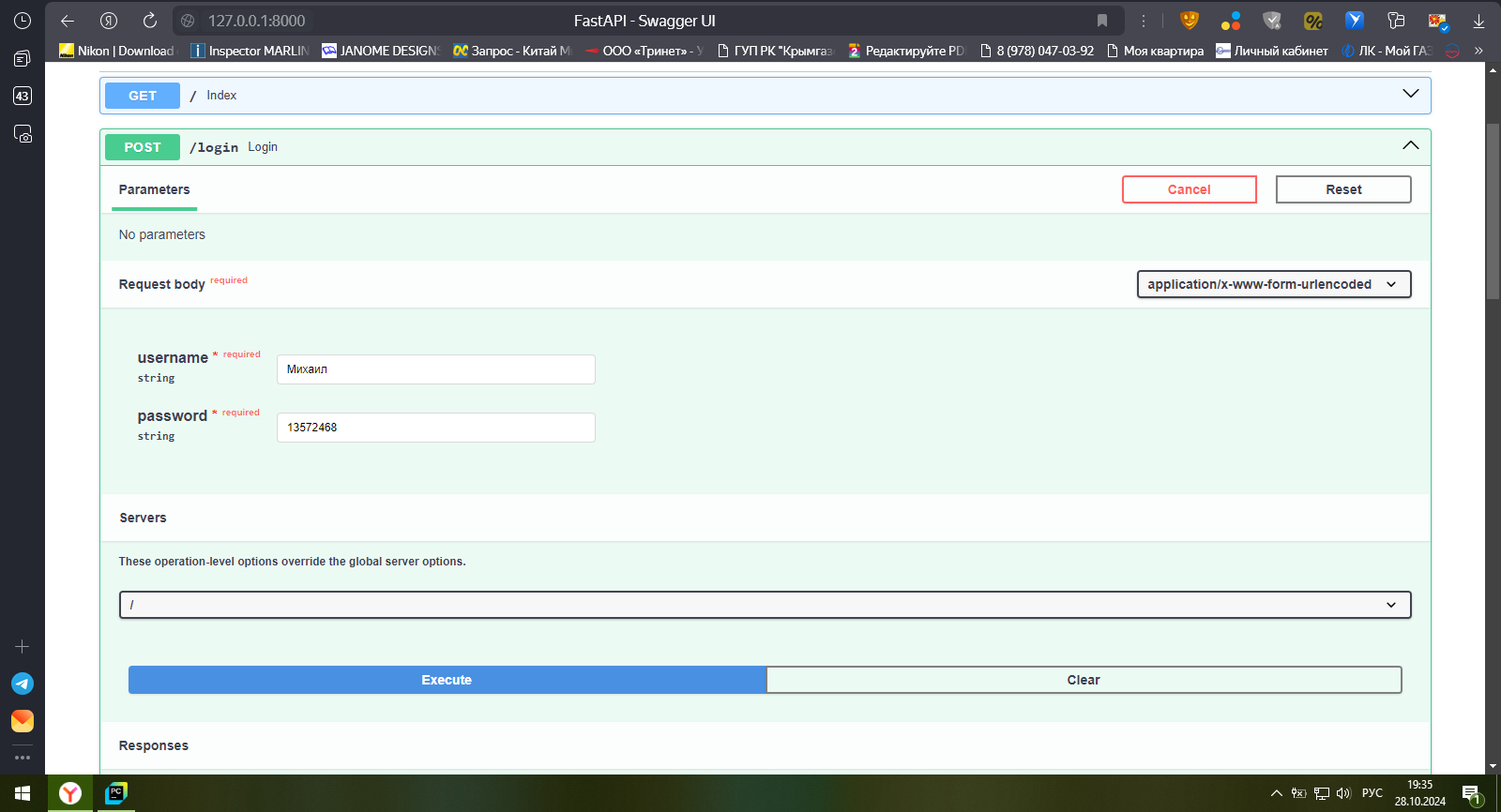


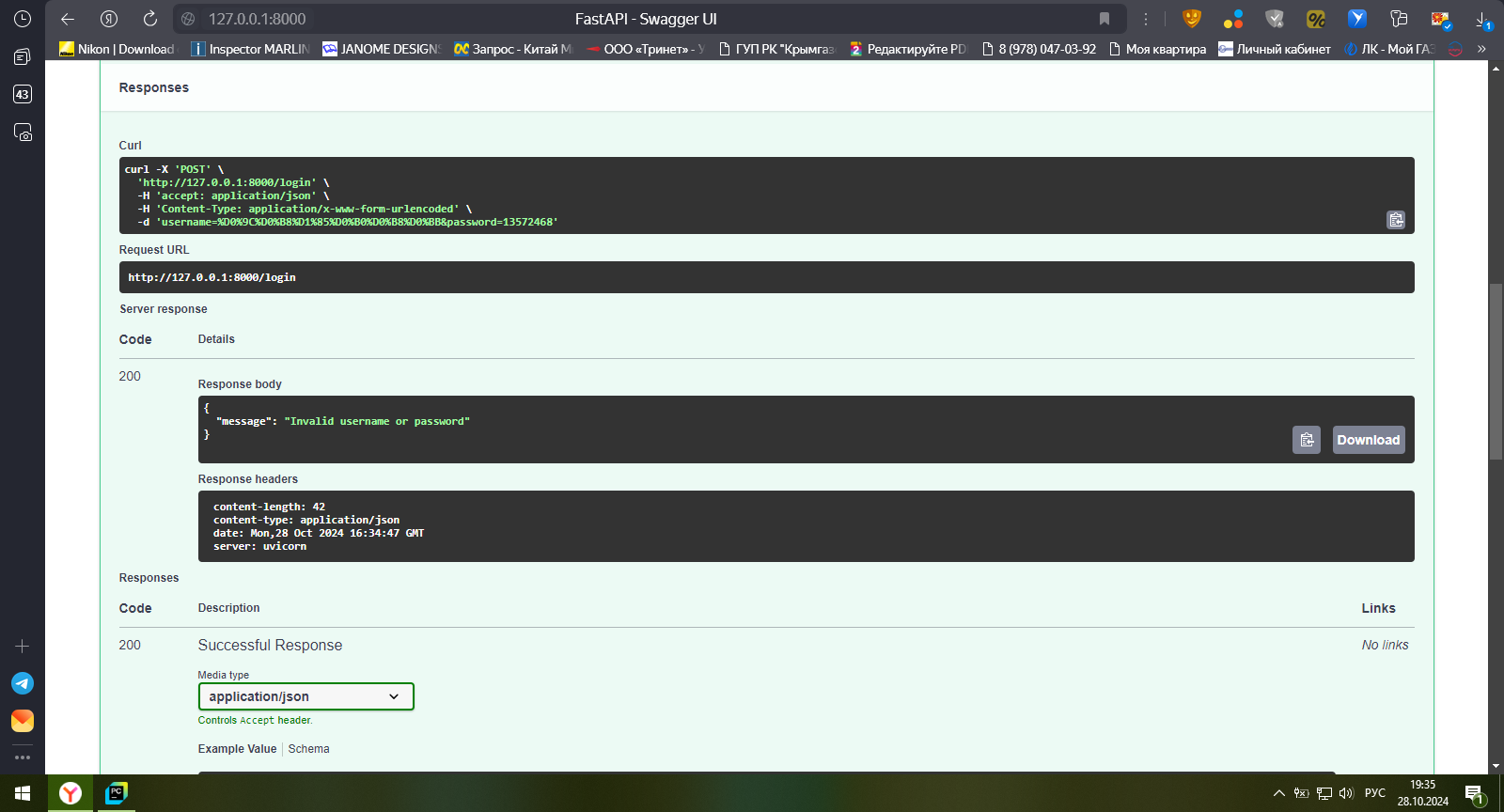
3.4.2 База данных задач



Набираем в адресной строке браузера <http://127.0.0.1:8000/docs/> и переходим о графическую оболочку FastAPI, которая значительно упрощает работу с базой данных, регистрации пользователей и вводимой информацией.







Личные оценки работы приложения на Фреймворке Fastapi по 5-ти бальной шкале:

Скорость : 4 Удобство добавления данных: 4 Скорость работы приложения: 5

Общие впечатления: есть как свои плюсы, так и минусы. Плюсом является скорость работы, потому, что фреймворк является полностью асинхронным, но программировать не очень удобно из-за большого количества объектов и классов фреймворка и перегрузки количества импортов в написанном коде. Также к плюсам можно отнести то, что декоратор и обработчик находятся рядом в одном файле. База данный строится на основе моделей SQLAlchemy, что с одной стороны является плюсом, так как позволяет не привязываться к определенной СУБД, но с другой требует дополнительного изучения объектов и методов самой SQLAlchemy которые часто не являются интуитивно понятными. Таким образом, для эффективной работы с данным фреймворком нужен определенный опыт и знания.

**Django**

Django — это полнофункциональный веб-фреймворк с открытым исходным кодом, написанный на Python, который следует архитектурному шаблону MTV (модель-шаблон-представление) и принципу DRY.

Требования к функционалу веб-приложения.

1. Регистрация и авторизация пользователей с сохранением в базе данных.
2. После авторизации пользователь попадает в форму Дашбод.
3. Требования к web-приложению.
   * Добавление, редактирование и удаление задач.
   * Отметка задач как выполненных.
   * Фильтрация задач по статусу (все, активные, выполненные).
   * Возможность сортировки по дате создания или приоритету.
   * Возможность создания базы данных через "Admin"- панель
   * Проверка внесенных данных в БД и их редактирование

## 1. Введение

Данное руководство предназначено для пользователей сайта База данных учета и исполнения входящей переписке организации, учета задач в учреждении разработанного на фреймворке **Django** в рамках дипломной работы и показывает порядок работы с web-приложением.

## 2. Запуск сайта

### 2.1. Настройка окружения

Перед первым запуском необходимо настроить виртуальное окружение Python. Для этого выполняем следующую команду в командной строке:

python -m venv venv

- следующей командой устанавливаем пакеты:

pip install -r requirements.txt

Дальнейший запуск сайта необходимо выполнять только работая в данном виртуальном окружении.

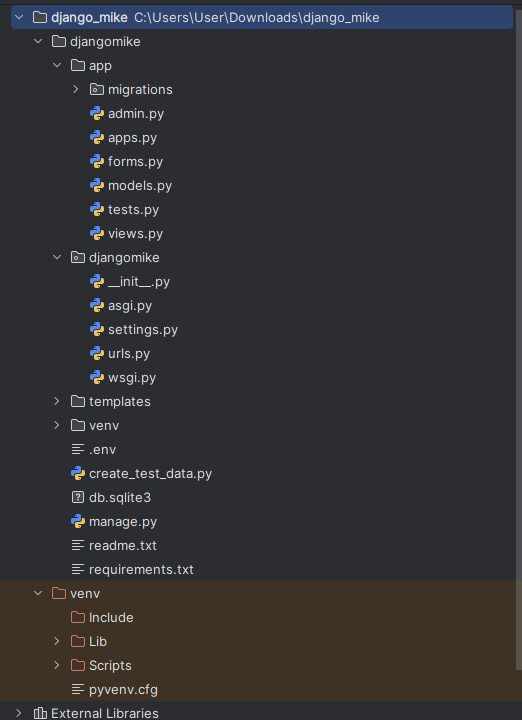
### 2.2. Запуск сайта

Для запуска сайта необходимо перейти в корневой каталог сайта **Django** и в командной строке запустить команду :

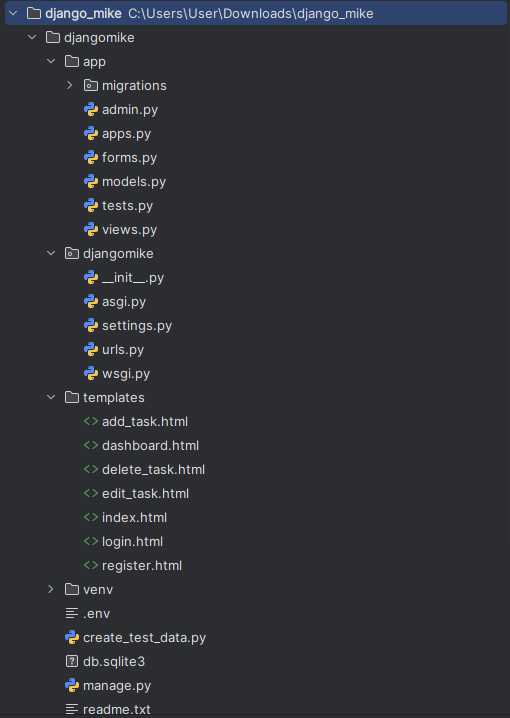
python manage.py runserver

После запуска сайта переход на главную страницу осуществляется по адресу [http://127.0.1:8000](http://127.0.0.1:8000/)

Структура директории django\_mike

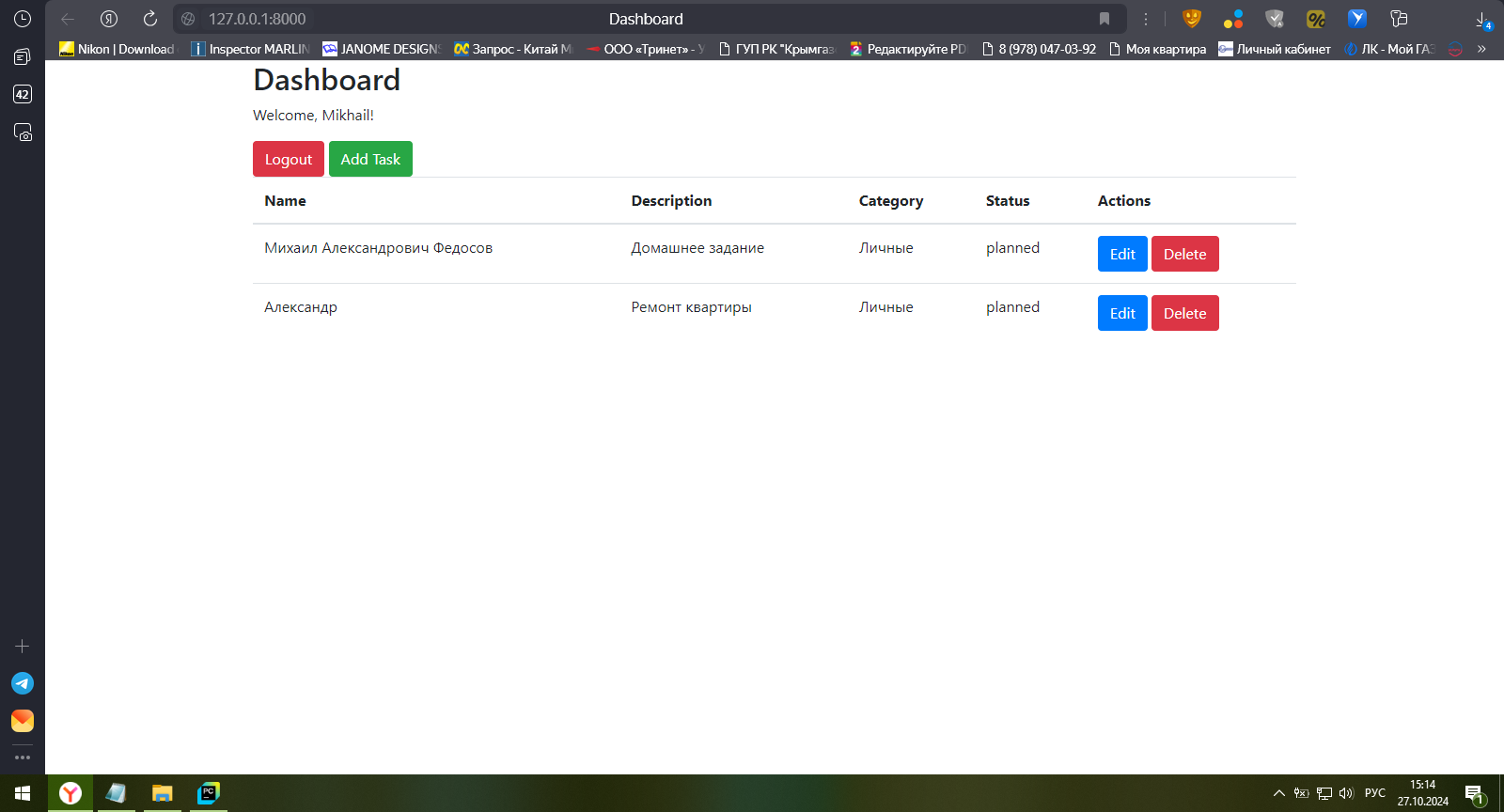


Структура директории «tempates»

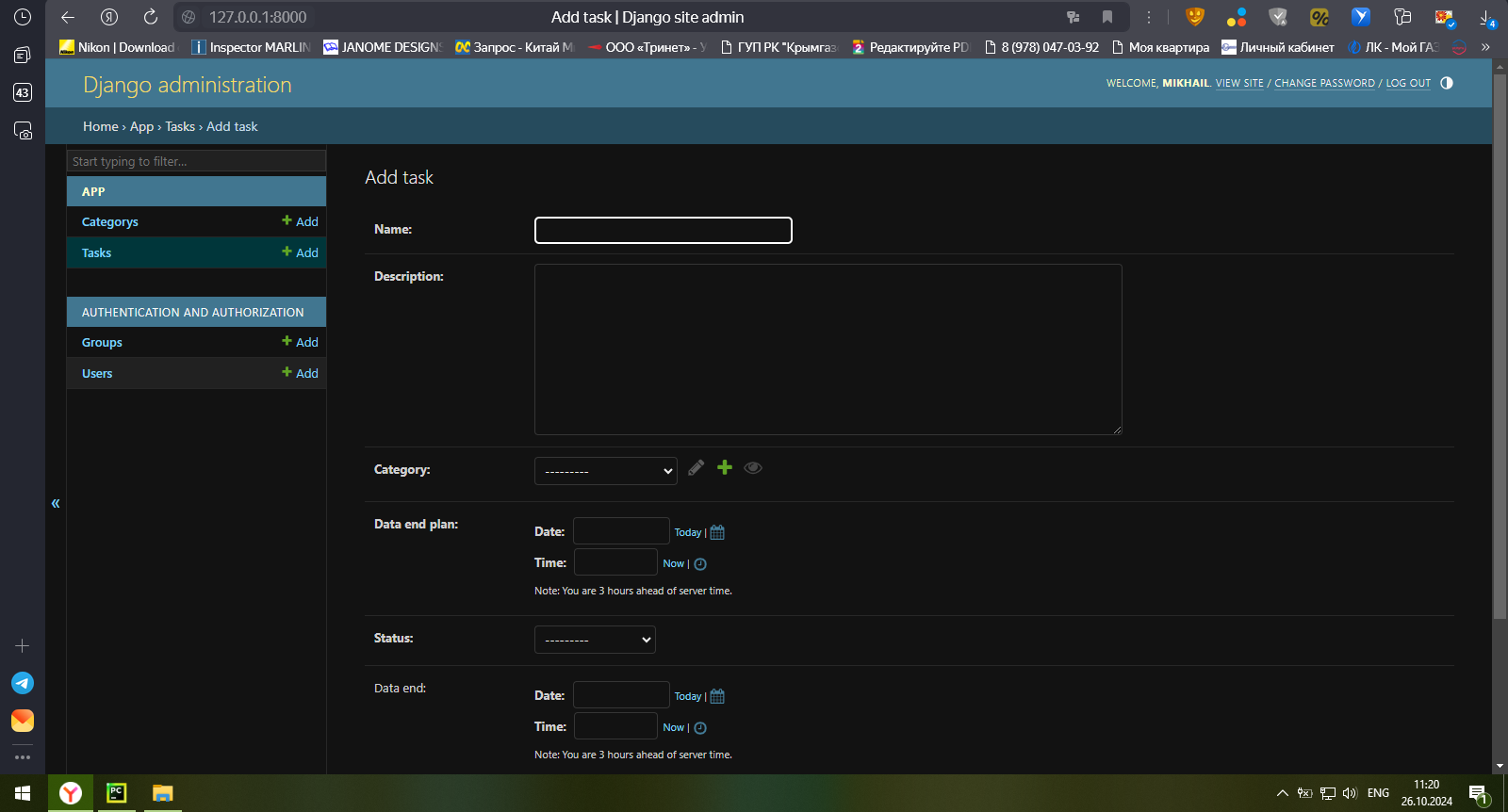


## 3. Структура сайта.

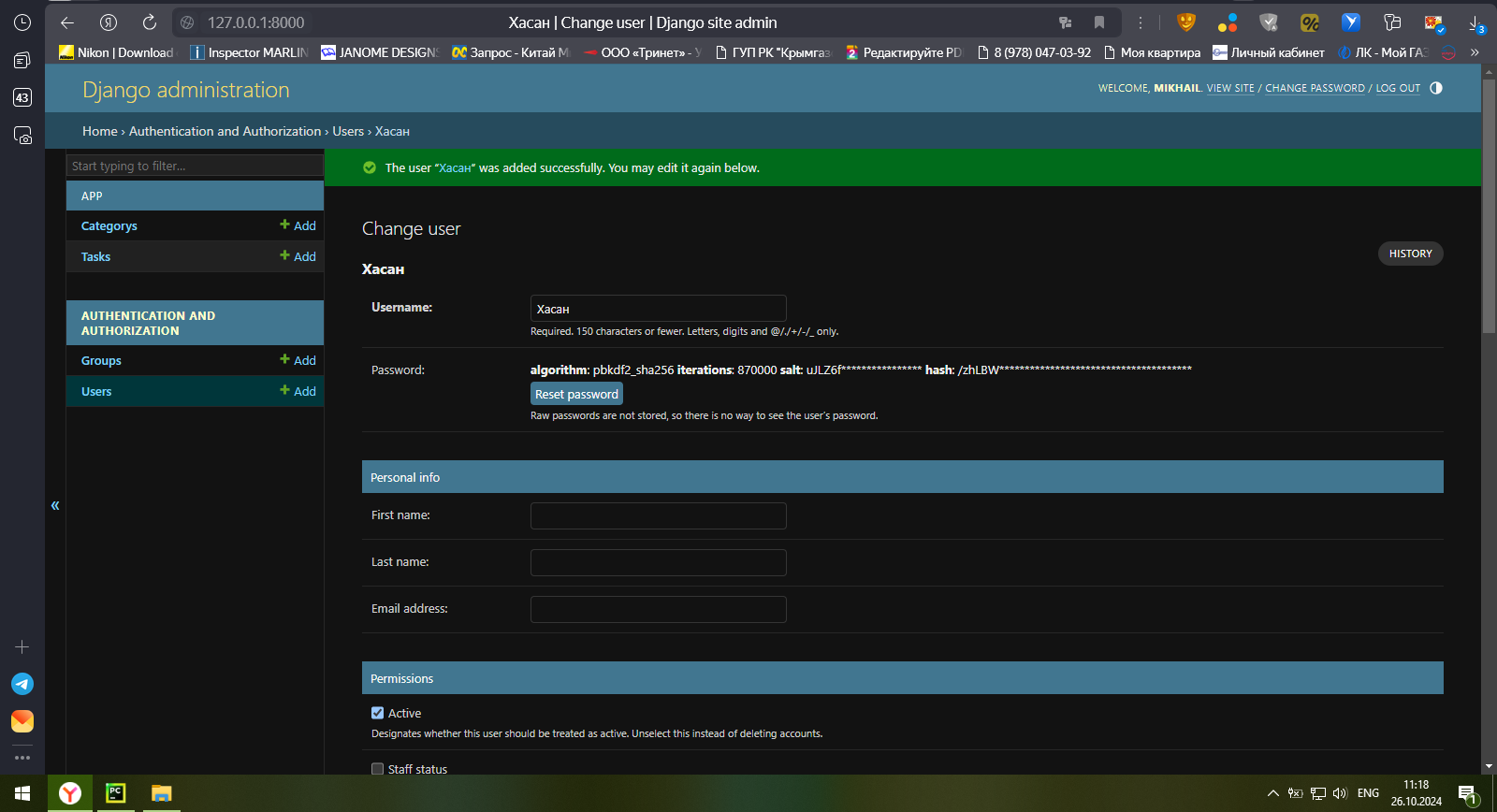
3.1. После запуска кода в командной строке (python manage.py runserver) переходим по адресу: <http://127.0.0.1:8000/dashboard/> (локально) где появляется возможность добавлять пользователя и задачи.



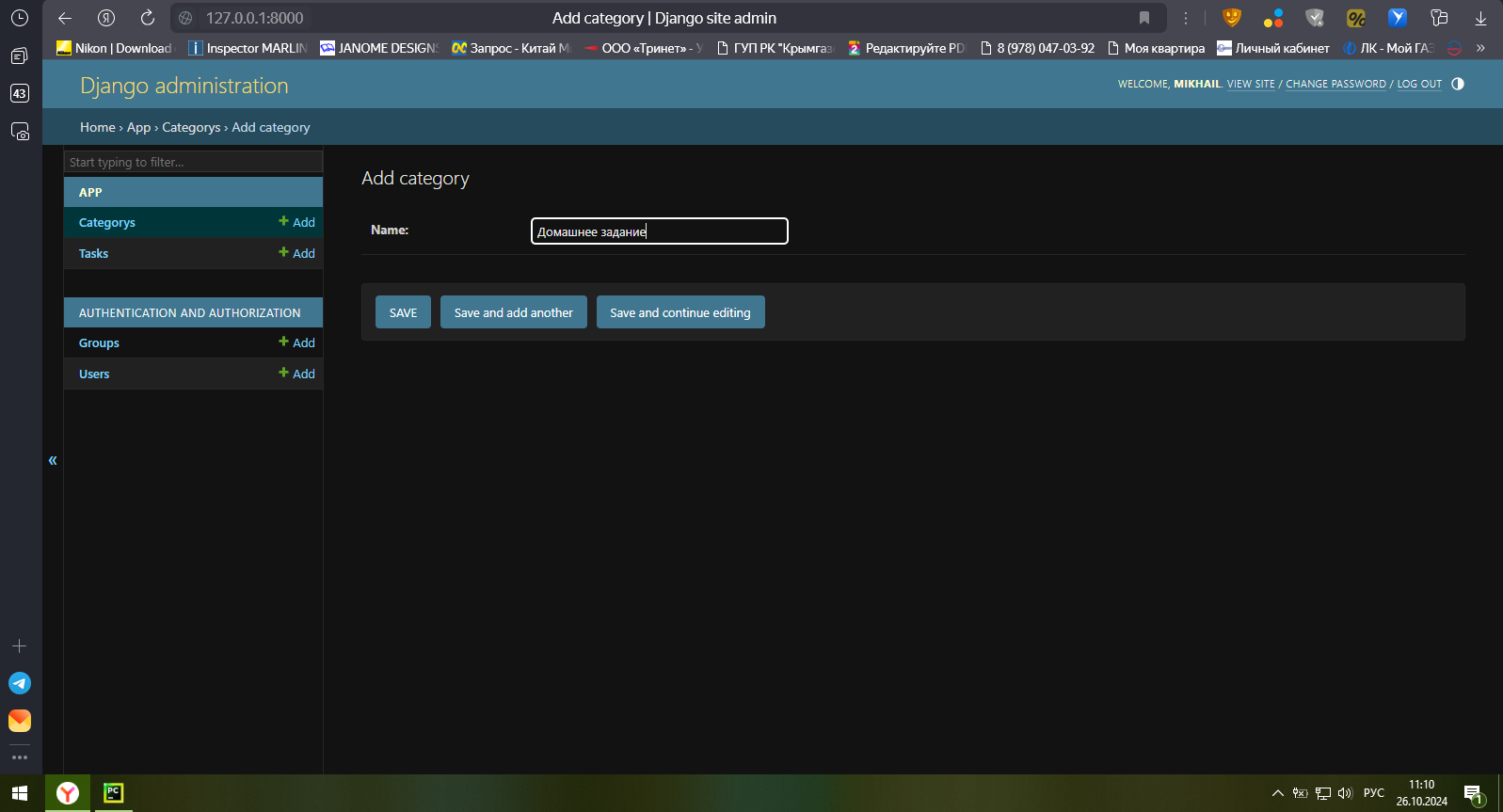
3.2. Набрав адрес на странице (<http://127.0.0.1:8000/admin/>) переходим в панель администрирования Django, где необходимо зарегистрироваться или войти под своей учетной записью и появляется возможность добавления одиночных пользователей, групп пользователей, категорий, задач:



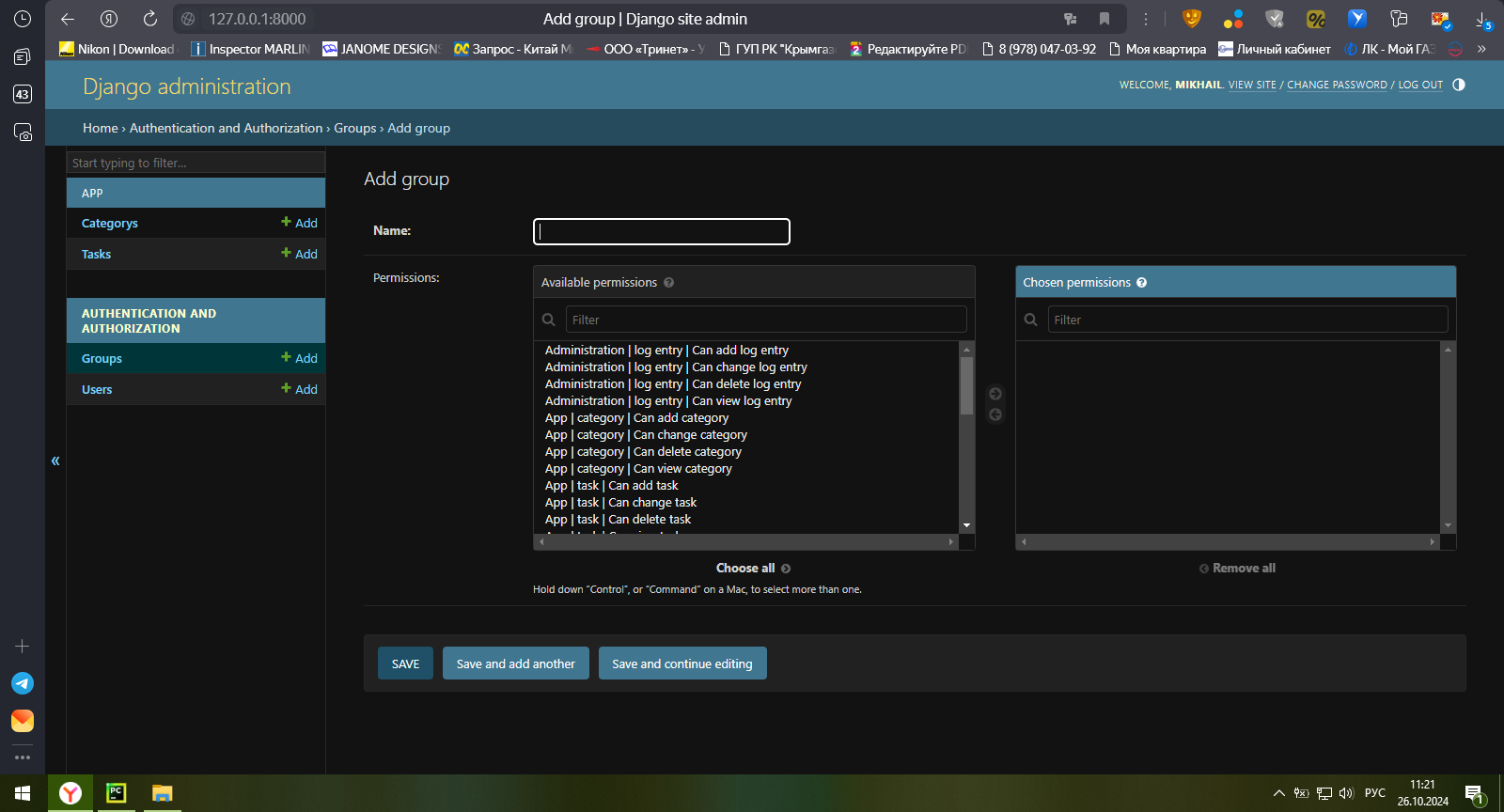
3.3. Заполнение полей пользователя:



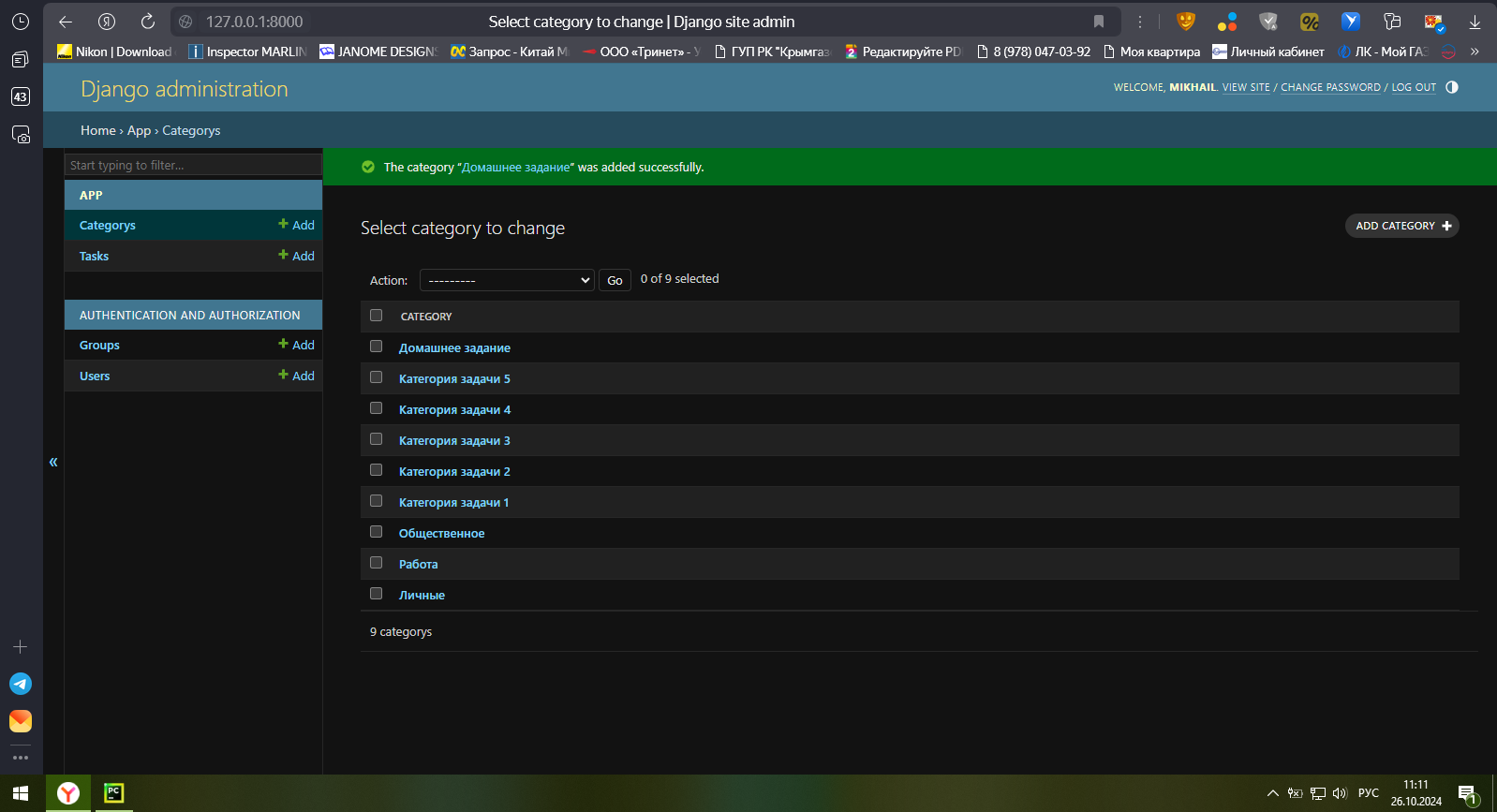
3.4 Создание категории:



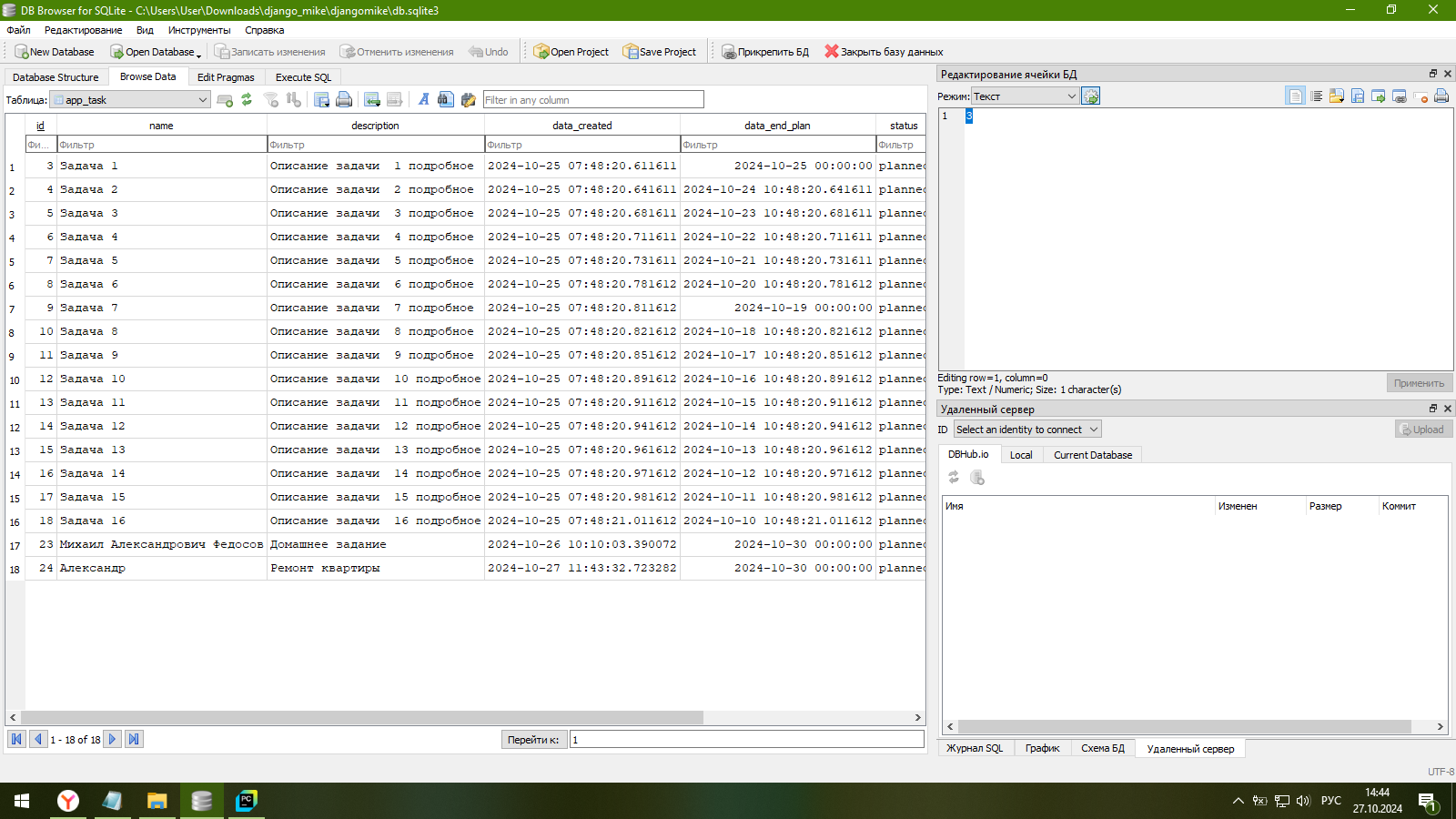
3.5 Создание группы пользователей:



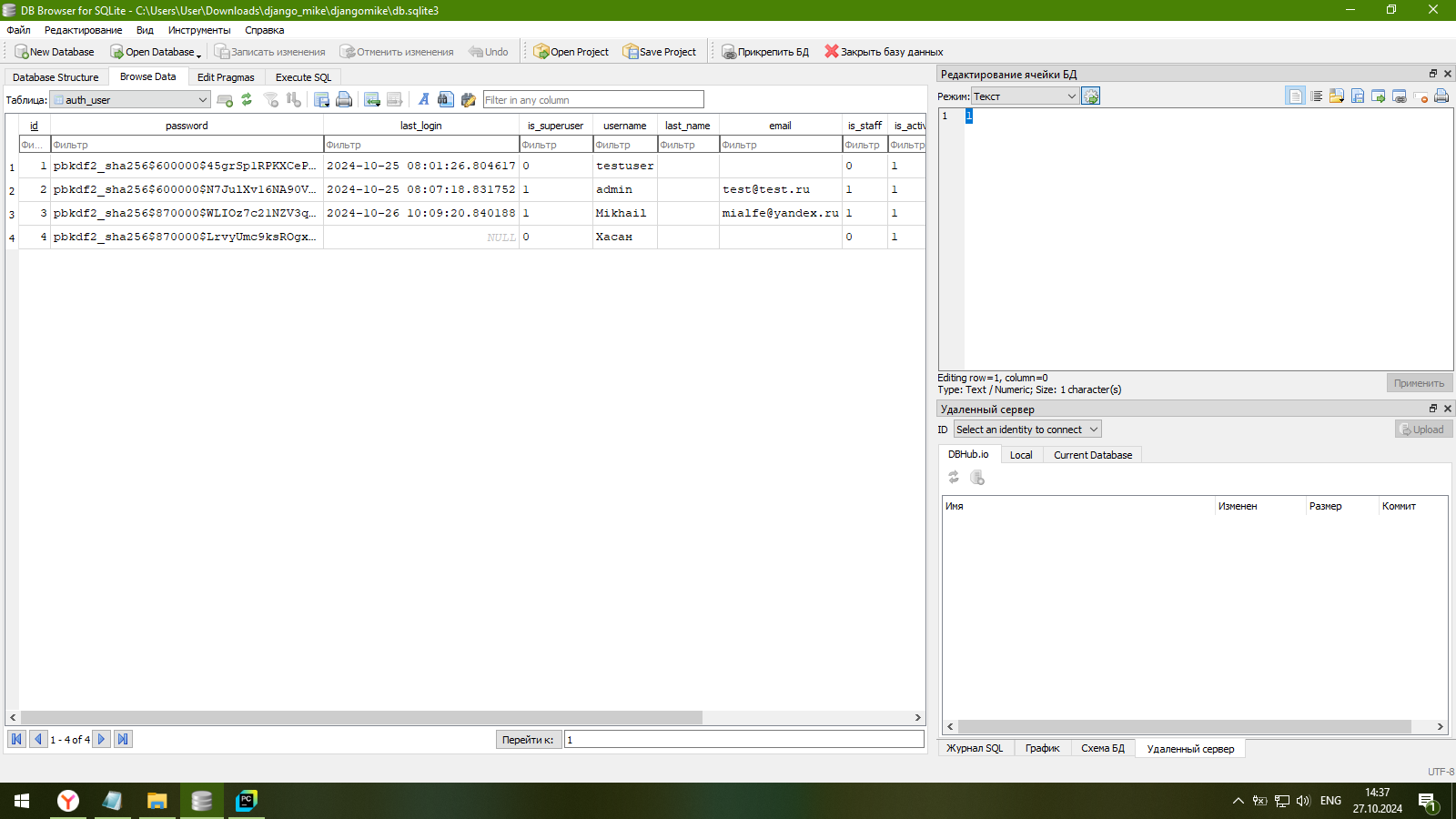
3.6 Созданные ранее категории:



3.7 Установив отдельное стороннее приложение DB Browser, можем осуществлять просмотр и редактирование информации в БД:



3.8 Пользователи загруженные в БД:



В качестве БД выбран sqlite. В качестве ORM использовалась Django ORM, а в качестве системы по управлению миграциями также используется Django. Никаких дополнительных инструментов для поддержки миграций устанавливать не требуется.

Все основные функции, включая создание проекта, приложения, запуск или работу с миграциями выполняются через системный файл manage.py. Фреймворк требует от разработчика соблюдение определенной дисциплины, поэтому весь написанный код должен находиться в определенных файлах, например модели приложений или отображения (views) должны находиться в файлах models.py и views.py в корне каталога приложения. А файл с маршрутами urls.py должен находиться в корне каталога проекта.

Общие впечатления: основным минусом работы с Django на мой взгляд не самая быстрая скорость работы (FastAPI работает значительно быстрее), но это не перекрывает плюсы фреймворка: наличие четкой структуры проекта. Присутствие Административной панели "из коробки" позволяет удобно манипулировать данными, включая полный CRUD. К минусам возможно отнести стандартный шаблонный язык Django (Django Template Language), который не до конца совместим с Jinja2, хотя и имеет такой же синтаксис. Но если быть до конца честным, Django позволяет настроить Jinja2, но это требует дополнительных манипуляций со стороны разработчика. Другим минусом считаю интуитивно непонятный перевод Django из режима отладки в 'боевой' режим, т.к. это требует изменения не только DEBUG, но и других параметров (ALLOWED\_HOSTS, настройки статики и т.д.). Но это мое чисто субъективное мнение.

**Личное мнение по работе приложений**:

Каждый фреймворк имеет свою нишу для применения, несмотря на свои плюсы и минусы. Таким образом, можно выделить основные показатели:

**Flask:**

Плюсы: Легковесный и гибкий, легко осваивается, большое сообщество и много расширений. Минусы: Ограниченный функционал "из коробки", для сложных проектов требует больше настроек. Когда использовать: Идеален для небольших проектов, прототипов, микросервисов или простых API.

**FastAPI:**

Плюсы: Высокая производительность, поддержка асинхронности, автоматическая генерация документации. Минусы: Новое сообщество, меньше готовых решений. Когда использовать: Отличен для высоконагруженных API и приложений с большим количеством запросов.

**Django:**

Плюсы: Полный набор инструментов для разработки веб-приложений, безопасность, быстрое прототипирование. Минусы: Тяжелый для небольших проектов, сложнее в освоении. Когда использовать: Подходит для крупных приложений с комплексной бизнес-логикой, таких как интернет-магазины и CRM-системы.

**Сравнительные таблица показатели фреймворков**

|  |  |  |  |
| --- | --- | --- | --- |
| **Параметры** | **Django** | **FastAPI** | Flask |
| **Тип** | Полноценный веб-фреймворк | Микро-веб-фреймворк | Микро-веб-фреймворк |
| **Вариант использования** | Возможность создания сложныех веб-приложений и API | Создание API и микросервисов | Идеально подходит для создания небольших веб-приложений и простых API. |
| **Производительность** | Быстро для создания больших веб-приложений | Очень быстро для создания API и микросервисов | Медленнее из-за ручной проверки и синхронизированного программирования. |
| **Масштабируемость** | Масштабируемость, но ORM и шаблонизатор могут замедлить масштабирование | Высокая масштабируемость, поскольку использует асинхронный код и аннотации типов. | Сложно масштабировать, так как нет встроенной поддержки ORM или кэширования. |
| **Обучение** | Комплекс для начинающих | Просто для новичков | Умеренный для начинающих |
| **Инструменты базы данных** | Полный набор | Ограничено, нет встроенной поддержки | Ограничено, нет встроенной поддержки |
| **Асинхронное программирование** | Да, это можно сделать с помощью Asyncio, но немного медленнее. | Да, но быстрее благодаря Pydantic. | Нет |
| **ORM** | Да | Нет | Нет |
| **Используется в сообществе** | Большой и часто используемый | Маленький, но привлекающий внимание | Часто и активно |
| **Документация** | Огромная | Маленькая, но увеличивающиеся | Значительная |
| **Преимущества** | Это отличный выбор для тех, кто хочет создавать безопасные, масштабируемые, гибкие и простые в обслуживании веб-приложения. | Это отличный выбор для создания высокопроизводительных API и микросервисов. | Это отличный выбор для создания небольших и средних веб-приложений, где производительность не имеет большого значения, а разработчику нужна гибкость. |
| **Недостатки** | Может быть сложным для новичков, трудным в отладке и неподходящим для небольших проектов. | Основные файлы могут быть переполнены и иметь недостаток встроенной безопасности. | Нет встроенной поддержки кэширования, ORM, асинхронизации и т. д. |