**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

**Курсова робота**

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

**РОЗРОБКА OSS-СЕРВІСУ веб-ПЛАТФОРМИ ДЛЯ ОРГАНІВ СТУДЕНТСЬКОГО САМОВРЯДУВАННЯ (ОСС) КНУ ІМ. ТАРАСА ШЕВЧЕНКА**

|  |  |
| --- | --- |
| Виконав студент 3-го курсу  Артем ВЕРБИЦЬКИЙ | (підпис) |
| Науковий керівник:  асистент кафедри інтелектуальних програмних систем, кандидат фіз.-мат. наук  Костянтин ЖЕРЕБ | (підпис) |

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

|  |  |
| --- | --- |
| Студент | (підпис) |

Київ — 2025

**Реферат**

Обсяг роботи [кількість] сторінок, [кількість] ілюстрацій, [кількість] таблиць, [кількість] джерел посилання.

API, DOCKER, NESTJS, POSTGRESQL, TYPESCRIPT, WEB-ПЛАТФОРМА, МІКРОСЕРВІСНА АРХІТЕКТУРА, РЕЛЯЦІЙНА БАЗА ДАНИХ, СТУДЕНТСЬКЕ САМОВРЯДУВАННЯ

Об'єктом роботи є процес автоматизації діяльності органів студентського самоврядування КНУ імені Тараса Шевченка. Предметом роботи є серверна частина веб-платформи (OSS-сервіс) для органів студентського самоврядування.

Метою курсової роботи є розробка високопродуктивної та масштабованої backend-частини веб-платформи для забезпечення функціонування органів студентського самоврядування КНУ імені Тараса Шевченка та інформування студентів щодо оних.

Методи розроблення: об'єктно-орієнтоване програмування, мікросервісна архітектура, RESTful API, асинхронна взаємодія між сервісами. Інструменти розроблення: TypeScript, Nest.js, Sequelize, PostgreSQL, Docker.

Результати роботи: спроектовано та реалізовано частину OSS-сервісу веб-платформи для органів студентського самоврядування, що включає RESTful API для управління користувачами-членами ОСС. Реалізовано взаємодію з іншими мікросервісами системи через брокер повідомлень RabbitMQ.

За методами розробки та інструментальними засобами робота виконувалася з урахуванням сучасних практик розробки веб-додатків.

Розроблений OSS-сервіс може бути інтегрований з іншими компонентами веб-платформи та застосований для автоматизації роботи органів студентського самоврядування КНУ імені Тараса Шевченка.

**Зміст**

**Скорочення та умовні позначення**

API – Application Programming Interface, прикладний програмний інтерфейс;

ОСС – органи студентського самоврядування;

REST – Representational State Transfer, передача репрезентативного стану;

CRUD – Create, Read, Update, Delete, операції створення, читання, оновлення, видалення;

ТЗ – технічне завдання;

БД – база даних;

ООП – об'єктно-орієнтоване програмування;

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту;

ORM – Object-Relation Mapping;

DTO – Data Transfer Object (об'єкт передачі даних);

SQL – Structured Query Language (мова структурованих запитів);

JWT – JSON Web Token, токен веб-автентифікації.

**Вступ**

**Оцінка сучасного стану об'єкта розробки.** Органи студентського самоврядування (ОСС) є важливою складовою університетського життя, що забезпечує представництво та захист прав та інтересів студентів, їх участь в управлінні вищим навчальним закладом. У сучасних умовах розвитку інформаційного суспільства ефективність роботи ОСС значною мірою залежить від рівня інформатизації їхньої діяльності.

На даний момент у КНУ імені Тараса Шевченка відсутнє єдине інформаційне середовище, яке б забезпечувало ефективну комунікацію між органами студентського самоврядування та студентами, що ускладнює процеси організації заходів, інформування, збору та обробки звернень студентів.

**Актуальність роботи та підстави для її виконання.** Створення веб-платформи для органів студентського самоврядування КНУ імені Тараса Шевченка є актуальним завданням, оскільки дозволить оптимізувати процеси взаємодії студентів з ОСС, підвищити прозорість діяльності органів самоврядування, спростити процеси організації заходів та збору зворотного зв'язку.

В умовах зростаючої діджиталізації освітнього процесу та студентського життя, розробка сучасної веб-платформи з використанням передових технологій є необхідним кроком для забезпечення ефективної роботи ОСС.

**Мета й завдання роботи.** Метою роботи є створення високопродуктивної та масштабованої backend-частини веб-платформи (OSS-сервісу) для забезпечення функціонування ОСС КНУ імені Тараса Шевченка.

Для досягнення цієї мети поставлено такі завдання:

* Дослідити існуючі рішення та підходи до розробки веб-платформ для студентського самоврядування.
* Розробити архітектуру backend-частини веб-платформи з використанням мікросервісного підходу.
* Спроектувати та реалізувати API для взаємодії з frontend-частиною.
* Розробити підсистему автентифікації та авторизації користувачів.
* Реалізувати механізми асинхронної взаємодії між сервісами.
* Забезпечити контейнеризацію розробленого рішення за допомогою Docker.

**Об'єкт, методи й засоби розроблення.** Об'єктом розроблення є backend-частина веб-платформи для ОСС КНУ імені Тараса Шевченка. Предметом розроблення є OSS-сервіс для забезпечення діяльності ОСС.

Методи розроблення включають:

* Об'єктно-орієнтоване програмування для структурування коду та забезпечення його модульності.
* Мікросервісну архітектуру для розділення функціональності на незалежні сервіси.
* REST API для забезпечення комунікації між клієнтською та серверною частинами.
* Асинхронну взаємодію між сервісами для підвищення продуктивності системи.

Технологічний стек проекту включає:

* TypeScript як основну мову програмування для забезпечення типобезпеки та кращої підтримки коду.
* Nest.js як фреймворк для розробки серверної частини, що забезпечує модульність та масштабованість.
* Sequelize для роботи з реляційною базою даних.
* Docker для контейнеризації та спрощення розгортання.

**Можливі сфери застосування.** Розроблений сервіс є частиною веб-платформи, яка може бути використана для:

* Організації роботи ОСС КНУ імені Тараса Шевченка.
* Підвищення ефективності комунікації між студентами та ОСС.
* Автоматизації процесів організації студентських заходів та ініціатив.
* Збору та обробки звернень студентів.
* Підвищення прозорості діяльності ОСС

**1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ**

* 1. **Аналіз діяльності органів студентського самоврядування**

Студентське самоврядування є невід'ємною частиною громадського самоврядування у закладах вищої освіти. Згідно з Законом України "Про вищу освіту", студентське самоврядування – це право і можливість студентів вирішувати питання навчання і побуту, захисту прав та інтересів студентів, а також брати участь в управлінні закладом вищої освіти [1].

У Київському національному університеті імені Тараса Шевченка органи студентського самоврядування (ОСС) функціонують як на рівні структурних підрозділів, так і на загальноуніверситетському рівні. Структура ОСС в КНУ імені Тараса Шевченка представлена на рисунку 1.1.

A diagram with many colorful squares

AI-generated content may be incorrect.

Рис. 1.1

**1.1.1 Структура ОСС**

В КНУ імені Тараса Шевченка система студентського самоврядування має таку структуру:

1. Представницькі органи:
   1. Конференція студентів Університету (КСУ);
   2. Конференція студентів структурного підрозділу (КССП);
   3. Конференція студентів-мешканців Студмістечка (КСМС);
   4. Конференція студентів-мешканців гуртожитку (КСМГ);
2. Виконавчі органи:
   1. Студентський парламент університету (СПУ);
   2. Студентський парламент структурного підрозділу (СПСП);
   3. Студентська рада Студмістечка (СРС);
   4. Студентська рада гуртожитку (СРГ);
3. Незалежні органи:
   1. Студентський омбудсмен – обирається КСУ;
   2. Центральна виборча комісія студентів (ЦВКс) – обирається КСУ, КСМС, СПУ та СРС;
   3. Контрольно-ревізійна комісія студентів (КРК) – обирається КСУ, КСМС, СПУ та СРС.

Відповідно до схеми, представницька гілка включає конференції студентів різних рівнів, а виконавча гілка представлена студентськими парламентами та радами. Незалежні органи забезпечують контроль та дотримання прав студентів у системі самоврядування.

**1.1.2 Основні задачі та процеси ОСС**

Аналіз діяльності ОСС КНУ імені Тараса Шевченка дозволив виявити основні задачі та процеси, які потребують автоматизації:

* Інформування студентів – поширення інформації про діяльність ОСС, важливі події та можливості;
* Проведення засідань – організація та проведення засідань ОСС, формування порядку денного, документування рішень;
* Організація заходів – планування, організація та проведення різноманітних заходів для студентів;
* Розгляд звернень – прийом, обробка та розгляд звернень студентів з різних питань;
* Управління проектами – планування, реалізація та моніторинг студентських проектів та ініціатив;
* Звітність – підготовка та оприлюднення звітів про діяльність ОСС.

**1.1.3 Проблеми організації роботи ОСС**

Аналіз діяльності ОСС КНУ імені Тараса Шевченка виявив низку проблем, які негативно впливають на ефективність їх роботи:

* Відсутність єдиної інформаційної системи – інформація про діяльність ОСС розпорошена між різними платформами та ресурсами;
* Складність в організації заходів – організація заходів потребує значних витрат часу та ресурсів через відсутність автоматизованих інструментів;
* Проблеми з обробкою звернень – відсутня єдина система для обробки звернень студентів, що призводить до втрати або затримки в розгляді деяких звернень;
* Відсутність прозорості – студенти не мають достатнього доступу до інформації про діяльність ОСС, прийняті рішення та використання коштів;
* Складність у підготовці звітності – підготовка звітів про діяльність ОСС є трудомісткою через відсутність єдиної системи збору та аналізу даних.

Ці проблеми свідчать про необхідність створення єдиної web-платформи, яка б забезпечила автоматизацію основних процесів діяльності ОСС та покращила комунікацію між різними структурними підрозділами та студентами.

* 1. **Огляд існуючих рішень для автоматизації діяльності ОСС**

Для визначення оптимального підходу до розробки web-платформи для органів студентського самоврядування КНУ імені Тараса Шевченка було проведено аналіз існуючих рішень, які використовуються для автоматизації діяльності ОСС в інших закладах вищої освіти та організаціях.

**1.2.1 Наявні платформи для студентського самоврядування**

* Студентська рада КПІ ім. Ігоря Сікорського – web-портал, що забезпечує інформування студентів про діяльність ОСС, анонси заходів та новини [3]. Портал не надає функціональності для автоматизації внутрішніх процесів ОСС;
* MyClassBoard – інтегрована система управління освітнім процесом, яка включає модулі для комунікації, обліку відвідуваності, управління заходами та академічною успішністю [4]. Система має функціонал для шкіл та коледжів, але має обмежену адаптацію до потреб студентського самоврядування в українських ЗВО;
* Електронний кабінет студента – система, впроваджена в деяких українських ЗВО, яка забезпечує доступ студентів до навчальних матеріалів, розкладу занять та іншої інформації. Однак, ця система не включає функціональність для роботи ОСС;
* Google Workspace for Education – набір інструментів Google, який використовується деякими ОСС для організації внутрішньої роботи [5]. Проте, ці інструменти не інтегровані в єдину систему та не покривають усі потреби ОСС;
* Microsoft Teams – платформа для командної роботи, яка використовується деякими ОСС для комунікації та спільної роботи над проектами [6]. Однак, як і Google Workspace, ця платформа не надає спеціалізованих інструментів для роботи ОСС.

Порівняльний аналіз функціональності існуючих рішень представлено в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз функціональності існуючих рішень

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Функціо-нальність** | **Студентська рада КПІ** | **MyClass-Board** | **Е-кабінет студента** | **Google Workspace** | **Microsoft Teams** |
| Інформува-ння студентів | + | + | + | +/- | +/- |
| Організація заходів | - | + | - | +/- | +/- |
| Обробка звернень | - | + | - | +/- | +/- |
| Управління проектами | - | +/- | - | +/- | + |
| Фінансова звітність | - | +/- | - | +/- | - |
| Інтеграція з системами університету | +/- | +/- | + | - | - |

Примітка. "+" – функціональність повністю реалізована, "+/-" – функціональність частково реалізована, "-" – функціональність відсутня.

**1.2.2 Аналіз технологічних підходів до розробки веб-платформ**

Окрім аналізу існуючих рішень, було досліджено різні технологічні підходи до розробки веб-платформ, які можуть бути використані для створення системи для ОСС КНУ імені Тараса Шевченка.

* Монолітна архітектура – традиційний підхід, при якому весь функціонал реалізований в рамках єдиного додатку. Перевагами є простота розробки та розгортання, недоліками – складність масштабування та підтримки при зростанні системи;
* Мікросервісна архітектура – підхід, при якому система розбивається на набір незалежних сервісів, кожен з яких відповідає за окрему функціональність. Перевагами є гнучкість, масштабованість та можливість незалежного розгортання сервісів, недоліками – складність у розробці та тестуванні;
* Серверний рендеринг (SSR) – підхід, при якому HTML-сторінки генеруються на сервері. Перевагами є краща SEO-оптимізація та швидше початкове завантаження, недоліками – більше навантаження на сервер;
* Клієнтський рендеринг (CSR) – підхід, при якому HTML-сторінки генеруються на клієнті з використанням JavaScript. Перевагами є краща інтерактивність та менше навантаження на сервер, недоліками – гірша SEO-оптимізація та повільніше початкове завантаження;
* Progressive Web Applications (PWA) – підхід, при якому веб-додаток має функціональність та поведінку, притаманні нативним мобільним додаткам. Перевагами є можливість роботи офлайн та покращений користувацький досвід, недоліками – складність у розробці та тестуванні.

Порівняльний аналіз технологічних підходів представлено в таблиці 1.3.

Таблиця 1.3 – Порівняльний аналіз технологічних підходів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Критерій** | **Монолітна архітектура** | **Мікро-**  **сервісна архітектура** | **SSR** | **CSR** | **PWA** |
| Масштабованість | Низька | Висока | Середня | Середня | Середня |
| Швидкість розробки | Висока | Низька | Середня | Висока | Низька |
| Надійність | Середня | Висока | Висока | Середня | Висока |
| SEO-оптимізація | - | - | Висока | Низька | Середня |
| Користувацький досвід | - | - | Середній | Високий | Високий |
| Можливість роботи офлайн | - | - | Низька | Низька | Висока |
| Складність підтримки | Висока | Середня | Середня | Середня | Висока |

Проведений аналіз показав, що для розробки веб-платформи для ОСС КНУ імені Тараса Шевченка найбільш доцільним є використання мікросервісної архітектури з елементами PWA, що забезпечить необхідну гнучкість, масштабованість та покращений користувацький досвід.

* 1. **Висновки до розділу**

На основі проведеного аналізу предметної області та існуючих рішень для автоматизації діяльності органів студентського самоврядування можна зробити наступні висновки:

* Діяльність органів студентського самоврядування в КНУ імені Тараса Шевченка характеризується складною організаційною структурою та широким спектром функцій, що потребують ефективної автоматизації;
* Існуючі рішення для автоматизації діяльності ОСС не задовольняють усіх потреб та не враховують специфіку роботи ОСС в КНУ імені Тараса Шевченка;
* Для розробки веб-платформи для ОСС найбільш доцільним є використання мікросервісної архітектури, яка забезпечить необхідну гнучкість та масштабованість системи;
* Розробка веб-платформи для органів студентського самоврядування КНУ імені Тараса Шевченка дозволить вирішити виявлені проблеми в організації роботи ОСС та підвищити ефективність їх діяльності.

Отже, проведений аналіз підтверджує необхідність та доцільність розробки спеціалізованої веб-платформи для органів студентського самоврядування КНУ імені Тараса Шевченка з використанням мікросервісного підходу до розробки.

**2 АРХІТЕКТУРА ТА ТЕХНОЛОГІЇ РОЗРОБКИ**

* 1. **Обґрунтування вибору архітектури системи**

Вибір архітектури є одним із ключових етапів розробки програмного забезпечення, який суттєво впливає на масштабованість, надійність, підтримуваність та продуктивність системи. При розробці веб-платформи для органів студентського самоврядування КНУ ім. Тараса Шевченка необхідно врахувати ряд факторів, що обумовлюють архітектурні рішення:

* **Різноманітність функціональних вимог**. Система повинна забезпечувати різноманітні функціональні можливості, включаючи управління користувачами, заходами, новинами, документами, голосуваннями, що призводить до складної доменної моделі;
* **Необхідність масштабування**. З ростом кількості користувачів та функціональних можливостей система повинна зберігати високу продуктивність та мати можливість горизонтального масштабування;
* **Розподіл відповідальності**. Різні аспекти системи (автентифікація, управління користувачами, управління заходами тощо) повинні бути логічно розділені для забезпечення чіткого розподілу відповідальності та можливості незалежного розвитку;
* **Стабільність і відмовостійкість**. Система повинна продовжувати працювати навіть у випадку відмови окремих компонентів;
* **Технологічна гетерогенність**. Різні компоненти системи можуть використовувати різні технології та мови програмування в залежності від їх специфічних вимог.

Враховуючи ці фактори, для розробки веб-платформи для ОСС було обрано мікросервісну архітектуру. Мікросервісна архітектура дозволяє розбити систему на незалежні, слабо пов'язані сервіси, кожен з яких відповідає за конкретну функціональну область. У контексті нашої системи ключовими мікросервісами є:

* Auth-сервіс – відповідає за автентифікацію та авторизацію користувачів;
* Oss-сервіс – основний сервіс для управління діяльністю органів студентського самоврядування;
* User-сервіс – відповідає за управління користувачами та їх профілями;
* Notification-сервіс – відповідає за надсилання сповіщень користувачам;
* Gateway-сервіс – виступає єдиною точкою входу для клієнтських запитів і маршрутизує їх до відповідних сервісів.

Така архітектура дозволяє забезпечити незалежний розвиток та розгортання кожного сервісу, горизонтальне масштабування, ізоляцію помилок, а також можливість використання різних технологій для кожного сервісу в залежності від його специфічних вимог.

Для взаємодії між мікросервісами було обрано як синхронну комунікацію через REST API (для безпосередніх запитів-відповідей), так і асинхронну комунікацію через брокер повідомлень RabbitMQ (для подій та довготривалих процесів). Це дозволяє забезпечити гнучкість у взаємодії між сервісами та підвищити надійність системи в цілому.

Окрім архітектури на рівні сервісів, для Oss-сервісу було застосовано підхід чистої архітектури (Clean Architecture), який забезпечує розділення рівнів відповідальності в рамках одного сервісу:

* Рівень представлення – відповідає за взаємодію з клієнтами через API;
* Рівень прикладної логіки – містить бізнес-логіку застосунку;
* Рівень доменної логіки – містить бізнес-сутності та правила предметної області;
* Рівень інфраструктури – відповідає за взаємодію з зовнішніми системами (базами даних, іншими сервісами).

Такий підхід дозволяє забезпечити чіткий розподіл відповідальності, спрощує тестування та підтримку коду, а також зменшує зв'язаність між різними частинами системи.

**2.2 Мікросервісна архітектура та її переваги**

Мікросервісна архітектура – це підхід до розробки програмного забезпечення, при якому додаток складається з набору невеликих, автономних сервісів, кожен з яких відповідає за конкретну функціональну область та взаємодіє з іншими сервісами через чітко визначені API [13]. На відміну від монолітної архітектури, де всі функціональні компоненти тісно інтегровані в єдиному застосунку, мікросервісна архітектура розділяє відповідальність між незалежними сервісами.

Ключові характеристики мікросервісної архітектури, що були враховані при розробці веб-платформи для ОСС, включають:

* **Декомпозиція за бізнес-можливостями**. Кожен мікросервіс відповідає за конкретну бізнес-можливість або функціональну область. Наприклад, Oss-сервіс відповідає за основні функції управління діяльністю органів студентського самоврядування, а Auth-сервіс – за автентифікацію та авторизацію користувачів;
* **Автономність**. Кожен мікросервіс може бути розроблений, розгорнутий та масштабований незалежно від інших сервісів. Це дозволяє забезпечити гнучкість розвитку системи, де кожен компонент може еволюціонувати з власною швидкістю;
* **Незалежність даних**. Кожен мікросервіс має власне сховище даних, яке відповідає його специфічним вимогам. Це забезпечує слабку зв'язаність між сервісами та дозволяє використовувати різні типи баз даних в залежності від потреб кожного сервісу;
* **Комунікація через API**. Мікросервіси взаємодіють один з одним через чітко визначені API, що забезпечує слабку зв'язаність та інкапсуляцію внутрішньої логіки кожного сервісу.

Архітектура веб-платформи для органів студентського самоврядування, заснована на мікросервісному підході, представлена на рисунку 2.1.

A diagram of a diagram

AI-generated content may be incorrect.

Рис. 2.1 – Архітектура мікросервісної системи для веб-платформи ОСС

Використання мікросервісної архітектури для розробки веб-платформи для ОСС надає ряд важливих переваг:

* **Масштабованість**. Кожен мікросервіс може бути масштабований незалежно, що дозволяє ефективно розподіляти ресурси системи. Наприклад, під час проведення виборів до ОСС можна масштабувати сервіс голосування, не змінюючи ресурси, виділені для інших сервісів;
* **Стійкість до відмов**. Ізоляція мікросервісів один від одного забезпечує стійкість системи в цілому. Відмова одного сервісу не призводить до відмови всієї системи. Наприклад, якщо сервіс сповіщень тимчасово недоступний, користувачі все ще можуть використовувати основні функції системи;
* **Технологічна гнучкість**. Різні мікросервіси можуть використовувати різні технології та мови програмування в залежності від їх специфічних вимог. Хоча для Oss-сервісу було обрано стек на основі TypeScript і Nest.js, інші сервіси можуть використовувати інші технології, якщо це буде доцільно;
* **Паралельна розробка**. Різні команди можуть одночасно працювати над різними мікросервісами, що прискорює розробку та впровадження нових функцій;
* **Локалізація змін**. Зміни в одному мікросервісі не впливають на інші сервіси, що спрощує підтримку та розвиток системи. Наприклад, зміна в логіці обробки заходів в Oss-сервісі не вплине на роботу Auth-сервісу.

Однак мікросервісна архітектура також має свої виклики:

* **Складність управління розподіленою системою**. Необхідно враховувати аспекти розподіленої системи, такі як мережеві затримки, синхронізація даних, моніторинг стану сервісів тощо;
* **Складність транзакцій між сервісами**. Оскільки кожен мікросервіс має власне сховище даних, забезпечення узгодженості даних між сервісами вимагає додаткових рішень, таких як використання шаблону Saga або використання подієвої архітектури;
* **Операційна складність**. Для ефективного функціонування мікросервісної системи необхідно забезпечити автоматизацію розгортання, моніторингу та масштабування сервісів.

Для подолання цих викликів у веб-платформі для ОСС використовуються такі рішення:

* API Gateway для централізованого управління запитами клієнтів до мікросервісів;
* Брокер повідомлень RabbitMQ для асинхронної комунікації між сервісами;
* Контейнеризація через Docker для спрощення розгортання та масштабування;
* Моніторинг та логування для відстеження стану системи та виявлення проблем.

Таким чином, мікросервісна архітектура, обрана для веб-платформи органів студентського самоврядування, забезпечує гнучкість, масштабованість та надійність системи, а також дозволяє ефективно організувати розробку та підтримку різних компонентів платформи.

**2.2 Нефункціональні вимоги до системи**

**2.3 Загальна архітектура системи**

**2.4 Проектування бази даних**

**2.5 Висновки до розділу 2**

**3 РОЗРОБКА WEB-платформи**

**3.1 Розробка API для автентифікації та авторизації**

**3.2 Розробка API для управління користувачами**

**3.3 Розробка модулю обміну повідомленнями**

**3.4 Документування API**

**3.5 Контейнеризація та розгортання**

**3.6 Тестування розробленого рішення**

**3.7 Висновки до розділу 3**

**Висновки**

**Перелік джерел посилання**

* 1. Закон України "Про вищу освіту" [Електронний ресурс] // Відомості Верховної Ради України. — 2014. — № 37-38. — Режим доступу: <https://zakon.rada.gov.ua/laws/show/1556-18#Text>
  2. Положення про студентське самоврядування Київського національного університету імені Тараса Шевченка [Електронний ресурс]. — Режим доступу: <https://knu.ua/pdfs/official/Student-government-regulations.pdf>
  3. Студентська рада КПІ ім. Ігоря Сікорського [Електронний ресурс] – Режим доступу до ресурсу: <https://sc.kpi.ua/>
  4. MyClassboard [Електронний ресурс] – Режим доступу до ресурсу: <https://www.myclassboard.com/>
  5. Google for Education [Електронний ресурс] – Режим доступу до ресурсу: <https://edu.google.com/workspace-for-education/editions/overview/>
  6. Microsoft Teams [Електронний ресурс] – Режим доступу до ресурсу: <https://www.microsoft.com/uk-ua/microsoft-teams/group-chat-software/>
  7. TypeScript Documentation [Електронний ресурс]. — Режим доступу: <https://www.typescriptlang.org/docs/>
  8. Nest.js Documentation [Електронний ресурс]. — Режим доступу: <https://docs.nestjs.com/>
  9. Sequelize Documentation [Електронний ресурс]. — Режим доступу: <https://sequelize.org/master/>
  10. RabbitMQ Documentation [Електронний ресурс]. — Режим доступу: <https://www.rabbitmq.com/documentation.html>
  11. Docker Documentation [Електронний ресурс]. — Режим доступу: <https://docs.docker.com/>
  12. Swagger Documentation [Електронний ресурс]. — Режим доступу: <https://swagger.io/docs/>
  13. Newman, S. Building Microservices: Designing Fine-Grained Systems / S. Newman. — O'Reilly Media, 2015. — 280 p.
  14. Fowler, M. Patterns of Enterprise Application Architecture / M. Fowler. — Addison-Wesley Professional, 2002. — 560 p.
  15. Evans, E. Domain-Driven Design: Tackling Complexity in the Heart of Software / E. Evans. — Addison-Wesley Professional, 2003. — 560 p.
  16. Richardson, C. Microservices Patterns: With examples in Java / C. Richardson. — Manning Publications, 2018. — 520 p.
  17. Martin, R. Clean Architecture: A Craftsman's Guide to Software Structure and Design / R. Martin. — Prentice Hall, 2017. — 432 p.
  18. Gamma, E. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. — Addison-Wesley Professional, 1994. — 416 p.
  19. Kleppmann, M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / M. Kleppmann. — O'Reilly Media, 2017. — 616 p.