ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Веб-приложение

«Система управления задачами MiemTask»

Версия документа: 1.0
Дата создания: 2025
Статус:

Разработали:
Мациевич Г.А., студент СКБ231
Федоров Д.В., студент СКБ231

Утверждено:

_____

(подпись руководителя проекта)

Дата утверждения:

_____

Содержание

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Наименование системы

Веб-приложение «Система управления задачами MiemTask».

1.2. Основание для разработки

Академическое задание в рамках изучения дисциплины «Язык программирования Python» с использованием фреймворка Django.

1.3. Цель создания системы

Разработка веб-приложения для автоматизации процессов планирования, учета и контроля выполнения задач с возможностью категоризации, распределения между исполнителями и мониторинга статусов.

1.4. Назначение системы

Система предназначена для:

- Централизованного хранения информации о задачах
- Организации задач по категориям
- Отслеживания статусов выполнения
- Распределения задач между исполнителями
- Контроля сроков выполнения

1.5. Требования к составу и содержанию работ

Разработка включает следующие этапы:

1. Проектирование архитектуры системы
2. Реализация моделей данных
3. Разработка серверной логики
4. Создание пользовательского интерфейса
5. Интеграция компонентов
6. Тестирование функционала
7. Подготовка документации

2. ТРЕБОВАНИЯ К СИСТЕМЕ

2.1. Требования к функциям системы

2.1.1. Модуль управления задачами

Функция 2.1.1.1. Создание задачи

- Система должна предоставлять форму для создания новой задачи
- Обязательные поля: название, категория
- Опциональные поля: описание, дедлайн, исполнитель
- Система должна выполнять валидацию введенных данных
- При успешном создании система должна перенаправлять пользователя на страницу со списком задач

Функция 2.1.1.2. Просмотр задач

- Система должна отображать список всех задач в табличном или карточном виде
- Для каждой задачи должны отображаться: название, категория, статус, дедлайн, исполнитель
- Система должна предоставлять возможность просмотра детальной информации о задаче

Функция 2.1.1.3. Редактирование задачи

- Система должна предоставлять форму для изменения параметров существующей задачи
- Должна быть возможность изменения всех полей задачи
- Система должна сохранять изменения в базе данных

Функция 2.1.1.4. Удаление задачи

- Система должна предоставлять функцию удаления задачи
- Перед удалением должен отображаться запрос на подтверждение действия
- При подтверждении задача должна быть удалена из базы данных безвозвратно

Функция 2.1.1.5. Изменение статуса задачи

- Система должна позволять изменять статус задачи (выполнено/не выполнено)
- Изменение статуса должно быть доступно как на странице списка, так и на странице детального просмотра

2.1.2. Модуль управления категориями

Функция 2.1.2.1. Создание категории

- Система должна предоставлять форму для создания новой категории
- Обязательное поле: название (уникальное значение)
- Опциональное поле: описание

Функция 2.1.2.2. Просмотр категорий

- Система должна отображать список всех категорий

Функция 2.1.2.3. Редактирование категории

- Система должна позволять изменять название и описание категории

Функция 2.1.2.4. Удаление категории

- Система должна позволять удалять категории
- При удалении категории, связанные с ней задачи должны быть также удалены

2.1.3. Модуль фильтрации и поиска

Функция 2.1.3.1. Фильтрация задач

Система должна предоставлять фильтрацию задач по следующим параметрам:

- Статус выполнения (все/выполнено/не выполнено)
- Категория
- Исполнитель

Функция 2.1.3.2. Поиск задач

- Система должна предоставлять функцию поиска задач по названию
- Поиск должен быть регистронезависимым

Функция 2.1.3.3. Сортировка задач

Система должна предоставлять сортировку по:

- Дате создания (по возрастанию/убыванию)
- Дедлайну (по возрастанию/убыванию)

2.1.4. Дополнительные функции

Функция 2.1.4.1. Управление приоритетами задач

Система должна предоставлять возможность назначения приоритета задаче:

- Система должна поддерживать три уровня приоритета: Низкий, Средний, Высокий
- При создании задачи должна быть возможность выбора приоритета из выпадающего списка
- При редактировании задачи должна быть возможность изменения приоритета
- В списке задач приоритет должен отображаться с помощью визуального индикатора

Функция 2.1.4.2. Система комментариев к задачам

Система должна предоставлять возможность добавления комментариев к задачам:

- На странице детального просмотра задачи должен отображаться список всех комментариев
- Система должна предоставлять форму для добавления нового комментария
- Каждый комментарий должен содержать: текст, автора, дату и время создания
- Комментарии должны отображаться в хронологическом порядке (от новых к старым)
- Должна быть возможность удаления комментария его автором или администратором

2.2. Требования к базе данных

2.2.1. Логическая структура данных

Сущность: Category (Категория)

Поле	Тип данных	Ограничения	Описание
id	Integer	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор
name	VARCHAR(100)	NOT NULL, UNIQUE	Название категории
description	TEXT	NULL	Описание категории
created_at	DATETIME	NOT NULL, DEFAULT NOW()	Дата создания записи

Таблица 1: Структура таблицы Category

Сущность: Task (Задача)

Поле	Тип данных	Ограничения	Описание
id	Integer	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентификатор
title	VARCHAR(200)	NOT NULL	Название задачи
description	TEXT	NULL	Подробное описание
created_at	DATETIME	NOT NULL, DEFAULT NOW()	Дата создания
deadline	DATETIME	NULL	Срок выполнения
is_completed	BOOLEAN	NOT NULL, DEFAULT FALSE	Статус выполнения
category_id	Integer	FOREIGN KEY, NOT NULL	Ссылка на категорию
assignee_id	Integer	FOREIGN KEY, NULL	Ссылка на исполнителя

Таблица 2: Структура таблицы Task

Сущность: User (Пользователь)

Используется встроенная модель Django `django.contrib.auth.models.User` со следующими основными полями:

- username (имя пользователя)
- email (электронная почта)
- first_name (имя)
- last_name (фамилия)

2.3. Требования к пользовательскому интерфейсу

2.3.1. Общие требования к интерфейсу

- Интерфейс должен быть интуитивно понятным
- Должна быть обеспечена единообразная навигация на всех страницах
- Время отклика на действия пользователя не должно превышать 1 секунды

2.3.2. Структура навигации

Система должна содержать следующие основные разделы:

- Главная страница
- Список задач
- Список категорий
- Панель администрирования Django

2.3.3. Требования к визуальному оформлению

- Использование CSS-фреймворка Bootstrap
- Единая цветовая схема на всех страницах
- Визуальная индикация статусов задач:
  - Зелёный цвет — задача выполнена
  - Красный цвет — дедлайн просрочен
  - Жёлтый цвет — дедлайн приближается ($\leq$ 3 дня)

2.3.4. Структура страниц

Таблица 3: URL-маршруты системы

URL	**HTTP-метод**	**Название страницы**	**Функциональное назначение**
`/`	GET	Главная страница	Отображение Dashboard
`/tasks/`	GET	Список задач	Просмотр всех задач с фильтрами
`/tasks/create/`	GET, POST	Создание задачи	Форма добавления новой задачи
`/tasks/<int:pk>/`	GET	Детали задачи	Детальная информация о задаче
`/tasks/<int:pk>/edit/`	GET, POST	Редактирование	Форма изменения задачи
`/tasks/<int:pk>/delete/`	GET, POST	Удаление задачи	Подтверждение и удаление
`/categories/`	GET	Список категорий	Просмотр всех категорий
`/categories/create/`	GET, POST	Создание категории	Форма добавления категории
`/admin/`	GET, POST	Админ-панель	Административный интерфейс Django

2.4. Требования к административному интерфейсу

2.4.1. Модель Task

Административный интерфейс для модели Task должен включать:

- **Отображаемые поля в списке:** title, category, assignee, deadline, is_completed, created_at
- **Фильтры:** по категории, статусу выполнения, исполнителю, дате создания
- **Поиск:** по полям title и description
- **Возможность массового изменения:** стату is_completed
- **Сортировка по умолчанию:** по дате создания

2.4.2. Модель Category

Административный интерфейс для модели Category должен включать:

- **Отображаемые поля в списке:** name, description, created_at
- **Поиск:** по полю name
- **Сортировка по умолчанию:** по алфавиту

3. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

3.1. Требования к программному обеспечению серверной части

Компонент	**Версия**	**Назначение**
Python	≥ 3.10	Язык программирования
Django	≥ 4.2	Веб-фреймворк
SQLite	3.x	Система управления БД

Таблица 4: Программное обеспечение сервера

3.2. Требования к программному обеспечению клиентской части

- Веб-браузер с поддержкой HTML5, CSS3, JavaScript ES6+
- Рекомендуемые браузеры: Google Chrome ≥ 90, Mozilla Firefox ≥ 88, Safari ≥ 14, Microsoft Edge ≥ 90

3.3. Зависимости проекта

Файл `requirements.txt` должен содержать следующие пакеты:

```
Django>=4.2,<5.0
python-decouple>=3.8
```

Листинг 1: Содержимое requirements.txt

4. СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ

4.1. Этап 1. Инициализация проекта

Длительность: 1 рабочий день

Задачи:

1. Создание Django-проекта командой
2. Создание приложения
3. Настройка конфигурационного файла
4. Создание файлов `requirements.txt` и `README.md`
5. Инициализация системы контроля версий (Git)

Результат: Базовая структура проекта готова к разработке

4.2. Этап 2. Проектирование и реализация моделей данных

Длительность: 1 рабочий день

Задачи:

1. Создание моделей `Task` и `Category` в файле `models.py`
2. Определение связей между моделями
3. Создание и применение миграций
4. Регистрация моделей в административном интерфейсе
5. Настройка отображения моделей в админке

Результат: Модели данных созданы, миграции применены, административный интерфейс настроен

4.3. Этап 3. Разработка серверной логики (Views и URLs)

Длительность: 2 рабочих дня

Задачи:

1. Реализация представлений для CRUD-операций с задачами
2. Реализация представлений для работы с категориями
3. Создание представления для главной страницы

4. Настройка URL-маршрутизации
5. Реализация фильтрации и поиска
6. Реализация пагинации

Результат: Серверная логика реализована, все эндпоинты функционируют

4.4. Этап 4. Разработка пользовательского интерфейса

Длительность: 2 рабочих дня

Задачи:

1. Создание базового шаблона (`base.html`) с навигацией
2. Создание шаблонов для списка задач и детального просмотра
3. Создание форм для создания и редактирования задач
4. Создание шаблонов для работы с категориями
5. Интеграция Bootstrap
6. Стилизация элементов интерфейса

Результат: Создан полнофункциональный пользовательский интерфейс

4.5. Этап 5. Реализация дополнительного функционала

Длительность: 1–2 рабочих дня

Задачи (на выбор):

1. Добавление системы приоритетов задач
2. Реализация экспорта данных
3. Добавление системы комментариев
4. Реализация тегов для задач
5. Добавление уведомлений о просроченных задачах

Результат: Создан расширенный функционал системы

4.6. Этап 6. Тестирование и документация

Длительность: 1 рабочий день

Задачи:

1. Функциональное тестирование всех модулей
2. Тестирование пользовательского интерфейса
3. Проверка корректности работы административного интерфейса
4. Написание документации в `README.md`
5. Проверка соответствия техническому заданию
6. Устранение выявленных дефектов

Результат: Протестирована система, разработана документация

5. ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

5.1. Файл README.md

Файл `README.md` должен содержать следующие разделы:

1. **Название проекта** — краткое описание назначения
2. **Технологический стек** — перечень используемых технологий
3. **Требования** — версии ПО
4. **Установка** — пошаговая инструкция по развёртыванию
5. **Настройка** — команды для миграций и создания суперпользователя
6. **Запуск** — команда запуска сервера разработки
7. **Использование** — описание основного функционала
8. **Структура проекта** — описание директорий и файлов
9. **Авторы** — информация о разработчиках

5.2. Комментирование кода

- Сложные функции должны быть прокомментированы
- Модели должны содержать описание полей через комментарии

6. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ СИСТЕМЫ

6.1. Виды испытаний

6.1.1. Функциональное тестирование

- Проверка работоспособности всех CRUD-операций
- Проверка фильтрации и поиска
- Проверка корректности отображения данных

6.1.2. Тестирование интерфейса

- Проверка адаптивности на различных разрешениях
- Проверка навигации между страницами
- Проверка валидации форм

6.1.3. Тестирование административного интерфейса

- Проверка доступности админ-панели
- Проверка корректности работы фильтров и поиска
- Проверка возможности управления данными

6.2. Критерии приёмки

Система считается готовой к приёмке при выполнении следующих условий:

Таблица 5: Критерии приёмки системы

№	Критерий	Требование
1	Запуск системы	Сервер успешно запускается командой `python manage.py runserver`
2	База данных	Миграции применены без ошибок, модели созданы корректно

Продолжение на следующей странице

Таблица 5 – продолжение с предыдущей страницы

№	**Критерий**	**Требование**
3	CRUD-операции	Все операции создания, чтения, обновления и удаления функционируют
4	Административный интерфейс	Админ-панель доступна, модели корректно отображаются
5	Пользовательский интерфейс	Все страницы отображаются корректно, навигация работает
6	Фильтрация и поиск	Фильтры и поиск возвращают корректные результаты
7	Валидация данных	Система корректно обрабатывает некорректный ввод
8	Документация	Файлы README.md и requirements.txt присутствуют и содержат полную информацию
9	Качество кода	Код соответствует стандартам PEP 8, структурирован
10	Дополнительный функционал	Реализовано не менее 2 дополнительных функций