Київський національний університет імені Тараса Шевченка Факультет комп’ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Лабораторна робота №2

Інструментальні засоби розробки програмного забезпечення

Виконала студентка 2-го курсу
Групи ІПС-22
Чергінець Юлія Дмитрівна

Київ – 2025

Тема:

Документація коду (Doxygen/JavaDoc/…) і автоматична публікація через GitHub Actions та GitHub Pages.

Мета:
Навчитися:

- документувати код стандартними коментарями (Doxygen, JavaDoc тощо);
- автоматично генерувати HTML-документацію в CI/CD;
- публікувати її на GitHub Pages після кожного коміту.

Результат:

1. **Посилання на репозиторій GitHub.**

https://github.com/raychiikk/lab1-instrumental

2. **Посилання на сторінку GitHub Pages із документацією.**

https://raychiikk.github.io/lab1-instrumental/

3. **Скріншоти або опис основних кроків (коментарі, CI, сторінка).**

Опис основних кроків

1. Документування коду (Коментарі JSDoc)

Першим кроком було додавання коментарів до існуючої кодової бази, щоб забезпечити можливість автоматичної генерації документації.

1. **Вибір інструменту:** Для проєкту на JavaScript було обрано **JSDoc**, оскільки це стандартний анотатор коду, синтаксично схожий на JavaDoc.
2. **Встановлення:** JSDoc було додано до проєкту як залежність для розробки (`npm install -D jsdoc`).
3. **Коментування:** До основних файлів логіки (`src/utils/todoUtils.js` та `src/hooks/useTodos.js`)

було додано JSDoc-блоки (`/** ... */`) перед кожною функцією та хуком.
    - Використано ключові теги, як того вимагало завдання:
        - `@brief`: Для короткого опису функції.
        - `@param`: Для опису кожного аргументу, його типу та призначення.
        - `@return`: Для опису значення, що повертається.
        - `@throws`: Для опису помилок, які може викинути функція (наприклад, у `validateTodoText`).
        - `@example`: Для надання прикладу використання коду.
        - `@typedef`: Для визначення кастомних типів даних (наприклад, `Todo` та `UseTodosReturn`), що значно покращило читабельність документації для складних об'єктів.
4. **Конфігурація:** Створено файл `jsdoc.conf.json` у корені проєкту. Цей файл налаштовує JSDoc, вказуючи йому:
    - Вихідні папки для сканування (`src/hooks`, `src/utils`).
    - Папку для виводу згенерованого HTML (`docs/`).
    - Патерни файлів, які слід ігнорувати (наприклад, `*.test.js`).
5. **Локальна перевірка:** За допомогою команди `npx jsdoc -c jsdoc.conf.json` було згенеровано документацію локально. Файл `docs/index.html` було відкрито в браузері для перевірки коректності генерації.
6. **Ігнорування:** Згенерована папка `docs/` була додана до файлу `.gitignore`, щоб уникнути її завантаження до репозиторію.

2. Налаштування CI/CD (GitHub Actions)

Для автоматизації процесу генерації та публікації документації було налаштовано воркфлоу (workflow) за допомогою GitHub Actions.

1. **Створення воркфлоу:** У репозиторії було створено файл `.github/workflows/docs.yml`.
2. **Тригери:** Воркфлоу налаштовано на автоматичний запуск при кожному `push` у гілки `main` та `feature/docs-ci`.

3. **Задачі (Jobs):** Воркфлоу складається з двох послідовних задач:
    - **`build` (Збірка):**
        1. Завантажує код репозиторію (`actions/checkout`).
        2. Встановлює середовище Node.js (`actions/setup-node`).
        3. Встановлює всі залежності проєкту (`npm ci`), включаючи `jsdoc`.
        4. Виконує ту саму команду, що й локально (`npx jsdoc -c jsdoc.conf.json`), для генерації HTML-документації у папку `docs/`.
        5. Запаковує згенеровану папку `docs/` як артефакт (`actions/upload-pages-artifact`) для передачі наступній задачі.
    - **`deploy` (Розгортання):**
        1. Ця задача залежить від успішного завершення `build`.
        2. Вона використовує стандартний екшн `actions/deploy-pages` для взяття артефакту та його публікації на сервісі GitHub Pages.
4. **Дозволи (Permissions):** Файлу воркфлоу було надано необхідні дозволи (`pages: write`, `id-token: write`) для того, щоб він мав право записувати дані на GitHub Pages.

3. Публікація (GitHub Pages)

Фінальним кроком було налаштування самого репозиторію GitHub для прийому та відображення згенерованої сторінки.

1. **Налаштування джерела:** У налаштуваннях репозиторію (`Settings` > `Pages`) у розділі "Build and deployment" джерело (`Source`) було змінено з "Deploy from a branch" на **"GitHub Actions"**. Це повідомило GitHub, що за оновлення сайту тепер відповідає CI/CD, а не конкретна гілка.
2. **Налаштування середовища (Environment):** Під час першого запуску виникла помилка безпеки. Для її вирішення:
    - Перейшли в `Settings` > `Environments`.
    - Обрали середовище `github-pages`.

    - У правилах "Deployment branches" було додано гілку `feature/docs-ci` (або обрано "All branches") до списку дозволених. Це необхідно, щоб feature-гілки могли публікувати документацію для тестування.
3. **Перевірка:** Після внесення цих налаштувань воркфлоу було перезапущено (`Re-run jobs`). Обидві задачі, `build` та `deploy`, успішно завершились.
4. **Результат:** У розділі `Settings` > `Pages` з'явилося активне посилання на опублікований сайт, де відображається згенерована JSDoc-документація.

4. Короткий висновок про набуті навички.

Висновок:

Під час виконання цієї лабораторної роботи я здобула практичні навички у:

- **Документуванні коду:** Навчилась застосовувати стандарт JSDoc для детального опису функцій, їхніх параметрів (`@param`), значень, що повертаються (`@return`), та кастомних типів даних (`@typedef`).
- **Автоматизації (CI/CD):** Здобула вміння створювати "з нуля" воркфлоу (workflow) в GitHub Actions для автоматизації рутинних завдань, зокрема генерації документації.
- **Публікації (Деплої):** Навчилась налаштовувати GitHub Pages для автоматичної публікації статичних веб-сайтів (документації) безпосередньо через CI/CD.
- **Інтеграції та вирішенні проблем:** Здобула досвід у поєднанні різних сервісів GitHub (Actions, Pages, Environments) та у вирішенні реальних проблем, пов'язаних з правами доступу та правилами захисту гілок (Environment protection rules).

У результаті, я вмію налаштовувати повний цикл автоматичної доставки документації: від написання коментаря в коді до його появи на опублікованому сайті.