Чефранов Михаил, 10 класс

МБОУ города Мурманска СОШ №36

Система управления проектами

Руководитель: Красовская А.Л.

# Введение

Современные организации все чаще используют специализированные системы для управления проектами, чтобы повысить эффективность работы и контролировать выполнение поставленных задач. Однако, не всегда доступны подходящие инструменты, которые позволяют пользователям легко создавать задачи, назначать их другим участникам команды и следить за прогрессом их выполнения. Существующие на данный момент программные продукты либо сложны в использовании (не обладают интуитивно понятным интерфейсом), либо поставляются в одном пакете с другими программными продуктами, которые не всегда нужны, и имеют достаточно высокую стоимость, а начинающие компании не могут себе позволить настолько дорогие финансовые решения.

Таким образом, можно сформулировать **проблему** исследования: организациям не хватает существующего ПО для эффективного управления проектами.

**Гипотеза:** реорганизация функциональной части и интуитивности дизайна позволит добиться понимания задач, поставленных человеку, а соответственно и продуктивность.

**Цель работы:** разработать систему управления проектами.

Исходя из цели, можно сформулировать список **задач**:

* Найти и проанализировать недостатки обычных источников.
* Смоделировать свой источник.
* Выбрать базу для хранения данных.
* Разработать программный код, способный собрать и обработать нужные данные.
* Создать интуитивно понятный, комфортный пользователю интерфейс.

Практическая значимость данной работы заключается в том, что данное приложение могут использовать все, кому необходимо управлять различными проектами: от образовательных учреждений, до крупных компаний.

# Теоретическая часть (приложение-планер задач)

Недостатки обычных источников

Решая первую задачу, я отметил, что затруднения возникают не только с количеством таких веб-приложений, но и качеством интерфейса, содержащегося в них.

Проблемы лежат на поверхности, и вот главные из них:

* Большое количество однотипных сервисов, которые выполняют одни и те же задачи пользователя.
* Платные подписки на данные сервисы за недемократичную цену.
* Отсутствие уведомлений об обновлении или незамеченные, незначительные обновления.
* Информация не всегда предоставляется в удобном для потребления виде (разбросана по разным частям приложения. Чтобы найти нужную информацию, пользователь тратить значительную часть времени).
* Реклама через определенные интервалы времени.

После этого я обратился к нашей аудитории с целью узнать, согласны ли они с нашим

видением проблемы. Аудиторией стали ученики школы и их родители, учителя. Среди

учеников 10 классов я провел небольшое анкетирование и устный опрос. Оказалось,

что мнение большинства полностью совпадает с моим – из 46 опрашиваемых 43

человека, или 96%, поддержали идею.

Моделирование замены

Изучая недостатки обычных источников, я сделал некоторые наброски модели и

сравнил её с “классикой”. Итак, преимущества нашего будущего продукта:

* Удобное редактирование задач, поставленных определенному человеку
* Практичный пользовательский интерфейс
* Пользователи могут создавать задачи, назначать их другим пользователям
* Можно отслеживать прогресс выполнения проекта.

**Подготовка среды разработки**

Приступая к главной задаче, написанию программного кода, следует подготовить нужные

инструменты для продуктивной работы.

Сначала настроим место, в котором мы будем проводить большую часть времени – среду

разработки Visual Studio Code:

1. Устанавливаем с официального сайта.

2. Создаём новый проект и настраиваем окружение.

a. В Visual Studio Code выбираем Open folder, указываем путь к проекту.

b. Создаем проект с использованием Django (команда: django-admin startproject taskmaster), создастся файл manager.py, с помощью которого мы будем управлять проектом.

3. Устанавливаем первые библиотеки.

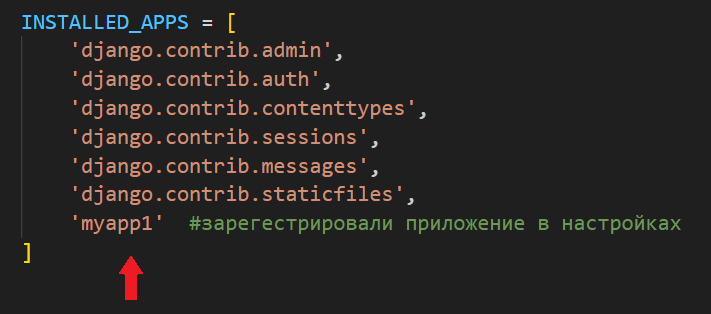
- Django [1] – веб-приложений на языке Python.

- Bootstrap [2] - бесплатный CSS-фреймворк

**Разработка серверной части кода**

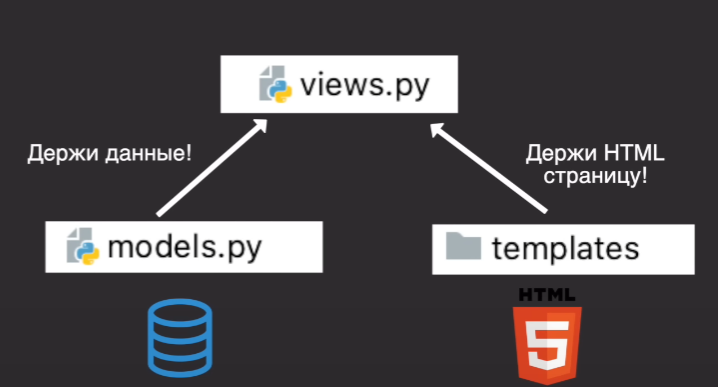
Сначала запустим локальный сервер, чтобы удостовериться, что веб-приложение работает корректно. Сделаем это в терминале командой “- *python manage.py runserver*”. Далее командой

“- *python manage.py startapp myapp1*” создадим веб-приложение с название myapp1. Чтобы с нашим приложением можно было работать, зарегистрируем его в главном файле настройки проекта: *settings.py.* Прописав в *settings.py* в переменной *INSTALLED\_APPS* название нашего веб-приложения, мы свяжем его с проектом.



Далее создадим папку template, в которой будут созданы html страницы, отвечающие за frontend страницы. Чтобы папку с шаблонами(templates) видел Django, нам также надо ее зарегистрировать ее в settings.py.

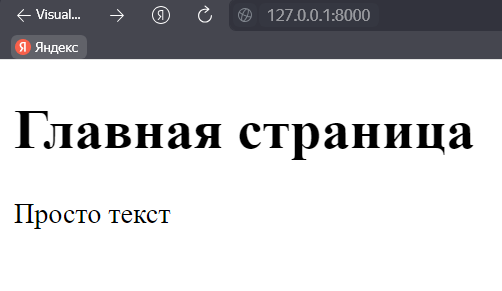
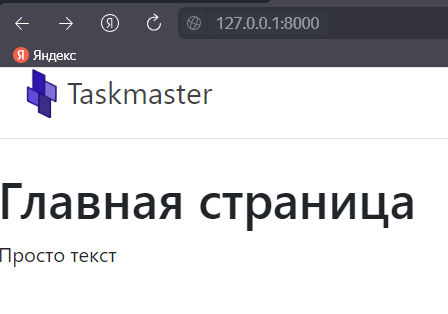
В папке templates буду лежать шаблоны html страниц, в файле models.py код для связей с базой данных, моделей, ответственных за данные. Чтобы связать модели и шаблоны нужен файл *views.py* для связи двух вышеописанных файлов. В Django это называется паттерном MVT (Models-View-Template).



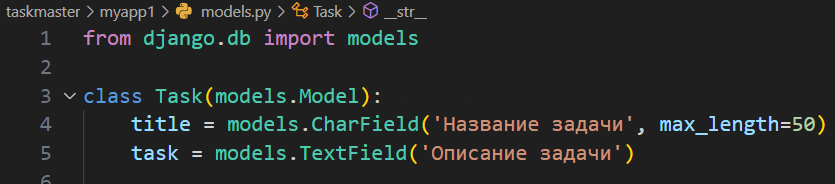
В файл *views.py* приходят http запросы от пользователя и отправляются http ответы.

В папке templates создали файл index.html и подключим стили из Bootstrap, так как это будет разумнее, чем использовать css.

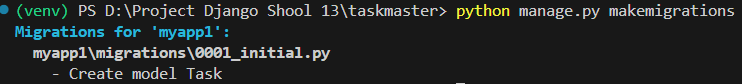
Было: Стало:

Создадим табличку в базе данных для хранения записей, которые заносит пользователь. Далее создадим класс, который отвечает за определенную табличку в базе данных.

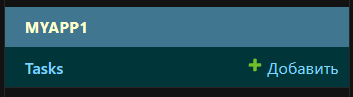


Далее проводим миграции (на основе прописанных моделей создадим таблички в базе данных) командой ‘*python3 manage.py makemigrations*’.

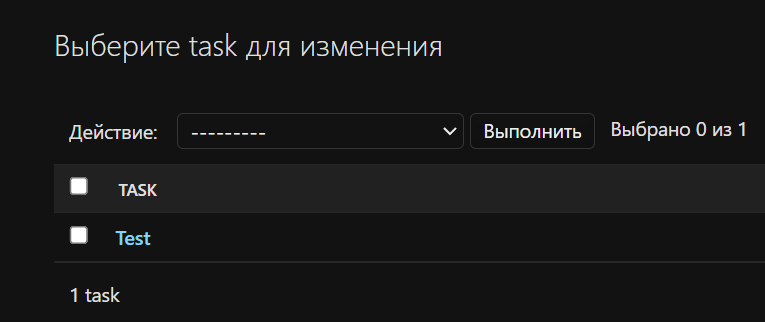


Добавляем таблицу в панель администратора, зарегистрировав ее в специальном файле *admin.py. Командой admin.site.register(Task)* регистрируем нашу модель Task.

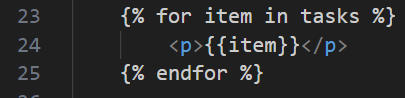
Она отобразилась в панели администратора.



Создаем и выводим тестовую модель task:

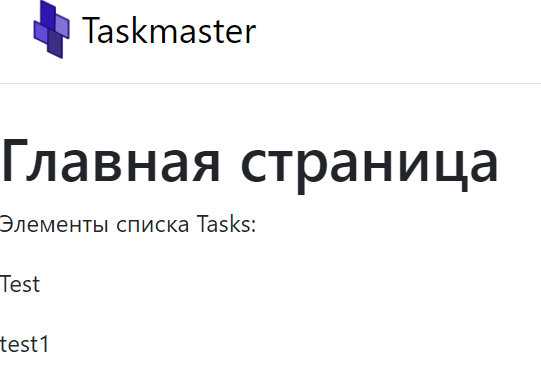
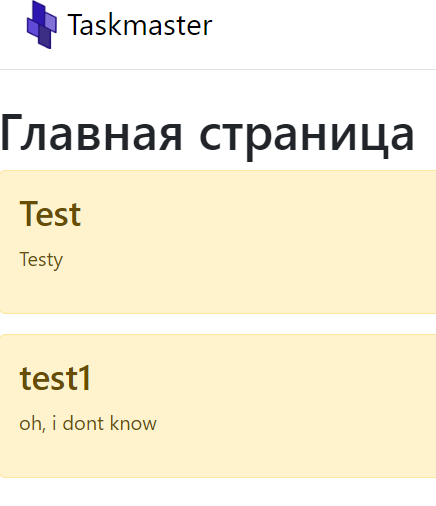


Вывод элементов списка на макет с помощью обработчика шаблонов Jinja:



В результате мы получаем данные, которые содержаться в списке tasks. Подредактировав код для красивого вывода элементов получаем:

До: После:

Добавлен сценарий: если действительных записей нет, то выводится соответствующая надпись:

