669452, Иркутская область, Аларский район, пос. Кутулик, ул.Матвеева, 47

Тел.8(395)64-37-174, факс 8(395)64-37-174. E-mail:kutulik@list.ru

ОКПО -41781506, ОГРН-102850056709, ИНН/КПП

8501000373/850101001

Проект на тему «Управление задачами. Приложение Задачник»

НАПРАВЛЕНИЕ: ИНФОРМАЦИОННОЕ

ВИД ПРОЕКТА:ПРАКТИКО-ОРИЕНТИРОВАННЫЙ

Выполнили: Ученики 10класса

Иванова Яна, Немчинова Кристина, Баргуев Виктор, Мозжанов Семён

Руководитель: Учитель математики

и информатики МБОУ Кутуликской СОШ

Прокопьев Алексей Анатольевич

2025г

Содержание:

**Введение:**

В современном мире, насыщенном информацией и постоянными изменениями, эффективное управление временем и задачами становится ключевым фактором успеха как в личной жизни, так и в профессиональной деятельности. Сложности в организации задач могут привести к потере времени, снижению продуктивности и повышению уровня стресса. В связи с этим возникает необходимость в разработке инструментов, которые помогут людям более эффективно планировать и отслеживать выполнение своих задач.

**Цель проекта:**

Основная цель нашего проекта — создание интуитивно понятного и функционального приложения для управления задачами, которое позволит пользователям выполнять разные личные или профессиональные задачи.

**Задачи проекта:**

1. Создавать и редактировать задачи: Легкий доступ к добавлению новых задач и изменению существующих.

2. Устанавливать приоритеты: Возможность назначения приоритетов для каждой задачи, что поможет сосредоточиться на наиболее важных делах.

3. Повышать общую продуктивность пользователей за счет структурированного подхода к планированию.

4. Снижать уровень стресса, связанного с забыванием о важных делах.

**Актуальность проекта:**

Согласно исследованиям, большинство людей сталкиваются с трудностями в управлении своими задачами, что негативно сказывается на их личной и профессиональной жизни. Приложение задачник призвано помочь пользователям систематизировать свои дела, устанавливать приоритеты и контролировать выполнение задач. Это позволит не только повысить продуктивность, но и снизить уровень стресса, связанного с забыванием о важных делах.

**Уникальность проекта:**

Платформа предлагает уникальный пользовательский интерфейс, который сочетает в себе простоту и функциональность. Функция совместного редактирования задач и напоминаний позволяет командам работать над задачами в реальном времени, что повышает продуктивность и улучшает коммуникацию.

**Разработка дизайна:**

* Создание начали с изучения программы Figma и освоения работы с ней.
* Рассмотрели похожие приложения, для того чтобы создать более удобный интерфейс.
* Определили, целевую аудиторию. Это помогло выбрать стиль, цветовую палитру и шрифты.
* Создали Frame для мобильного и компьютерного приложения.
* Продумали, как элементы будут расположены на странице. Использовали сетки для упрощения компоновки.
* Подобрали нужные цвета и формы, которые хотели использовать для данной аудитории.
* Также подобрали иконки и шрифты.
* Следили за контрастом между текстом и фоном, чтобы обеспечить хорошую читаемость.
* Поддержали единый стиль во всех элементах дизайна. Это включает в себя использование одинаковых цветовых схем, шрифтов и стилей кнопок на всех страницах.
* Создали все страницы и адаптировали их под мобильное приложение.
* Провели тестирование с реальными пользователями, чтобы выявить проблемы в дизайне и понять, что можно улучшить.
* Выявили проблемы и исправили их.

**Теория и практика**

**Основные функции и требования:**

Создание новой заметки:

Интерфейс для ввода заголовка заметки (текстовое поле, обязательное для заполнения).

Интерфейс для ввода содержания заметки (многострочное текстовое поле/текстовый редактор, обязательное для заполнения).

Кнопка “Сохранить” для сохранения заметки в базе данных.

Валидация введенных данных (проверка на пустоту заголовка и содержания).

Просмотр списка заметок:

Отображение краткой информации о каждой заметке (заголовок, первые несколько строк содержания или превью).

Возможность сортировки списка заметок по дате создания, заголовку или другим критериям (опционально).

Пагинация (разбиение на страницы) для большого количества заметок (опционально).

Просмотр отдельной заметки:

Отображение полного заголовка и содержания заметки.

Кнопки для редактирования и удаления заметки.

Отображение даты создания заметки (и даты последнего изменения, если применимо).

Редактирование заметки:

Предоставление интерфейса, аналогичного созданию новой заметки, с предварительно заполненными полями (заголовок и содержание).

Кнопка “Сохранить” для сохранения изменений в базе данных.

Валидация введенных данных (аналогично созданию новой заметки).

Удаление заметки:

Кнопка “Удалить” на странице просмотра отдельной заметки.

Подтверждение удаления (всплывающее окно или другое предупреждение, чтобы предотвратить случайное удаление).

Удаление заметки из базы данных.

Управление избранными заметками (расширенная функциональность):

Кнопка “Добавить в избранное” / “Удалить из избранного” на странице просмотра отдельной заметки.

Возможность фильтрации списка заметок для отображения только избранных.

Визуальное обозначение избранных заметок в общем списке.

Поиск заметок (расширенная функциональность):

Поле для ввода поискового запроса.

Поиск заметок по совпадениям в заголовке и/или содержании.

Отображение результатов поиска.

Учет регистра и/или морфологии при поиске (опционально).

3. Используемые технологии:

Python (язык программирования):

Выбор Python обусловлен его простотой, читаемостью и широким набором библиотек для веб-разработки.

Python используется для реализации логики серверной части приложения, обработки запросов, взаимодействия с базой данных и генерации HTML-страниц.

Flask (микрофреймворк):

Flask обеспечивает необходимый минимум для создания веб-приложения, предоставляя гибкость и контроль над процессом разработки.

Flask используется для маршрутизации URL-адресов, обработки HTTP-запросов и ответов, и интеграции с шаблонизатором Jinja2.

HTML (язык разметки):

HTML используется для определения структуры и содержимого веб-страниц.

HTML-шаблоны создаются для главной страницы, страницы создания новой заметки, страницы просмотра отдельной заметки и других страниц приложения.

CSS (язык стилей):

CSS используется для определения внешнего вида веб-страниц, включая цвета, шрифты, размеры, расположение элементов и т.д.

CSS позволяет создать привлекательный и удобный пользовательский интерфейс.

JavaScript (язык программирования):

JavaScript используется для добавления интерактивности на веб-страницы, например, для реализации подтверждения удаления, динамической фильтрации заметок или расширенного текстового редактора.

Использование JavaScript опционально и зависит от требуемого уровня интерактивности приложения.

SQLite (легковесная база данных):

SQLite – это небольшая, самодостаточная и файловая база данных, которая не требует отдельного сервера.

SQLite подходит для небольших приложений и прототипирования.

Для более крупных проектов может быть использована другая СУБД, такая как PostgreSQL или MySQL.

SQLAlchemy (ORM библиотека):

SQLAlchemy предоставляет абстракцию над базой данных, позволяя работать с базой данных как с объектами Python.

SQLAlchemy упрощает взаимодействие с базой данных, обеспечивая удобный и безопасный способ выполнения операций CRUD.

Jinja2 (шаблонизатор):

Jinja2 используется для динамической генерации HTML-страниц на основе данных.

Jinja2 позволяет создавать гибкие и переиспользуемые HTML-шаблоны.

4. Архитектура приложения (Model-View-Controller):

Приложение будет организовано в соответствии с архитектурным паттерном MVC:

Model (Модель):

Представляет данные приложения (класс Note).

Определяет структуру данных (поля класса Note).

Предоставляет методы для работы с данными (создание, чтение, обновление, удаление).

Взаимодействует с базой данных через SQLAlchemy.

View (Представление):

HTML-шаблоны, определяющие внешний вид веб-страниц.

Отображают данные пользователю.

Используют Jinja2 для вставки динамических данных.

Controller (Контроллер):

Функции Python (маршруты Flask), обрабатывающие HTTP-запросы.

Взаимодействуют с моделью для получения и изменения данных.

Выбирают соответствующее представление для отображения данных.

Управляют логикой приложения.

5. Этапы реализации:

**Настройка окружения:**

Установка Python (версии 3.6 или выше).

Создание виртуального окружения:

bash

python3 -m venv venv

source venv/bin/activate # Linux/macOS

venv\Scripts\activate.bat # Windows

Установка необходимых библиотек:

bash

pip install flask flask-sqlalchemy

Создание структуры проекта:

my\_notes\_app/

├── app.py # Основной файл приложения

├── models.py # Определение моделей данных (класс Note)

├── templates/ # Каталог для HTML-шаблонов

│ ├── index.html

│ ├── create.html

│ ├── note.html

│ └── edit.html

└── static/ # Каталог для статических файлов (CSS, JavaScript)

└── style.css

Определение модели данных (models.py):

python

from flask\_sqlalchemy import SQLAlchemy

from your\_app import app # Замените your\_app на имя вашего файла app.py

db = SQLAlchemy(app)

class Note(db.Model):

id = db.Column(db.Integer, primary\_key=True)

title = db.Column(db.String(100), nullable=False)

content = db.Column(db.Text, nullable=False)

created\_at = db.Column(db.DateTime, server\_default=db.func.now()) # Дата создания

is\_favorite = db.Column(db.Boolean, default=False)

def \_\_repr\_\_(self):

return f'<Note {self.title}>'

Настройка базы данных (app.py):

python

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

import os

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///notes.db' # URI базы данных

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False # Отключаем отслеживание изменений SQLAlchemy (для экономии ресурсов)

app.secret\_key = os.urandom(24) # Секретный ключ для сессий (необходим для Flash-сообщений)

db = SQLAlchemy(app)

from your\_app import models # Импортируем models после инициализации db, чтобы избежать циклического импорта

with app.app\_context():

db.create\_all()

Реализация маршрутов и представлений (app.py):

python

from flask import render\_template, request, redirect, url\_for, flash

from your\_app import app, db, models # Замените your\_app на имя вашего файла app.py

from your\_app.models import Note

@app.route('/')

def index():

notes = Note.query.all()

return render\_template('index.html', notes=notes)

@app.route('/create', methods=['POST', 'GET'])

def create():

if request.method == 'POST':

title = request.form['title']

content = request.form['content']

if not title or not content:

flash('Заголовок и содержание обязательны!', 'error') # Выводим сообщение об ошибке

return render\_template('create.html')

new\_note = Note(title=title, content=content)

db.session.add(new\_note)

db.session.commit()

flash('Заметка успешно создана!', 'success') # Выводим сообщение об успехе

return redirect(url\_for('index'))

else:

return render\_template('create.html')

@app.route('/note/<int:id>')

def note(id):

note = Note.query.get\_or\_404(id)

return render\_template('note.html', note=note)

# Реализация маршрутов для редактирования, удаления, добавления в избранное и т.д. (аналогично)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Создание HTML-шаблонов (templates):

Создание шаблонов index.html, create.html, note.html, edit.html с использованием HTML, CSS и Jinja2.

Обеспечение базового стилистического оформления для удобства использования.

Примеры содержимого шаблонов (упрощенные):

*html*

*<!-- templates/index.html -->*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<title>Мои Заметки</title>*

*<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">*

*</head>*

*<body>*

*<h1>Мои Заметки</h1>*

*{% with messages = get\_flashed\_messages(with\_categories=true) %}*

*{% if messages %}*

*<ul class=flashes>*

*{% for category, message in messages %}*

*<li class="{{ category }}">{{ message }}</li>*

*{% endfor %}*

*</ul>*

*{% endif %}*

*{% endwith %}*

*<a href="{{ url\_for('create') }}">Создать заметку</a>*

*<ul>*

*{% for note in notes %}*

*<li><a href="{{ url\_for('note', id=note.id) }}">{{ note.title }}</a></li>*

*{% endfor %}*

*</ul>*

*</body>*

*</html>*

*<!-- templates/create.html -->*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<title>Создать заметку</title>*

*<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">*

*</head>*

*<body>*

*<h1>Создать заметку</h1>*

*<form method="POST">*

*<label for="title">Заголовок:</label><br>*

*<input type="text" id="title" name="title" required><br><br>*

*<label for="content">Содержание:</label><br>*

*<textarea id="content" name="content" required></textarea><br><br>*

*<input type="submit" value="Создать">*

*</form>*

*<a href="{{ url\_for('index') }}">Отмена</a>*

*</body>*

*</html>*

Запуск и тестирование:

Запуск приложения ( python app.py).

Открытие веб-браузера и переход по адресу http://127.0.0.1:5000/.

Тестирование основных функций приложения (создание, просмотр, редактирование, удаление заметок).

**Заключение:**

В ходе разработки приложения для управления задачами мы достигли поставленных целей и реализовали основные функции, которые помогают пользователям эффективно организовывать свое время и контролировать выполнение задач. Приложение предлагает интуитивно понятный интерфейс, позволяющий пользователям легко создавать, редактировать и отслеживать свои задачи. Мы внедрили такие ключевые функции, как установка приоритетов, настройка напоминаний и возможность интеграции с другими сервисами, что значительно улучшает пользовательский опыт.

Проведенное тестирование показало, что приложение отвечает требованиям целевой аудитории и позволяет пользователям значительно повысить свою продуктивность. Обратная связь от пользователей подтвердила, что предложенные функции действительно помогают в управлении задачами и способствуют более организованному подходу к работе и личным делам.

Несмотря на достигнутые результаты, в ходе работы над проектом были выявлены и некоторые ограничения. Например, необходимость дальнейшей оптимизации приложения для работы на различных устройствах и улучшения функционала интеграции с популярными календарями и планировщиками. В будущем мы планируем расширить функционал приложения, добавив возможности анализа производительности и создания отчетов о выполнении задач.

Таким образом, разработанное приложение для управления задачами представляет собой эффективный инструмент для пользователей, стремящихся к улучшению своей организованности и повышению продуктивности. Мы уверены, что дальнейшее развитие и совершенствование приложения сделает его еще более полезным и востребованным среди широкой аудитории.

**В будущем мы планируем добавить:**

* Интегрироваться с другими сервисами: Возможность синхронизации с календарями и другими приложениями для более удобного управления временем.
* Настраиваемые напоминания о сроках выполнения задач.
* Возможность делиться задачами с другими пользователями, назначать задачи коллегам.