**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

|  |  |
| --- | --- |
|  |  |

**Курсова робота**

**з дисципліни «Програмування»**

**на тему: «Планувальник задач»**

Виконав:

студент 1 курсу, групи ІА-33

Перегуда Павло Олександрович \_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник:

асистент кафедри ІСТ

Мягкий Михайло Юрійович \_\_\_\_\_\_\_\_\_\_

(підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2024 року

ЗМІСТ

[ВСТУП 3](#_Toc158383261)

[1 ВИМОГИ ДО СИСТЕМИ 4](#_Toc158383262)

[1.1 Функціональні вимоги до системи 4](#_Toc158383263)

[1.2 Нефункціональні вимоги до системи 4](#_Toc158383264)

[2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ 5](#_Toc158383265)

[2.1 Діаграма прецедентів 5](#_Toc158383266)

[2.2 Опис сценаріїв використання системи 6](#_Toc158383267)

[3 АРХІТЕКТУРА СИСТЕМИ 11](#_Toc158383268)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 13](#_Toc158383269)

[4.1 Загальна структура проекту 13](#_Toc158383270)

[4.2 Компоненти рівня доступу до даних 14](#_Toc158383271)

[4.3 Компоненти рівня бізнес-логіки 15](#_Toc158383272)

[4.4 Компоненти рівня інтерфейсу користувача 16](#_Toc158383273)

[ВИСНОВКИ 17](#_Toc158383274)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc158383275)

[ДОДАТОК А Лістінг програми 22](#_Toc158383276)

[ДОДАТОК Б Назва додатку Б (за наявності) 23](#_Toc158383277)

[ДОДАТОК В Назва додатку В (за наявності) 24](#_Toc158383278)

# ВСТУП

# З початком карантину дуже сильно набув популярності “культ успішного успіху”. І всі почали невпинно працювати по 18 годин на день, без вихідних, для того щоб, встигати за такими самими як і вони (ненормальними), та все ж таки “чогось досягти в цьому житті”. Насправді, це велика зараз проблема з цим культом, адже через нього багато хто просто втрачає сенс життя і взагалі не розуміє: “А що я взагалі зараз роблю?” Але зараз не про це. Для того, щоб організувати свій день, тиждень чи місяць нам обов’язково варто десь записувати наший список задач. Тим самим, ми вже себе розвантажуємо зайвою інформацією. Тому, що якщо все тримати в голові, то хорошим аж ніяк не закінчиться. І цьому всьому просте рішення. І на допомогу нам приходить “планувальник задач”. Нині, на жаль, багато людей не виділяють часу для того, аби просто записати список справ, задач які необхідно їм виконати. Ця справа займе лише декілька хвилин часу, але це неймовірно сильно полегшує день. Тепер більше не доведеться тримати у голові всю інформацію. Адже вона у вас буде вже записана і вам не доведеться постійно згадувати: “А що ще я сьогодні не зробив?”

І рішенням цього є мій простенький “Планувальник задач”, який має такий самий простенький функціонал. Він звісно не стане рішенням проблем (бо з ними власноруч треба розбиратися), але організуватися, що за чим робити точно допоможе “Планувальник задач”. І зараз коротко опишу його мету та функціонал.

Метою роботи є створення системи, що дозволяє записати список задач, які можна розділити за пріоритетом, датою. Для цього система повинна мати певний перелік властивостей та вирішувати такі задачі:

* система повинна створювати, редагувати, видаляти інформації про задачу
* система повинна мати функцію, відмітити задачу як «виконана»
* система повинна мати, сортування за пріоритетом чи датою

# 1 ВИМОГИ ДО СИСТЕМИ

## 1.1 Функціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

– незареєстрований користувач повинен мати можливість переглядати інформацію про задачі;

– незареєстрований користувач повинен мати можливість додавати та редагувати задачі до списку;

– незареєстрований користувач повинен мати можливість сортувати список задач за датою та пріоритетом;

– незареєстрований користувач повинен мати можливість переглядати задачі;

## 1.2 Нефункціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

– система повинна мати відкриту архітектуру;

– система повинна мати веб-інтерфейс;

– інтерфейс користувача має бути зручним та інтуїтивно-зрозумілим;

– система повинна бути крос-платформною.

# 2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

# 2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

користувачі системи є: Гість

Гостю доступна уся функціональність, тобто гість може додавати, редагувати, сортувати та позначати виконаність задач

Зображення, що містить схема, текст, коло, ряд

Автоматично згенерований опис

Рисунок 2.1 – Діаграма прецедентів

## 2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.17.

Таблиця 2.1 – Сценарій використання «Додавання задачі»

|  |  |
| --- | --- |
| Назва | Додавання задачі |
| ID | 1 |
| Опис | Користувач, використовуючи поле “назва задачі”, пише назву задачі. Встановлює дату, пріоритет та натискає “Додати задачу” |
| Актори | Користувач |
| Вигоди компанії | Без можливості додавання задачі, втрачається повністю весь сенс проєкту сайту та його головна мета |
| Частота користування | Постійно |
| Тригери | Користувач вводить у поле задачу у відповідному полі |
| Передумови | Поле для додавання задачі знаходиться у головному та єдиному вікні |
| Постумови | Користувач потрапляє на вікно з результатами пошуку |
| Основний розвиток | Користувач вводить задачу у строку, натискає на кнопку “Додати задачу” |
| Альтернативні розвитки | – |
| Виняткові ситуації | Якщо користувач не введе в поле будь-який символ, він не зможе додати задачу |

Таблиця 2.2 – Сценарій використання «Видалення задачі»

|  |  |
| --- | --- |
| Назва | Видалення задачі |
| ID | 2 |
| Опис | Після того як користувач додає задачу у списку всіх доданих задач буде присутня кнопка “Видалити” |
| Актори | Користувач |
| Вигоди компанії | Можливість видалення задачі є необхідним для комфортного користуванням проєкту |
| Частота користування | Постійно |
| Тригери | Користувач натискає на кнопку “Видалити ” біля задачі |
| Передумови | У вікні є список задач, які можна видаляти |
| Постумови | Задача видалена |
| Основний розвиток | Користувач натискає на кнопку “Видалити ”  Задача видаляється |
| Альтернативні розвитки | - |
| Виняткові ситуації | - |

Таблиця 2.3 – Сценарій використання «Редагування задачі»

|  |  |
| --- | --- |
| Назва | Редагування задачі |
| ID | 3 |
| Опис | Користувач переглядає головну сторінку і натискає “Редагувати”  Після цього він може змінювати назву задачі, пріоритет та дату |
| Актори | Користувач |
| Вигоди компанії | Можливість редагування задачі - є однією з найважливіших та основних аспектів користування для будь-якого схожого сайта. Зручний і простий інтерфейс редагування буде лише приваблювати користувачів залишатися на сайті |
| Частота користування | Постійно |
| Тригери | Користувач натискає “Редагувати” |
| Передумови | У вікні має бути список задач, які можна редагувати |
| Постумови | Відредагована задача |
| Основний розвиток | Користувач натискає “Редагувати” та на виході має відредаговану задачу |
| Альтернативні розвитки | – |
| Виняткові ситуації |  |

В таблиці 2.4 представлений сценарій використання «Пошук композицій по наданому користувачем аудіо із мікрофону або файлу»

Таблиця 2.4 – Сценарій використання «Позначення виконаної задачі»

|  |  |
| --- | --- |
| Назва | Позначення виконаної задачі |
| ID | 4 |
| Опис | Користувач може натиснути на місце для галочки. І після цього задача закреслиться |
| Актори | Користувач |
| Вигоди компанії | Один з ключових компонентів схожих програм, які показують виконаність задачі. Без цієї функції втрачається головна мета планувальників задач |
| Частота користування | Постійно |
| Тригери | Користувач натискає на відповідну кнопку |
| Передумови | Користувач прописав список задач |
| Постумови | Користувач шляхом натискання відповідної кнопки, закреслює задачу |
| Основний розвиток | Користувач натискає на відповідні кнопки |
| Альтернативні розвитки | - |
| Виняткові ситуації | – |

Таблиця 2.5 – Сценарій використання «Сортування задачі»

|  |  |
| --- | --- |
| Назва | Сортування задачі |
| ID | 5 |
| Опис | Після написання певного переліку задач їх можна посортувати за двома критеріями: дата та пріоритет |
| Актори | Користувач |
| Вигоди компанії | Легкий в реалізації інтерфейс, який буде утримувати користувачів на сайті |
| Частота користування | Постійно |
| Тригери | Користувач натискає “Сортувати” |
| Передумови | Користувач знаходиться на головній сторінці |
| Постумови | Користувач має відсортований список в залежності від фільтру |
| Основний розвиток | У вікні із результатами пошуку користувач натискає на спеціальну кнопку |
| Альтернативні розвитки | – |
| Виняткові ситуації | Якщо немає задач, то немає і функції сортування |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1.

Зображення, що містить схема, текст, Креслення, знімок екрана

Автоматично згенерований опис

Рисунок 3.1 – Загальна архітектура системи

Система складається з наступних елементів:

– графічний інтерфейс;

– localStorage;

Графічний інтерфейс необхідний для взаємодії з користувачем. HTTP запит надходить до серверної частини, де оброблюється і повертається відповідь. На серверній частині виконується основна логіка системи. Дані, отриманні з графічного інтерфейсу валідуються, конвертуються. Також, серверна частина формує запит до бази даних та оброблює відповідь і передає її до графічного інтерфейсу.

На контролер надходять дані з графічного інтерфейсу. З контролеру, дані формуються в сервісі для запиту в репозиторій. З репозиторію дані надсилаються до бази даних і зберігаються. Також в контролері формується вид, тобто об’єкт і його ім’я для відображення на графічному інтерфейсі.

# 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

## 4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

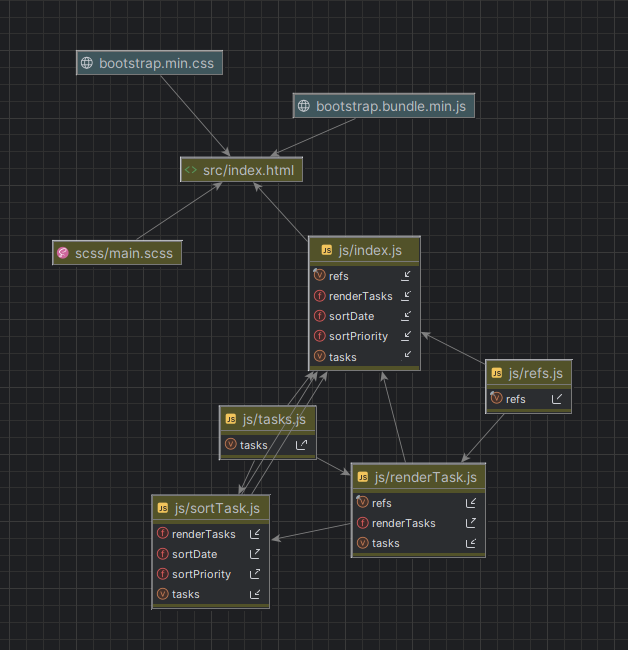


Рисунок 4.1 – Загальна структура проекту

Проект складається з веб-ресурсів, бібліотек, та вихідного коду, який в свою чергу можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти. Проект складається з веб-ресурсів, бібліотек та вихідного

Веб-ресурси включають HTML файли, які забезпечують структуру веб-сторінок (index.html), Bootstrap, який забезпечує стилізацію веб-сторінок, та JavaScript файли, які забезпечують інтерактивність та динамічну поведінку веб-сторінок (index.js, renderTask.js, tasks.js, refs.js, sortTask.js).

Бібліотеки включають сторонні бібліотеки, необхідні для роботи проекту. Такі як Вootstrap, який забезпечує стилізацію веб-сторінки.

## 4.2 Компоненти рівня доступу до даних

Зображення, що містить текст, знімок екрана, Мультимедійне програмне забезпечення, Графічний редактор

Автоматично згенерований опис

Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

## 4.3 Компоненти рівня бізнес-логіки

Зображення, що містить текст, знімок екрана, Шрифт, схема

Автоматично згенерований опис

Рисунок 4.3 – Компоненти рівня бізнес-логіки

## 4.4 Компоненти рівня інтерфейсу користувача

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

## Рисунок 4.4 – Компоненти рівня інтерфейсу користувача

# ВИСНОВКИ

Під час розробки планувальника задач було проведено аналіз та визначено ключові вимоги до системи, як функціональні, так і нефункціональні. Що допомогло окреслити поведінку системи і закласти основу для подальшої розробки.

Основним вибором мови програмування став JavaScript, через його популярністю та простоту. Вибір середовища розробки це Visual Studio Code (VS Code) завдяки його функціям, багатству розширень, які спрощують процес написання коду та і вже всі звикли до нього. Використання локального сховища як системи управління даними обрано через його простоту в реалізації та відсутність необхідності в додаткових серверних ресурсах, що підвищує автономність і швидкодію додатка.

Зважаючи на вимогу мати зручний і зрозумілий графічний інтерфейс, були використані HTML, CSS і Bootstrap. HTML забезпечує структуру веб-сторінок, CSS надає стильове оформлення, а Bootstrap, зі своїми готовими компонентами і гнучкими класами, дозволяє швидко та ефективно створювати адаптивні і круті інтерфейси користувача.

У розробці системи було створено кілька сценаріїв використання, що охоплюють основний функціонал планувальника задач: додавання, редагування, видалення та позначення виконаних задач. Ці функції є критично важливими для будь-якого планувальника задач і забезпечують користувачам необхідні інструменти для ефективного управління своїми справами.

Основні переваги створеної системи полягають в її простоті налаштування функціоналу. Відкрита архітектура додатка дозволяє легко додавати нові можливості та адаптувати систему під конкретні потреби користувачів. Це робить її гнучкою та масштабованою, що є важливим аспектом для майбутнього розвитку.

Підсумовуючи, створена система є потужним інструментом для управління задачами з простим у налаштуванні інтерфейсом та великою кількістю можливостей для подальшого розвитку. Її гнучкість, простота використання і можливість швидкого розширення роблять її привабливим вибором як для індивідуальних користувачів, так і для невеликих команд, які шукають ефективне рішення для організації своєї роботи.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
2. Приклади діаграм: <https://www.jetbrains.com>
3. Бібліотека Bootstrap: <https://getbootstrap.com>

# ДОДАТОК А Лістінг програми

index.js

import { refs } from './refs';

import { renderTasks } from './renderTask';

import { tasks } from './tasks';

import { sortDate } from './sortTask';

import { sortPriority } from './sortTask';

function handleSubmin(e) {

  e.preventDefault();

  const form = e.target;

  const task = {

    id: Date.now(),

    title: form.elements.taskTitle.value,

    date: form.elements.taskDate.value,

    priority: form.elements.taskPriority.value,

    completed: false,

  };

  tasks.push(task);

  localStorage.setItem('tasks', JSON.stringify(tasks));

  form.elements.taskTitle.value = '';

  form.elements.taskDate.value = '';

  form.elements.taskPriority.value = 'high';

  renderTasks();

}

refs.taskForm.addEventListener('submit', handleSubmin);

refs.sortByDate.addEventListener('click', sortDate);

refs.sortByPriority.addEventListener('click', sortPriority);

renderTasks();

refs.js

export const refs = {

  taskForm: document.querySelector('.taskForm'),

  taskList: document.querySelector('.taskList'),

  sortByPriority: document.getElementById('sortByPriority'),

  sortByDate: document.getElementById('sortByDate'),

};

renderTask.js

import { refs } from './refs';

import { tasks } from './tasks';

export function renderTasks() {

  refs.taskList.innerHTML = '';

  tasks.forEach(task => {

    const taskItem = document.createElement('div');

    taskItem.classList.add('taskItem', 'card', 'mb-3');

    taskItem.innerHTML = `

      <div class="card-body d-flex justify-content-between align-items-center">

        <div class="form-check flex-grow-1">

          <input type="checkbox" ${task.completed ? 'checked' : ''} data-id="${

      task.id

    }" class="form-check-input taskComplete" />

          <label class="form-check-label ${

            task.completed ? 'text-decoration-line-through' : ''

          } w-100">

            <ul class="list-group list-group-horizontal">

              <li class="list-group-item flex-fill text-center">

                <div>Задача</div>

                <div>${task.title}</div>

              </li>

              <li class="list-group-item flex-fill text-center">

                <div>Дата</div>

                <div>${task.date}</div>

              </li>

              <li class="list-group-item flex-fill text-center">

                <div>Пріоритет</div>

                <div>${task.priority}</div>

              </li>

            </ul>

          </label>

        </div>

        <div class="d-flex ms-3"> <!-- Відступ зліва -->

          <button data-id="${

            task.id

          }" class="btn btn-warning btn-sm taskEdit me-2">Редагувати</button>

          <button data-id="${

            task.id

          }" class="btn btn-danger btn-sm taskDelete">Видалити</button>

        </div>

      </div>

    `;

    refs.taskList.appendChild(taskItem);

  });

  document.querySelectorAll('.taskComplete').forEach(checkbox => {

    checkbox.addEventListener('change', toggleComplete);

  });

  document.querySelectorAll('.taskEdit').forEach(button => {

    button.addEventListener('click', editTask);

  });

  document.querySelectorAll('.taskDelete').forEach(button => {

    button.addEventListener('click', deleteTask);

  });

}

function editTask(e) {

  const taskId = e.target.getAttribute('data-id');

  const task = tasks.find(t => t.id == taskId);

  document.getElementById('taskTitle').value = task.title;

  document.getElementById('taskDate').value = task.date;

  document.getElementById('taskPriority').value = task.priority;

  renderTasks();

  const index = tasks.findIndex(t => t.id == taskId);

  if (index !== -1) {

    tasks.splice(index, 1);

  }

  localStorage.setItem('tasks', JSON.stringify(tasks));

}

function toggleComplete(e) {

  const taskId = e.target.getAttribute('data-id');

  const task = tasks.find(t => t.id == taskId);

  task.completed = !task.completed;

  localStorage.setItem('tasks', JSON.stringify(tasks));

  renderTasks();

}

function deleteTask(e) {

  const taskId = e.target.getAttribute('data-id');

  const index = tasks.findIndex(t => t.id == taskId);

  if (index !== -1) {

    tasks.splice(index, 1);

  }

  localStorage.setItem('tasks', JSON.stringify(tasks));

  renderTasks();

}

sortTask.js

import { renderTasks } from './renderTask';

import { tasks } from './tasks';

export function sortPriority() {

  const priorityOrder = { high: 1, medium: 2, low: 3 };

  tasks.sort((a, b) => priorityOrder[a.priority] - priorityOrder[b.priority]);

  renderTasks();

}

export function sortDate() {

  tasks.sort((a, b) => new Date(a.date) - new Date(b.date));

  renderTasks();

}

task.js

export let tasks = [];

const savedTasks = localStorage.getItem('tasks');

if (savedTasks) {

  tasks = JSON.parse(savedTasks);

} else {

  tasks = [];

}

index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link

      href="https://stackpath.bootstrapcdn.com/bootstrap/5.1.3/css/bootstrap.min.css"

      rel="stylesheet"

    />

    <link rel="stylesheet" href="./scss/main.scss" />

    <title>Планувальник задач</title>

  </head>

  <body>

    <div class="container my-4">

      <h1 class="text-center mb-4">Планувальник задач</h1>

      <form action="" class="taskForm mb-4">

        <div class="mb-3">

          <label for="taskTitle" class="form-label">Назва задачі:</label>

          <input

            type="text"

            id="taskTitle"

            name="taskTitle"

            class="form-control"

            required

          />

        </div>

        <div class="mb-3">

          <label for="taskDate" class="form-label">Дата:</label>

          <input

            type="date"

            id="taskDate"

            name="taskDate"

            class="form-control"

            required

          />

        </div>

        <div class="mb-3">

          <label for="taskPriority" class="form-label">Пріоритет:</label>

          <select id="taskPriority" name="taskPriority" class="form-select">

            <option value="high">Високий</option>

            <option value="medium">Середній</option>

            <option value="low">Низький</option>

          </select>

        </div>

        <button class="btn btn-primary" type="submit">Додати задачу</button>

      </form>

      <h3 class="mb-3">Сортувати за:</h3>

      <button id="sortByDate" class="btn btn-outline-primary me-2 mb-3">

        Датою

      </button>

      <button id="sortByPriority" class="btn btn-outline-primary mb-3">

        Пріоритетом

      </button>

      <h2>Список задач</h2>

      <div class="taskList"></div>

    </div>

    <script src="./js/index.js" type="module"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.1.3/js/bootstrap.bundle.min.js"></script>

  </body>

</html>