Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

ЗВІТ
з виробничої практики

Виконав:

студент 3-го курсу
Артем ВЕРБИЦЬКИЙ

Керівник практики:

асистент кафедри інтелектуальних програмних систем,
кандидат фіз.-мат. наук
Костянтин ЖЕРЕБ

Київ – 2025

Зміст

Вступ 3
Робота над проєктом “Confero” 4
Висновки 9
Перелік джерел посилання 10

Вступ

У межах виробничої практики я та моя команда розпочали розробку Web-платформи для структуризації інформації про органи студентського самоврядування (ОСС) Університету та автоматизації їхньої роботи. Наша команда складається з п’ятьох студентів, кожен із яких відповідає за певний напрям розробки. Моя роль у проєкті – створення серверної частини (backend).

Метою розробки є впорядкування роботи студентського самоврядування, спрощення документообігу, організації голосувань, комунікації між учасниками та інших ключових процесів. Ми прагнемо створити зручний, ефективний та безпечний інструмент для автоматизації управлінських завдань.

У процесі роботи ми спроєктували архітектуру системи, обрали технологічний стек, розробили серверну частину, базу даних та API для взаємодії з клієнтським інтерфейсом. Окрему увагу приділили безпеці, реалізувавши механізми автентифікації та контролю доступу.

Для покращення знань та навичок у межах проєкту я ознайомився з новими технологіями та підходами до розробки. Зокрема, я вивчив додаткові можливості фреймворку NestJS, оптимізацію баз даних та принципи побудови безпечних API.

Робота над проєктом “Confero”

Мета проєкту “Confero” – розробити платформу, яка зробить знайомство студентів з ОСС Університету легшим і більш зрозумілим, а також автоматизує певні внутрішні процеси цих органів, такі як скликання засідань, голосування, документообіг тощо. Мої завдання – це робота над серверною частиною платформи (backend).

1. Технологічний стек

У ході обговорень з командою було вирішено застосувати мікросервісну архітектуру для даної платформи, адже вона найкраще відповідає поставленим завданням. Для реалізації серверної частини було обрано такий стек:

- Nest.js – фреймворк для створення сервісної частини проєкту;
- Sequelize – ORM для роботи з базами даних;
- PosgreSQL, MongoDB – для баз даних;
- RabbitMQ –для обміну повідомленнями;
- Docker – для зручної розробки.

2. Ознайомлення з предметною областю

Перед початком розробки Web-платформи для органів студентського самоврядування було проведено детальне дослідження предметної області. Це допомогло визначити основні потреби користувачів, виявити ключові проблеми, що потребують автоматизації та сформувати вимоги до функціональності системи.

Аналіз діяльності ОСС КНУ імені Тараса Шевченка дозволив виявити основні потреби користувачів, які потребують автоматизації. До них відносяться інформування студентів про діяльність ОСС, важливі події та можливості, а також проведення засідань з формуванням порядку денного та документуванням рішень. Важливими потребами є також організація різноманітних заходів для студентів, прийом та обробка звернень з різних питань, планування й моніторинг студентських проєктів та ініціатив, підготовка та оприлюднення звітів про діяльність ОСС.

Дослідження виявило ряд недоліків, які негативно впливають на ефективність роботи ОСС. Основною проблемою є відсутність єдиної інформаційної системи, через що інформація про діяльність ОСС розпорошена між різними платформами та ресурсами. Це призводить до складності в організації заходів, які потребують

значних витрат часу та засобів через відсутність автоматизованих інструментів. Серйозною проблемою є відсутність єдиної системи для обробки звернень студентів, що призводить до втрати або затримки в розгляді деяких звернень. Студенти не мають достатнього доступу до інформації про діяльність ОСС, що знижує прозорість роботи органів самоврядування. Підготовка звітів про діяльність ОСС також є трудомісткою.

На основі виявлених потреб та проблем було сформульовано вимоги до платформи. Система повинна забезпечити створення високопродуктивної та масштабованої серверної частини Web-платформи для функціонування органів студентського самоврядування. Важливими вимогами є забезпечення ефективної комунікації між студентами та ОСС, автоматизація процесів організації студентських заходів та ініціатив. Система має впроваджувати єдину систему збору та обробки звернень студентів і підвищувати прозорість діяльності ОСС через надання доступу до інформації про діяльність органів самоврядування.

3. Структура студентського самоврядування

Органи студентського самоврядування в університетах відіграють важливу роль у представництві інтересів студентів, організації заходів, захисті прав та контролі за якістю освіти. Вони включають різні рівні:

- **Рівень Університету та Студмістечка** – органи самоврядування, що координують діяльність на рівні всього університету та студмістечка відповідно;
- **Рівень структурних підрозділів та гуртожитків** – студентські органи структурних підрозділів та гуртожитків що займаються локальними питаннями.

А також типи управління:

а) Студентський парламент;
б) Студентська рада;
в) Конференція студентів;
г) Контрольно-ревізійна комісія студентів;
д) Центральна виборча комісія студентів;
1) Окружна виборча комісія;
2) Дільнична виборча комісія.

4. Основні функції студентського самоврядування

На основі аналізу внутрішніх процесів самоврядування були визначені ключові завдання, які має підтримувати система:

- **Документообіг** – зберігання та обмін документами (протоколи, рішення, звернення);
- **Голосування та опитування** – організація виборів, прийняття рішень шляхом голосування;
- **Комунікація** – взаємодія між учасниками через оголошення;
- **Календар подій** – планування заходів, зустрічей та їх координація;
- **Аналітика та звітність** – збір статистичних даних щодо діяльності самоврядування.

5. Проблеми, що потребують вирішення

Аналіз існуючого підходу до управління студентським самоврядуванням виявив такі основні проблеми:

- **Відсутність єдиного цифрового простору** – інформація зберігається в різних джерелах (Google Docs, соціальні мережі, месенджери), що ускладнює доступ і координацію;
- **Складність документообігу** – паперові або неструктуровані електронні документи ускладнюють управління;
- **Низький рівень автоматизації** – багато процесів, таких як голосування або звітність, виконуються вручну;
- **Обмежена прозорість діяльності** – учасники самоврядування не завжди мають швидкий доступ до важливої інформації.

Отримані результати дослідження предметної області лягли в основу вимог до розробки Web-платформи. Запропонована система дозволить оптимізувати діяльність студентського самоврядування, покращити координацію та забезпечити ефективну взаємодію між усіма учасниками.

6. Розробка серверної частини проєкту (backend)

Перший етап нашої роботи – **проєктування архітектури системи**. Ми прагнули створити гнучку та масштабовану платформу, тому обрали клієнт-серверну архітектуру з розподіленими мікросервісами, представлену на рисунку 1. Серверну частину вирішили реалізувати на **NestJS**, оскільки цей фреймворк забезпечує модульність, стабільність та гарну підтримку TypeScript.

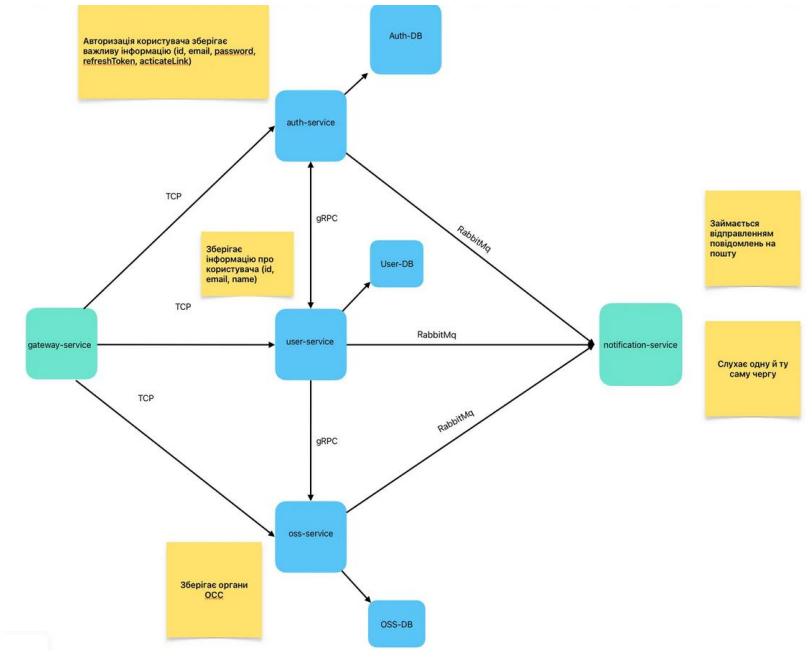

Рис. 1 – Архітектура мікросервісної системи для веб-платформи ОСС

Для роботи з базою даних (БД) ми використали **Sequelize** – ORM (Object-Relational Mapping) для **PostgreSQL**, що дозволяє ефективно керувати моделями, виконувати запити та взаємодіяти з даними на високому рівні абстракції. Це значно спрощує розробку, забезпечує захист від SQL-ін'єкцій та дає можливість міграції БД. Було відповідно розроблено базу даних oss-сервісу, схему якої наведено на рисунку 2.

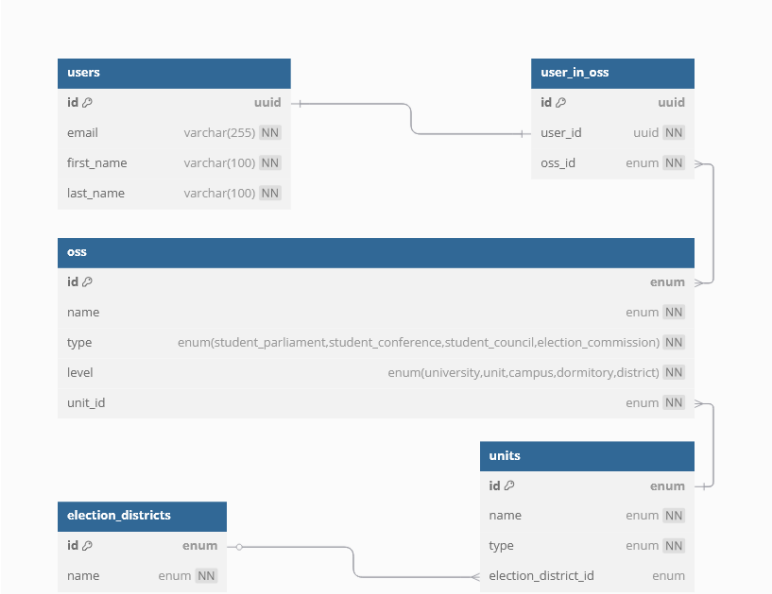

Рис. 2 – Схема бази даних oss-сервісу

Примітка. Певні поля були опущені для кращого розуміння.

Однією з найважливіших частин розробки була **серверна логіка**. Ми створили базовий REST API, який обробляє запити від клієнтів, керує структурою бази даних та надає інформацію, яку вона містить.

Наведемо приклади деяких ендпоінтів oss-сервісу:

Базові CRUD операції:

- **GET /oss** - отримати всі OSS організації з фільтрацією;
- **GET /oss/:id** - отримати OSS за ID;
- **POST /oss** - створити нову OSS організацію;
- **PUT /oss/:id** - оновити OSS організацію;
- **DELETE /oss/:id** - видалити OSS організацію.

Спеціалізовані ендпоінти:

- **GET /oss/by-type/:type** - отримати OSS за типом. Підтримувані типи: student_parliament, student_conference, student_council, election_commission;
- **GET /oss/by-level/:level** - отримати OSS за рівнем. Підтримувані рівні: university, unit, campus, dormitory, district;
- **GET /oss/unit/:unitId/oss** - отримати OSS для конкретного підрозділу.

Важливим аспектом стало впровадження **автентифікації та авторизації**, оскільки безпека даних – ключова вимога нашої системи. Крім того, для підвищення стабільності система розгортається в **Docker-контейнерах**, що дозволяє автоматизувати оновлення та розгортання сервісів.

Висновки

У результаті виробничої практики було почато роботу над створенням масштабованої та функціональної платформи, яка вирішує основні проблеми студентського самоврядування, автоматизує документообіг, голосування, комунікацію та інші процеси.

Завдання на практику дозволили мені поглибити власні знання у галузі програмної інженерії та Web-технологій, справитися з викликами роботи у команді, застосувати власні вміння у створенні надійної архітектури проєкту.

Завдання є близькими до тих, що зустрічаються на підприємствах. Нові уміння надають мені можливість впевнено почати роботу над реальними проєктами; проходити технічні співбесіди, де потрібно виконувати тестові завдання; більш глибоко розвиватися у напрямі розробки Web-застосунків.

Я ретельно опрацював матеріали та опанував нові технології, покращив свої навички у проєктуванні, написанні, тестуванні, деплойменті програмного забезпечення.

Перелік джерел посилання

1. Node.js [Електронний ресурс] // Офіційний веб-сайт. – Режим доступу: https://nodejs.org/
2. TypeScript [Електронний ресурс] // Офіційна документація Microsoft. – Режим доступу: https://www.typescriptlang.org/
3. NestJS Framework [Електронний ресурс] // Офіційна документація. – Режим доступу: https://nestjs.com/
4. NestJS Documentation [Електронний ресурс] // Повна документація фреймворку. – Режим доступу: https://docs.nestjs.com/
5. NestJS Controllers [Електронний ресурс] // Документація по контролерах. – Режим доступу: https://docs.nestjs.com/controllers
6. NestJS Providers and Services [Електронний ресурс] // Документація по провайдерах. – Режим доступу: https://docs.nestjs.com/providers
7. Sequelize ORM [Електронний ресурс] // Офіційна документація. – Режим доступу: https://sequelize.org/
8. NestJS Sequelize Integration [Електронний ресурс] // Документація інтеграції з базами даних. – Режим доступу: https://docs.nestjs.com/techniques/database#sequelize-integration
9. NestJS Microservices [Електронний ресурс] // Документація мікросервісів. – Режим доступу: https://docs.nestjs.com/microservices/basics
10. JetBrains WebStorm [Електронний ресурс] // Офіційний сайт IDE. – Режим доступу: https://www.jetbrains.com/webstorm/
11. WebStorm Documentation [Електронний ресурс] // Документація по WebStorm. – Режим доступу: https://www.jetbrains.com/help/webstorm/
12. Prettier [Електронний ресурс] // Форматувач коду. – Режим доступу: https://prettier.io/
13. Docker [Електронний ресурс] // Платформа контейнеризації. – Режим доступу: https://www.docker.com/
14. Docker Compose [Електронний ресурс] // Документація Docker Compose. – Режим доступу: https://docs.docker.com/compose/
15. npm Registry [Електронний ресурс] // Реєстр пакетів Node.js. – Режим доступу: https://www.npmjs.com/