1 ВВЕДЕНИЕ

ToDo — это современное веб-приложение, предназначенное для управления личными задачами. Оно обеспечивает пользователей удобным интерфейсом для создания, редактирования и удаления задач, что позволяет легко отслеживать выполнение дел. Основной функционал включает возможность быстро добавлять новые задачи, а также изменять или удалять их, если они становятся неактуальными.

Пользователи могут отмечать задачи как выполненные, что помогает поддерживать актуальность списка. Кроме того, приложение предлагает функции фильтрации задач по статусу и приоритету, что позволяет сосредоточиться на наиболее важных делах. Сортировка по различным параметрам, таким как дата создания или срок выполнения, значительно упрощает организацию рабочего процесса.

ToDo Django также включает проверки корректности вводимых сроков, чтобы избежать ошибок. Просроченные задачи выделяются, что помогает пользователям своевременно реагировать на них. Есть возможность очищать список от завершённых задач, что способствует поддержанию порядка.

Несмотря на множество полезных функций, у приложения есть и ограничения. Оно не поддерживает многопользовательскую аутентификацию, что исключает возможность совместного использования с другими пользователями. Все задачи хранятся в одном списке, что может быть неудобно для пользователей с несколькими проектами. Кроме того, отсутствуют уведомления о предстоящих сроках выполнения, что может усложнить планирование. Приложение не интегрируется с внешними сервисами, такими как календари или почтовые клиенты, и в настоящее время доступно только через веб-браузер, что ограничивает использование на мобильных устройствах.

2 ТРЕБОВАНИЯ ПОЛЬЗОВАТЕЛЯ

2.1 Программные интерфейсы

Продукт взаимодействует с Django 5.x как с основным веб-фреймворком (представления, маршрутизация, шаблоны, система сообщений, CSRF, раздача статики) и с локальной базой данных SQLite через Django ORM (файл db.sqlite3, транзакции под управлением Django). Клиентская сторона — современный браузер, который рендерит HTML/CSS, выполняет JavaScript и отправляет AJAX-запросы через fetch (например, для переключения статуса задач и очистки выполненных), а также использует стандартное поле ввода дат и времени input[type="datetime-local"], дополненное серверными проверками. Для типографики подключается шрифт Inter с CDN Google Fonts по HTTPS (fonts.googleapis.com и fonts.gstatic.com).
Статические ресурсы приложения (CSS и JS)

обслуживаются подсистемой Django StaticFiles по HTTP. В продакшене при необходимости могут использоваться WSGI/ASGI-серверы (например, gunicorn или uvicorn/daphne) для публикации Django-приложения, а также внешняя СУБД (PostgreSQL или MySQL) вместо SQLite через соответствующие драйверы и настройки Django.

2.2 Интерфейс пользователя

Пользователь взаимодействует с системой через веб-интерфейс в браузере. После открытия главной страницы отображается список задач в виде карточек, расположенных в адаптивной сетке; в шапке страницы доступна кнопка «Новая задача». Над списком размещена панель управления с выпадающими списками для фильтрации по статусу и приоритету, выпадающим списком сортировки (по дате создания, сроку, приоритету, статусу), кнопкой применения фильтров и кнопкой «Очистить выполненные». Каждая карточка содержит название задачи, бейджи приоритета и статуса, опциональное описание, даты создания/срока/завершения, а также чекбокс, позволяющий отметить задачу выполненной. Если срок задачи истёк и она не завершена, карточка визуально подсвечивается и получает отметку «Просрочено». При нажатии чекбокса статус задачи переключается асинхронно без перезагрузки страницы, карточка становится полупрозрачной, а статус меняется на «Выполнено»; при повторном нажатии задача возвращается в «К выполнению». Нажатие «Редактировать» открывает форму с предзаполненными полями; «Удалить» открывает страницу подтверждения, после подтверждения карточка исчезает и выводится уведомление об успешном удалении. Кнопка «Очистить выполненные» удаляет все завершённые задачи и показывает уведомление о количестве удалённых. Создание/редактирование выполняется через отдельную форму с полями «Название» (обязательно), «Описание», «Срок выполнения», «Приоритет» и «Статус». Поле срока использует формат datetime-local; при вводе значения в прошлом, вне допустимого диапазона лет или с секундами серверная валидация отклоняет запрос и возвращает страницу формы с сообщениями об ошибках, подсвечивая проблемные поля. Применение фильтров и сортировки приводит к мгновенному обновлению списка согласно выбранным параметрам; при пустом списке отображается информативное сообщение о том, что задач пока нет. Интерфейс выполнен в тёмной теме с контрастной типографикой, корректно адаптируется под узкие экраны: элементы управления переносятся в несколько рядов, карточки растягиваются

на всю ширину, интерактивные элементы остаются крупными и удобными для тач-ввода

2.3 Характеристики пользователя

Основная группа пользователей — индивидуальные пользователи, которым нужен простой инструмент для личного планирования задач. Это студенты, специалисты начального и среднего уровней, фрилансеры и сотрудники небольших команд, ведущие свои списки дел без сложной коллаборации. Уровень образования — средний и выше; специфического технического бэкграунда не требуется. Техническая грамотность — базовая: умение работать с современным браузером, формами ввода, выпадающими списками и кнопками действий. Пользователи ожидают интуитивный интерфейс без необходимости обучения, понятные подписи полей и предсказуемую реакцию системы (создал — увидел, нажал — выполнено). Для части аудитории важна адаптация под мобильные устройства и комфортная тёмная тема. Предполагается, что пользователи способны формулировать задачи краткими текстами, иногда указывать сроки и приоритеты, но не готовы тратить много времени на настройку, поэтому система минимизирует количество обязательных полей и шагов. Административная роль не выделяется: приложение предназначено для личного пользования, без разграничения прав и многопользовательских сценариев.

2.4 Предположения и зависимости

Требования зависят от среды запуска (версии Python/Django, конфигурации WSGI/ASGI и БД; при переходе с SQLite на PostgreSQL/MySQL меняются миграции, настройки и нефункциональные ограничения),
от поддержки браузером элемента input[type="datetime-local"] (при слабой поддержке растёт роль серверной валидации и возможны UX-правки),
от настроек часовых поясов и системного времени (расхождение часов, смена таймзоны, летнее время влияют на определение «просрочено»),
от объёмов данных (рост количества задач потребует пагинации, индексов, кэширования и может изменить SLO по отклику), от целевых устройств (мобильный трафик требует усиленной адаптивности, доступности и тестов на узких экранах),
от требований доступности (контраст, клавиатурная навигация, ARIA могут превратиться в обязательные),
от политики безопасности (CSRF/HTTPS, заголовки, изоляция статики; при появлении аутентификации — хранение персональных данных и

требования к паролям/сессиям), от локализации (изменение языка/форматов дат и чисел), от внешней сети и CDN (доступность Google Fonts влияет на типографику и время загрузки, возможно потребуется локальный бандл шрифтов), от инфраструктуры развертывания (контейнеризация, реверс-прокси, балансировщики и их влияние на заголовки и кеширование), а также от будущих функциональных расширений (многопользовательский режим, уведомления, интеграции с календарём/почтой), которые потребуют пересмотра границ, моделей, прав доступа и нефункциональных метрик

3 СИСТЕМНЫЕ ТРЕБОВАНИЯ

3.1 Функциональные требования

- Приложение должно позволять создавать задачу с полями: название (обязательно), описание (необязательно), срок выполнения (необязательно), приоритет {low, medium, high}, статус {todo, in_progress, done}.
- Система должна отображать список задач в виде карточек с бейджами приоритета и статуса, датами создания/срока/завершения и чекбоксом выполнения.
- Должна быть фильтрация задач по статусу.
- Должна быть фильтрация задач по приоритету.
- Должна быть сортировка задач по дате создания (возр./убыв.), по сроку выполнения, по приоритету и по статусу.
- Должно быть редактирование задачи (все перечисленные поля).
- Должно быть удаление задачи с подтверждением.
- Переключение выполнения задачи (чекбокс) должно работать без перезагрузки страницы, синхронизируя флаг completed и статус (done/todo).
- Просроченные невыполненные задачи должны визуально подсвечиваться и отмечаться как «Просрочено».
- Должна быть кнопка «Очистить выполненные», удаляющая все задачи со статусом завершено/флагом completed.
- Система должна возвращать пользователю уведомления об успехе/ошибках операций (создание, обновление, удаление, очистка).
- Интерфейс должен быть русскоязычным и адаптивным, корректно работать на узких экранах.
- Ввод срока выполнения через datetime-local должен иметь клиентские ограничения (min/max/step) и серверную валидацию; сохранение с невалидной датой должно быть отклонено с сообщением об ошибке.

- При завершении задачи система должна автоматически устанавливать дату завершения; при отмене завершения — очищать её.
- Все операции изменения данных (создание, редактирование, удаление, переключение, очистка) должны быть защищены от CSRF и выполняться по POST.

3.2 Нефункциональные требования

Атрибуты качества. Надёжность: важна для сохранности данных задач и предсказуемости поведения; измеряется отсутствием необработанных исключений в основных CRUD-сценариях (0 критических ошибок за регрессионный прогон), успешным прохождением негативных кейсов валидации сроков (100% отклонений некорректных дат), консистентностью статуса и флага completed (0 расхождений в выборках). Производительность: важна для мгновенного отклика интерфейса; измеряется временем рендера списка при 200 задачах < 200 мс на локальной машине и временем обработки AJAX-переключения < 100 мс на сервере разработки (лог/профилировка). Юзабилити: важна для низкого порога входа; измеряется успешным прохождением ключевых сценариев без инструкций (создание/переключение/фильтры/очистка), отсутствием горизонтального скролла при ширине 320 px и количеством кликов не более 3 до целевого действия. Безопасность: важна для целостности данных и защиты от CSRF; измеряется наличием и валидацией CSRF-токена во всех POST-запросах (100%), отсутствием смешанного контента, корректной обработкой недопустимого ввода (отсутствие XSS-инъекций через поля формы). Поддерживаемость: важна для быстрого внесения изменений; измеряется временем добавления нового поля задачи (модель, форма, шаблон) ≤ 30 минут, долей переиспользуемого кода, линтер-ошибок = 0. Совместимость: важна для корректной работы в популярных браузерах; измеряется прохождением ручного чек-листа в актуальных версиях Chromium/Firefox/Edge и современном Safari (100% сценариев), корректной деградацией при ограниченной поддержке datetime-local (серверная валидация блокирует неверные даты). Доступность: важна для удобства широкой аудитории; измеряется контрастностью по WCAG AA для основных элементов, возможностью навигации клавиатурой по интерактивным элементам и наличием фокуса. Масштабируемость (ограниченно актуальна): важна при росте числа задач; измеряется линейной деградацией времени рендера при увеличении списка (например, 500 задач — < 400 мс) и возможностью включить пагинацию/индексы без переработки архитектуры. Наблюдаемость:

важна для диагностики; измеряется наличием логирования ошибок сервера и предупреждений валидации, отсутствием ошибок в консоли браузера при стандартных сценариях. Портируемость/развёртываемость: важна для переноса с dev на prod; измеряется установкой на чистое окружение за $\leq$ 10 минут (venv, миграции, collectstatic при необходимости) и минимальным числом конфигурационных параметров (БД, таймзона, DEBUG).