МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

ПРОЕКТУВАННЯ ТА РОЗРОБКА МОДУЛЯ НАУКОВОЇ ЗВІТНОСТІ КАФЕДРИ НА ПРИКЛАДІ СЕРВІСУ "ЕНАУКОВАДЕКЛАРАЦІЯ"

Кваліфікаційна робота
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу 441 групи
Спеціальності: 121 Інженерія програмного забезпечення
Освітньо-професійної програми:
Інженерія програмного забезпечення
Новіков М.М.
Керівник: кандидат пед. н., доцент Вінник М.О.
Рецензент: Єрмакова-Черченко Н.О., доцент кафедри фізики, ХДУ

Херсон – 2023

ЗМІСТ

ВСТУП

Кожен університет, окрім освітніх послуг, надає можливість студентам та викладачам займатися науковими дослідженнями. Під час цього науковець видає публікації (статті, монографії, підручники, тези лекцій), бере участь у конференціях, а також у проєктах, також отримує патенти на винаходи або авторські права на них. Викладачі ще часто залучають студентів своїх факультетів до досліджень, і ті теж публікують власні наукові праці з посиланням на роботи керівника. До наукової діяльності університету також залучають іноземних спеціалістів, зазвичай у рамках міжнародного співробітництва навчальних закладів.

У кінці кожного року кафедри публікують звіти зі своєї наукової діяльності та викладають на сайті університету. Для цього в кожної кафедри є шаблон, який заповнюють її працівники. Це досить довгий процес, що потребує уваги та часу. Найкращим виходом із ситуації на сьогодні є використання спеціалізованих систем наукового звітування.

У цій роботі буде розглянуто веб сервіс Херсонського державного університету під назвою «еНауковаДекларація», призначений для створення звітів з наукової діяльності кафедри, а також для відображення даних про науковців вишу, а на його основі буде розроблено повноцінний кросплатформовий додаток для наукового звітування.

Актуальність роботи полягає в тому, що сервіс «еНауковаДекларація» буде дуже корисним для співробітників ХДУ, оскільки дозволить їм робити наукові звіти, не користуючись сторонніми сервісами. Сайт розташований у середовищі університету, а отже, матиме значно нижче навантаження, крім того, не міститиме надлишкової інформації з інших університетів.

Об’єктом дослідження є розробка веб сервісу «еНауковаДекларація» та дослідження можливостей його майбутнього покращення.

Предметом дослідження є модуль наукової звітності «еНауковаДекларація».

Метою дослідження є розробити модуль наукової звітності «еНауковаДекларація», визначити об’єм змін, що потрібно внести, описати шляхи їх внесення та продумати механізми розміщення сервісу на серверах.

Зазначену вище мету дослідження буде досягнуто виконанням наступних **завдань:**

- дослідити аналогічні сервіси наукового звітування, що вже існують;
- визначити основні переваги й недоліки таких систем;
- описати специфікацію додатку за допомогою UML-діаграм;
- розробити власне вихідний код сервісу, його бази даних, бекенд та фронтенд частини;
- продумати конфігурацію додатку для розміщення на сервері, використовуючи засоби контейнеризації Docker.

Робота складається зі вступу, основної частини, що поділена на три розділи, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1
ОГЛЯД АНАЛОГІВ СИСТЕМ НАУКОВОЇ ЗВІТНОСТІ ДЛЯ УНІВЕРСИТЕТІВ ТА АНАЛІЗ ЇХНІХ ОСОБЛИВОСТЕЙ

1.1. Основні можливості систем наукового звітування для університетів та найпоширеніші приклади

Існує багато різних систем наукової звітності для університетів, які забезпечують збір та аналіз даних про наукову діяльність вузу та його науковців. Деякі з них можуть бути спеціалізовані для певних дисциплін або факультетів, тоді як інші можуть бути загальними для всього університету. Вони надають можливість ефективніше планувати та управляти дослідницькою роботою вишу. Серед найпопулярніших рішень можна виділити наступні сайти:

1. Pure (Elsevier)
2. Symplectic
3. Converis
4. OpenScholar

Це, звісно, далеко не повний список таких систем, проте ці є найвідомішими та використовуються університетами США, Європи і т.д. Розглянемо ближче кожну з них, їхні можливості та спектр використання.

Розпочнімо з Pure. Над цією системою працює компанія Elsevier, головний офіс якої знаходиться в Амстердамі, Нідерланди. Як указано на головному сайті, вони є лідерами з аналітики та інформування у сфері досліджень та охорони здоров’я, таким чином допомагаючи клієнтам у прийнятті критичних рішень. [1] Ця система використовується такими

відомими у світі вишами, як Гонконгський політехнічний університет, Прінстонський університет, Міннесотський університет тощо. Загальна кількість таких упроваджених систем перевищує 300 на території 47 країн світу, до того ж, за статистикою, 90% клієнтів задоволені її використанням. [2] Pure працює з перевіреною моделлю взаємопов'язаних даних, що включає автоматичне джерело даних, розширені параметри обміну даними та складний детальний доступ, робочий процес і контроль якості даних. Це означає об'єднання дослідницької інформації з доступних джерел даних, що забезпечує швидкий доступ до неї. Схема взаємодії даних зображена на рис. 1.1.

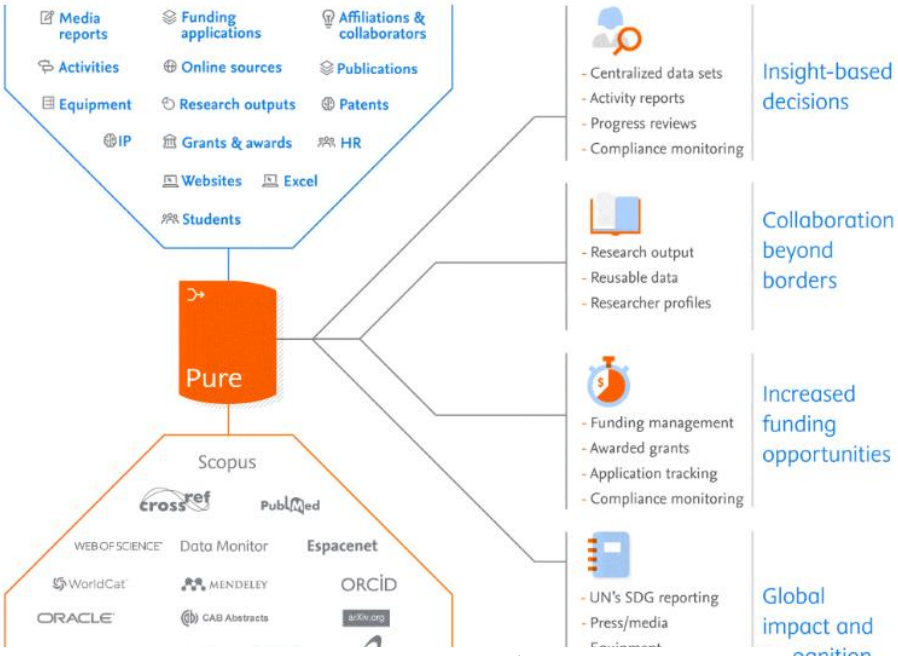


Рисунок 1.1. Схема-принцип роботи системи Pure

На схемі відображено, що Pure взаємодіє з популярними наукометричними базами даних, такими як Orcid, Scopus, Web of Science та інші. Окрім цього, вона індексує й інші онлайн ресурси на предмет наукових робіт та результатів наукової діяльності. Усі дані компонуються в зручному форматі та можуть бути використані повторно для звітування, для чого в системі передбачений окремий модуль – Reporting Module. [3]

Наступною подивимося на Symplectic. Це система виробництва компанії Digital Science, головний офіс якої знаходиться в Лондоні, Великобританія, і ще 2 офіси в Кембрідж (Массачусетс, США) та Ясси (Румунія). Цією системою користуються такі провідні навчальні заклади світу, як Оксфордський університет, Імперський коледж Лондона, Мельбурнський університет, університет імені Карнегі Меллона, у загальній кількості – більше 115 закладів по всьому світу. До того ж, на головному сайті згадується, що система оцінена як одна з найкращих систем управління грантами. Symplectic працює над розвитком знань, надаючи широкі можливості для менеджменту знаннями, які допомагають університетамта іншим установам досягати своїх дослідницьких цілей. Головні продукти компанії Digital Science – це Symplectic Elements, система, що фіксує, досліджує та відображає наукову діяльність, і Symplectic Grant Tracker – додаток, що надає спонсорам і грантовим організаціям засоби оптимізації життєвого циклу управління грантами. Elements — це система керування науковими дослідженнями з широким спектром налаштувань, що отримує дані з багатьох джерел для утворення організаційних даних. Навчальні заклади можуть об’єднувати наукову інформацію та систематизувати її протягом усього дослідження, зменшуючи адміністративне навантаження, отримуючи нові цікаві ідеї та демонструючи вплив досліджень у реальному світі. [4] Як і Pure, Symplectic бере дані про результати наукових досліджень із популярних наукометричних баз даних: Altmetric, ARXIV, CINII, Crossref, Dimensions (база даних від розробника Digital Science), DBLP, Europe-PMC, Figshare, Google Books, MLA, ORCID, PubMed, REPEC, SSRN, Scopus та Web Of Science. [5] Таким чином, система дозволяє викладачам зберігати власні наукові роботи в університетських онлайн сховищах, а також надає засоби для моніторингу та забезпечення відкритого доступу до будь-яких даних.

Перейдемо до розгляду Converis. Це система, розроблена компанією Clarivate, що також є власником відомої наукометричної бази даних Web of Science. Converis забезпечує інтеграцію управління інформацією про дослідження та робочими процесами зсередини, поєднуючи внутрішні та зовнішні дані в єдину платформу. Кожен суб’єкт має власні потреби, тому Converis пропонує цілісний підхід й охоплює весь процес досліджень. Converis — це інтерфейс, який пов'язує різні системи навчальних закладів, що здійснюють збір інформації щодо досліджень, забезпечує стандартизацію для формування профілів та дозволяє зменшити кількіть інформації, що треба вводити повторно. Converis є вузлом для об'єднання репозиторіїв, особистих бібліотек, баз даних вишів та систем їх організації. Також система дозволяє автоматично виймати нові дані з наукометричної бази даних Web of Science та співставляти з науковцями. Основні задачі, що допомагає виконати система:

- Направляти та керувати робочими процесами та твердженнями в дослідженнях
- Отримувати консультування з питань бюджету, видатків та прогнозування
- Призначати та контролювати завдання та строки
- Збирати інформацію щодо проекту, яка пов'язує результати з фінансуванням
- Перевіряти вільний доступ до репозиторію та бібліометрії
- Об'єднувати дані про найважливіші дослідницькі процеси та діяльність для внутрішньої та зовнішньої звітності
- Проводити ліцензування інтелектуальної власності для патентів, експертиз та продуктів з управління робочими процесами
- Просувати та рекламувати дослідження та діяльність вашої установи через публічний та доступний для пошуку веб-портал. [6]

І, нарешті, перейдемо до OpenScholar. Система вироблена в Бостоні (Массачусетс, США) і використовується переважно університетами США, наприклад, Каліфорнійський, Гарвардський, Віргінський. Вона надає доступ до так званої мережі LYNX – це мережа приблизно з 40 000 вчених, дослідників і науковців, які об'єднуються, співпрацюють і діляться своїми поточними дослідженнями. Дослідники з веб-сайтами OpenScholar автоматично отримують доступ до LYNX, щоб шукати та бути знайденими. [7, 8] Ця система дозволяє вести каталог і профілі дослідників, вчених і викладачів, автоматично форматувати академічні публікації та публікувати їх у сховищах, публікувати та керувати всіма типами презентацій, налаштувати теги та відображення колекцій посилань, публікувати та керувати подіями, включаючи реєстрації, відстежувати популярні сторінки, області досліджень і демографічні показники відвідувачів через Google Analytics. Сервіс також дозволяє швидко публікувати свої роботи, обравши бібліографічний формат, тип публікації, а також автоматично поширити їх через Google Scholar, Orcid, PubMed & RePEc. [8]

З наведених вище прикладів можна підсумувати, що у світі існує багато систем наукового звітування, і найчастіше вони орієнтовані не лише на університети, а й на інші установи, наприклад, медичні заклади. До того ж, ці сервіси розрізняються за функціоналом та територіями поширення, хоча й помітно, що більшість із них спрямована на використання у всьому світі.

Але існують і більш локальні рішення, що створюються на базі конкретного навчального закладу. Вони містять дедалі менше даних про науковців, проте зорієнтовані саме на працівників вишу, а тому швидше обробляють дані й за необхідності формують звіти не тільки по всьому університету, а й по конкретній кафедрі. До подібних рішень належать, наприклад, система «Інтелект» (НУ «КПІ ім. І. Сікорського»)[9], а також

Е-Портфоліо (Київський університет ім. Б. Грінченка). [10] Обидва сайти мають приблизно однакову мету: збір та зберігання даних про науковців, їхню діяльність та оформлення звітів за певний проміжок часу.

1.2. SWOT-аналіз систем наукового звітування

Отже, коли знайдені аналогічні рішення, що вже існують, та визначені їхні основні можливості, потрібно провести аналіз їхніх переваг і недоліків. Розглядатимемо в тому самому порядку, як і в розділі 1.1.

Система звітування Pure має наступні переваги:

- Багато можливостей інтеграції: Pure інтегрується з багатьма системами, а саме Web of Science, Scopus, ORCID та інші. Це дозволяє автоматизувати процес збору та аналізу даних, а також ділитися даними з іншими системами.
- Просте налаштування: Pure має широкі можливості налаштування, що дозволяє відповідати потребам користувачів та організацій.
- Легкість використання: інтерфейс Pure прозорий та зрозумілий для користувачів, що дозволяє оперативно вивчити систему та використовувати її майже одразу.
- Підтримка: Pure надає значну підтримку для користувачів, а саме навчальні матеріали, онлайн-підтримку та інше.
- Аналітика: Pure надає можливості для аналізу даних та звітності, що дозволяє організаціям отримувати інформацію про свою наукову діяльність та її результати.

Недоліки системи:

- Вартість: Pure є досить дорогим рішенням, особливо для малого бізнесу з обмеженим бюджетом.

- Системні вимоги: Pure вимагає певного рівня технічної інфраструктури, що може бути недоступним для деяких організацій.
- Імпорт даних: Pure обмежено імпортує дані з інших систем, що може становити проблему для організацій, які використовують багато систем.
- Локалізація: Pure має обмежені можливості локалізації для користувачів з різних країн, що може бути неприємним для організацій, які мають міжнародний характер.
- Необхідність навчання: хоча інтерфейс Pure є легким для використання, все ж може знадобитися додаткове навчання для повного використання системи та отримання максимальної користі з її функцій.

Перейдемо до переваг Symplectic:

- Широкі можливості інтеграції: Symplectic може бути інтегрований з різними системами, такими як системи керування дослідженнями, електронні бібліотеки, системи керування науково-дослідними проектами тощо. Це дозволяє забезпечити єдиний підхід до зберігання та використання наукової інформації в організації.
- Легкість використання: інтерфейс Symplectic є дуже простим та легким у використанні, що дозволяє користувачам швидко зрозуміти, як користуватися системою.
- Аналітика: Symplectic надає можливість отримати детальну статистику про наукові досягнення організації, що дозволяє зробити аналіз та визначити деякі тенденції у розвитку наукової діяльності організації.
- Підтримка користувачів: Symplectic надає високу якість підтримки користувачів, що дозволяє швидко вирішувати будь-які питання та проблеми, які можуть виникнути при використанні системи.

Недоліки Symplectic:

- Висока вартість: Symplectic є досить дорогою системою, що може бути не підходять для деяких організацій.
- Обмеження в імпорті даних: Symplectic може мати обмеження в імпорті даних з інших джерел, що може бути проблемою для деяких організацій, які мають великий обсяг даних.
- Необхідність постійного оновлення: як і будь-яка інша система, Symplectic потребує постійного оновлення та підтримки, що може бути витратним та часомірним процесом.
- Можливість виникнення помилок: Symplectic може мати деякі помилки та бізнес-процеси, які не завжди підходять для всіх організацій.
- Складність інтеграції: Symplectic може бути складним для інтеграції з деякими сторонніми системами, що може створити проблеми для деяких організацій.

Тепер розглянемо переваги системи Converis:

- Широкі можливості інтеграції: Converis може бути легко інтегрована з іншими системами, такими як Pure та Symplectic, що забезпечує більш повне та точне зібрання наукової інформації.
- Розширені аналітичні можливості: Converis забезпечує широкий спектр аналітичних можливостей для аналізу наукової продуктивності та впливу наукових досліджень.
- Підтримка глобальних стандартів: Converis підтримує глобальні стандарти, такі як CERIF (Common European Research Information Format), що забезпечує сумісність з іншими системами зберігання та обробки наукової інформації.
- Легка інтеграція з Web of Science: Converis може бути легко інтегрована з Web of Science, що дозволяє більш повно та точно зібрати наукову інформацію та підвищити видимість наукових досліджень.

Одним з недоліків системи Converis може бути складність налаштування та використання, особливо для користувачів з обмеженим досвідом роботи зі схожими системами. Також, вартість розгортання та підтримки Converis може бути значною для менших університетів або наукових установ. Крім того, інтерфейс користувача Converis не завжди є інтуїтивно зрозумілим та зручним для використання.

Інший недолік системи Converis полягає у її обмеженості щодо функціоналу порівняно з іншими системами наукового звітування, наприклад, Symplectic та Pure. Наприклад, у Converis може бути менше можливостей для налаштування профілю дослідника та імпорту даних з інших джерел порівняно з іншими системами.

Також слід зазначити, що Converis, як і інші системи наукового звітування, має деякі обмеження у збереженні та обробці даних. Наприклад, система може не підтримувати деякі типи даних або мати обмеження у кількості записів, що можна додати до профілю дослідника.

Далі – про систему OpenScholar. Вона є безкоштовною та відкритою, що дозволяє користувачам з легкістю розгорнути та налаштувати свій веб-сайт. Завдяки цьому, вона може бути особливо привабливою для наукових установ з обмеженим бюджетом, а також для дослідників, які шукають можливості публікації своїх досліджень та створення персонального веб-сайту.

Окрім того, OpenScholar має простий та зрозумілий інтерфейс користувача, який дозволяє швидко та легко створювати та публікувати нові матеріали, такі як статті, презентації та інші ресурси. Вона також має вбудований функціонал для зберігання та відображення бібліографічних записів, що дозволяє дослідникам легко відстежувати свої публікації.

Недоліки OpenScholar включають обмежені можливості налаштування та інтеграції з іншими системами. Наприклад, вона може

не мати можливостей для автоматичного імпорту даних з інших джерел або може бути обмежена у підтримці деяких форматів даних. Також, вона не має такого ж широкого спектру функцій, які доступні у деяких інших системах наукового звітування, таких як Symplectic та Pure.

Крім того, вона не має тих же розширених можливостей для зберігання та обробки даних, які доступні у деяких інших системах, і може бути менш ефективною для великих та складних проектів досліджень.

Тепер розглянемо систему «Інтелект» Київського політехнічного інституту.

Переваги:

- Автоматизована підготовка звітів, зменшення часу, необхідного для їх заповнення;
- Можливість зручного ведення наукової діяльності із підтримкою оптимальних процесів та високого рівня автоматизації;
- Полегшення процесу збирання та обробки даних наукової діяльності, а також їх подальший аналіз;
- Висока ступінь безпеки даних та забезпечення конфіденційності;
- Зручне та інтуїтивно зрозуміле користування системою.

Недоліки:

- Система розроблена тільки для потреб КПІ ім. Ігоря Сікорського, тому вона не є універсальною для інших університетів або наукових установ;
- Можливість виникнення труднощів під час реалізації та підтримки системи;
- Відсутність розширених можливостей звітування порівняно з іншими системами наукового звітування.

І, нарешті, система «Е-Портфоліо» Київського університету ім. Грінченка.

Переваги:

- Підвищення ефективності наукової діяльності: Система дозволяє створювати та оновлювати індивідуальні портфоліо, що дозволяє науковцям зосередитися на своїй діяльності, не витрачаючи багато часу на підготовку документів для звітності.
- Зручне зберігання та пошук інформації: Портфоліо зберігає всі дані про наукову діяльність в одному місці, що дозволяє легко знайти потрібну інформацію та ділитися нею з колегами.
- Підвищення рівня професіоналізму: Використання цієї системи забезпечує створення індивідуальної кар'єри науковця, дозволяючи презентувати свої досягнення та розвивати свої навички наукової роботи.
- Відкритість та доступність: Система є відкритою та безкоштовною для користувачів, що дозволяє будь-якому науковцю використовувати її для своїх потреб.

Недоліки:

- Складність використання: деякі користувачі можуть знайти інтерфейс складним у використанні, особливо якщо вони не мають досвіду використання подібних систем.
- Обмежені можливості настройки: система може бути обмеженою у налаштуванні користувачами, що обмежує їх можливості створення унікального дизайну та інших настроюваних параметрів.
- Залежність від доступу до Інтернету: щоб використовувати систему ePortfolio, користувач повинен мати доступ до Інтернету, що може бути проблемою для деяких користувачів.
- Відсутність інтеграції з іншими системами: система може не бути повністю інтегрованою з іншими системами, такими як бібліографічні менеджери або системи звітності, що може вимагати додаткових зусиль користувача для виконання повних завдань.

- Потреба у підтримці: так як це локальна система, потрібна підтримка для її функціонування, оновлення та забезпечення безпеки.

Якщо підсумувати наведені вище факти, то отримаємо наступне: на жаль, не існує ідеальних систем наукової звітності, яка підходила б усім, тому деякі університети або використовують декілька таких сервісів одразу, або залучають спеціалістів та розробляють власні лише для своїх потреб, а іноді ще комбінують використання окремих існуючих систем та внутрішніх, розроблених виключно для певної інституції. Оскільки багато з таких рішень є платними, а інші не підходять через обмеженість їх використання лише співробітниками окремого закладу, то найкращим рішенням є розроблення власної системи наукової звітності. У Херсонському державному університеті це сервіс «еНаковаДекларація».

РОЗДІЛ 2
ОПИС СПЕЦИФІКАЦІЇ ДОДАТКУ «ЕНАУКОВАДЕКЛАРАЦІЯ»

2.1. Технології

Головна задача при розробці сучасного додатку – це адаптивність та кросплатформовість, тому важливо правильно обрати потрібні технології. Для створення додатку «еНауковаДекларація» необхідні наступні:

- для API – ASP.NET Core 6 (мова програмування C# 10), фреймворк для роботи з базою даних – Entity Framework Core 6, провайдер бази даних – Microsoft SQL Server останньої версії. За роботу додатку відповідатиме .NET runtime версії 6.
- для фронтенд додатку потрібні популярний веб фреймворк Angular версії 15, для написання коду використовує мову програмування TypeScript (для логіки додатку), а для розмітки веб сторінок – мови розмітки HTML та CSS. Сам додаток запускатиметься за допомогою NPM версії 9.2.0 та Node.js версії 18.12.1.
- сервіс також буде адаптований для запуску з-під Docker, тому на комп'ютері має бути встановлений Docker Desktop.

Вибір технологій та архітектури сервісу обумовлений їхніми можливостями та перевагами. .NET 6 – це кросплатформове середовище, може бути запущене з-під Windows або Linux, просте у вивченні та розробці. Версія 6 є довгопідтримуваною (LTS), що дозволить не оновлювати версію кожного року, коли виходить нова, відповідно, це відбуватиметься набагато рідше, до того ж, на сьогодні вона поступається в продуктивності та швидкості роботи лише версії 7. Екосистема .NET 6 пропонує: спрощену розробку – нові функції мови в C# 10 зменшують обсяг коду, який потрібно написати. А інвестиції у веб-стек і мінімальну

кількість API дозволяють легко швидко писати менші та швидші мікросервіси; кращу продуктивність – .NET 6 — це найшвидша веб-платформа з повним стеком, яка знижує витрати на обчислення, якщо ви працюєте в хмарі; висока продуктивність – .NET 6 і Visual Studio 2022 забезпечують гаряче перезавантаження, нові інструменти Git, інтелектуальне редагування коду, надійні інструменти діагностики та тестування, а також кращу командну співпрацю. [11]

Для фронтенду обрано Angular через його інтуїтивну зрозумілість, легкість у вивченні та майбутній підтримці, а також хорошу структурованість коду, до того ж, він також є кросплатформовим. Його середовище запуску Node.js також матиме LTS-версію, що дозволить довше працювати на ній без оновлення. Мова TypeScript є наступником мови JavaScript, що спрощує розробку й читабельність коду. Також до переваг Angular можна віднести детальну документацію, зручний інтерфейс командного рядка, модульність, підтримка сервісів та використання залежностей (dependency injection). [12]

Система керування базами даних SQL Server обрана через свою популярність та найкращу адаптованість до .NET додатків. Серед переваг SQL Server можна виділити наступні: проста установка, високий рівень безпеки (шифрування даних, закритість платформи), різноманіття версій, можливість відновлення втрачених чи пошкоджених даних. [13]

Система контейнеризації Docker обрана через її прозорість, доступність та адаптивність до будь-якого середовища, єдиною умовою для нього є наявність достатнього об’єму оперативної пам’яті на сервері (рекомендується не менше ніж 16 ГБ). Переваги Docker у створенні та розгортанні програм:

- Кешування кластера контейнерів
- Гнучкий обмін ресурсами

- Масштабованість - на одному хості можна розмістити багато контейнерів
- Запуск служби на обладнанні, яке набагато дешевше, ніж стандартні сервери
- Швидке розгортання, легкість створення нових примірників і швидші міграції.
- Легкість переміщення та обслуговування ваших програм
- Краща безпека, менший доступ, необхідний для роботи з кодом, що працює всередині контейнерів, і менше програмних залежностей. [14]

Отже, таким є остаточний перелік технологій, потрібних для розробки сервісу.

2.2. Вимоги до проєкту та UML діаграми

Для того, щоб розробляти додаток, потрібно визначити вимоги до проєкту та зобразити основні процеси на UML діаграмах. Розпочнемо з вимог.

Цілі й задачі:

- Відображення й збереження інформації про всі публікації викладачів
- Відображення й збереження записів про конференції, в яких брали участь викладачі
- Збереження записів про підвищення кваліфікації викладачів
- Розміщення посилань на сторінки викладачів у наукометричних базах даних (наприклад, Scopus)
- Генерація наукових звітів на основі наявної інформації про викладача

Мова додатку:

- Українська

Цільова аудиторія:

- Викладачі ХДУ
- Працівники кафедр факультетів ХДУ, що їх реєструють

Платформа додатку:

- Web (ASP.NET Core + Angular)

Авторизація: обов'язкова для розміщення інформації

Поля авторизації:

- Повне ім'я викладача (обов’ язкове)
- Працююча електронна адреса (обов'язкове)
- Кафедра (обов’язкове, список кафедр оновлюваний)
- Стать
- Місце проживання
- Дата народження
- Місце праці
- Посада
- Науковий ступінь
- Учене звання
- Профілі в наукометричних базах (Google Scholar, Scopus, …) (обов’язкове)

Вимоги:

- Авторизація
- Реєстрація (адміністратори, модератори - створювачі профілів)
- Ролі (адміністратор, створювач профілів, звичайний користувач)
- Відображення списку всіх викладачів
- Відображення повної інформації про кожного викладача
- Додавання публікацій по одній
- Додавання списку публікацій через CSV-файл

- Додавання записів про підвищення кваліфікації викладача
- Додавання проектів, в яких бере участь викладач
- Додавання охоронних документів
- Додавання записів про участь у конференціях
- Генерація й відображення наукових звітів про діяльність викладача за певний період часу

Таким чином, визначені основні вимоги до сервісу «еНауковаДекларація». Також сформовані та побудовані відповідні UML діаграми прецедентів, активностей та послідовностей (див. Додаток А, Додаток Б, Додаток В).

РОЗДІЛ 3
ПРОГРАМУВАННЯ ДОДАТКУ «ЕНАУКОВАДЕКЛАРАЦІЯ»

3.1. Бекенд частина додатку (API) та база даних

Для розробки бекенд частини потрібно створити проєкт ASP.NET Core Web API під назвою PrepodPortal.WebAPI і викласти його до GitHub репозиторію [15]. Окрім нього, у рішенні створюються ще 3 проєкти: PrepodPortal.Common, що містить загальні класи та перерахування, PrepodPortal.Core, що містить бізнес-логіку, та PrepodPortal.DataAccess, що містить логіку роботи з базою даних. Для розробки потрібні наступні пакети NuGet (перерахуємо лише головні): Microsoft.EntityFrameworkCore; AutoMapper; FluentValidation; Microsoft.AspNetCore.Authentication.JwtBearer; Microsoft.AspNetCore.Identity.EntityFrameworkCore; Serilog; Swashbuckle.AspNetCore. У проєкт уже за замовчуванням інтегровано Swagger, оскільки це дуже спрощує роботу з API. Swagger дозволяє описати структуру ваших API, щоб машини могли їх читати. Здатність API описувати власну структуру є основою успішності модуля Swagger. [16]

Аутентифікація в сервісі виконується за допомогою JSON Web Token (JWT), що генерується на бекенді та зберігається на фронтенді. Робота з даними користувача та загальна система авторизації забезпечується бібліотекою ASP.NET Core Identity, яка має в собі багатий функціонал щодо роботи з обліковими записами, наприклад, шифрування паролів за допомогою алгоритму SHA256, зручні методи зміни або скидання пароля, пошуку користувача за email, іменем користувача або ID (виглядає як унікальний рядок типу Guid).

Модуль реєстрації нових науковців дозволяє додавати нові облікові записи лише з ім’ям, email та покликаннями науковця на наукометричні бази даних (Orcid, Scopus, WoS). Лише коли всі ці дані введені, створюється користувач. У планах також допрацювати систему надсилання email листів після реєстрації та інтегрувати реєстрацію в цьому сервісі з реєстрацією на KSU Publication. [17]

Структура додатку така, що кожен контролер відповідає за свій певний функціонал або сутність, з якою працює, наприклад, UserController містить усе, що стосується роботи з користувачами. Частина запитів, наприклад, на отримання даних про науковців для головної сторінки, доступні без авторизації й вимагають лише певних додаткових параметрів, найчастіше ідентифікаторів потрібної сутності.

База даних у додатку є версіонованою, тобто кожна зміна в сутностях або структурі бази обов’язково фіксується міграцією, тобто кодом, що оновлює базу даних до певного стану. Усі міграції зберігаються в папці Migrations проєкту PrepodPortal.DataAccess.

Валідація з боку сервера виконується в багатьох контролерах за допомогою бібліотеки FluentValidation, оскільки вона більш гнучка, аніж стандартні валідаційні атрибути в .NET. Для моделей, що є параметрами в методах контролерів, створюються класи-валідатори, де за допомогою методів розширення впроваджуються різні перевірки.

Для перетворення сутностей на моделі та data transferring objects (DTO) використано бібліотеку AutoMapper, але в певних місцях, наприклад, у методах Update, її бажано уникати, бо вона перезаписує об’єкт і база даних сприймає його як новий, а не як зміну старого.

3.2. Фронтенд частина додатку

Для розробки фронтенд частини портібно створити Angular проєкт prepod-portal-webapp у відповідному репозиторії GitHub. [18]

Верхній та нижній блоки сторінки виносяться в header та footer відповідно

На головній сторінці можна побачити всі дані про науковця, які він або вона побажали ввести про себе.

На другій вкладці, Наукова діяльність, розміщено дані про публікації науковця та інші результати його наукової діяльності. Публікації, а саме статті, можна додавати по одній або імпортувати через CSV файл. Імпорт CSV для інших видів публікацій, скоріш за все, буде впроваджено в майбутньому.

На третій вкладці, Наукові звіти, розміщено найголовніший функціонал додатку – модуль наукового звітування. Він дозволяє зробити звіт за кафедрою, загальний або персональний. Принцип його роботи досить простий: він збирає дані по всіх науковцях із заданими параметрами (дати, за які потрібен звіт, кафедра), систематизує та відображає на екрані.

На четвертій вкладці, Проєкти, показано дослідницькі проєкти, в яких брав участь науковець.

У верхній частині екрану можна побачити кнопку Список викладачів – це сторінка, що показує повний список науковців, зареєстрованих на сайті. Цей елемент доступний навіть без авторизації, тому можна подивитися короткі відомості про викладача. Особисті сторінки викладачів також доступні для анонімного перегляду. Друга кнопка дозволяє перейти на сторінку реєстрації викладача, вона доступна усім, окрім регулярних користувачів. А пошукове поле дозволяє здійснити пошук за іменем викладача або частиною імені.

Якщо перейти на власну сторінку, то будуть також доступні посилання на редагування особистих даних та налаштування акаунту. Лінку Редагувати можуть бачити також адміністратори, бо мають права змінювати чужі дані. На рис. 3.1 можна побачити зовнішній вигляд сторінки користувача. Посилання Моя сторінка, очікувано, переводить на

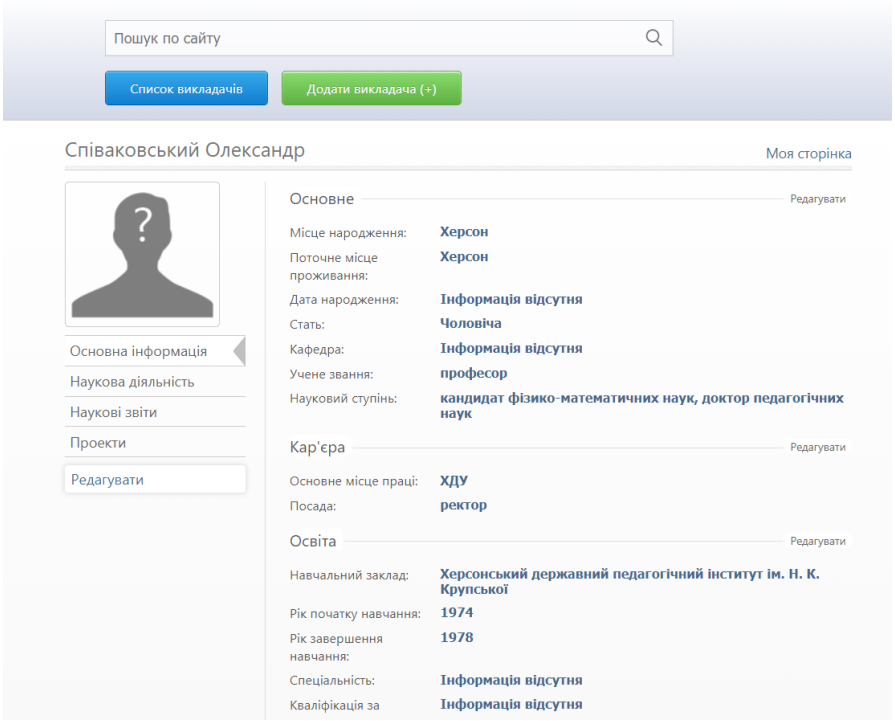


Рисунок 3.1. Сторінка користувача

особисту сторінку користувача.

3.3. Контейнеризація додатку в Docker та Docker-Compose

Для контейнеризації сервісу потрібно внести декілька змін у додатки: по-перше, потрібно додати Dockerfile для бекенд та фронтенд проєктів; по-друге, необхідно створити й налаштувати файл Docker-Compose; по-третє, потрібно додати до обидвох проектів конфігурації для запуску з-під Docker. У цій роботі буде розглянуто запуск апплікацій на операційній системі Windows.

Найголовніша умова – на комп’ютері має бути встановлений Docker Desktop, для інсталювання якого потрібно встановити Windows Subsystem

for Linux, а також будь-який дистрибутив операційної системи Linux, наприклад, Ubuntu.

Після інсталювання потрібних додаткових програмних засобів потрібно відкрити проект PrepodPortal.WebAPI та додати підтримку

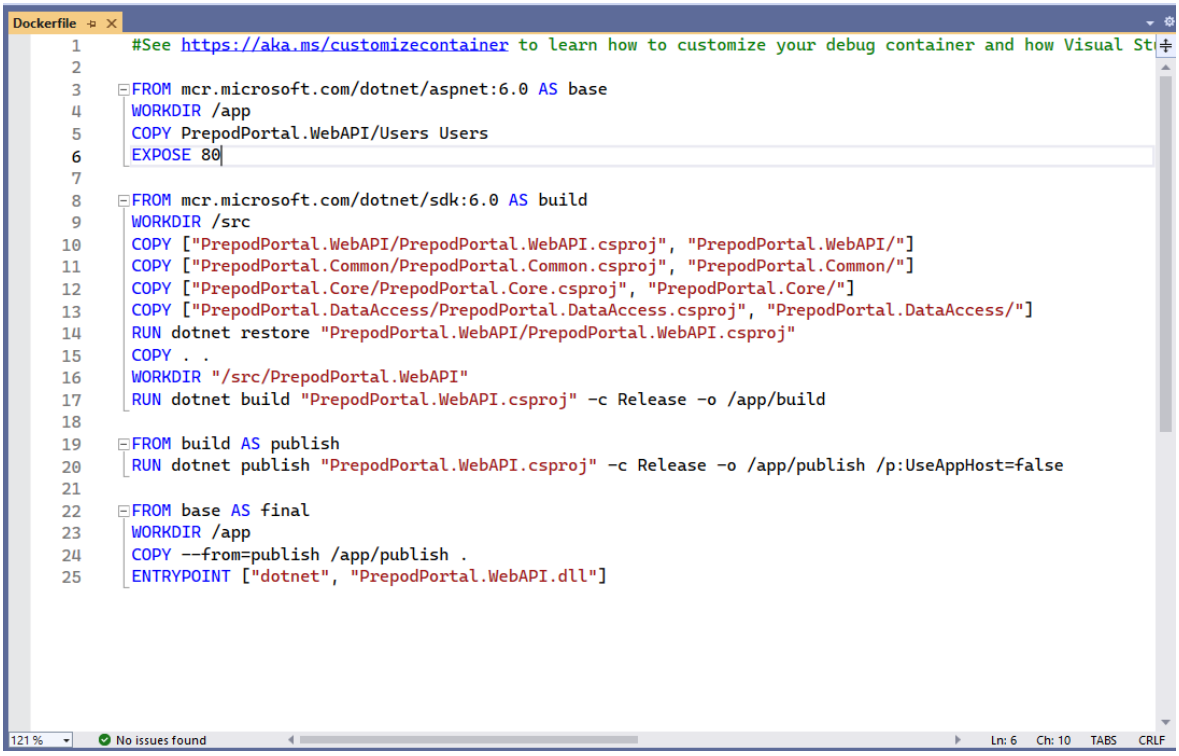


Рисунок 3.2. Конфігурація Dockerfile для бекенд додатку

Docker під Linux-контейнери. Після цього в файлі launchSettings.json з’явиться додаткова конфігурація запуску, а в кореневій папці проєкту створиться Dockerfile (файл не має розширення, при додаванні розширення збирання не працюватиме). Visual Studio власноруч створить потрібну конфігурацію для запуску додатку, проте потрібно уважно перевірити файл на наявність усіх потрібних компонент. Готова конфігурація файлу зображена на рис. 3.2.

Як видно з рядків 3-6, ми використовуємо ASP.NET Core 6 у контейнері, а також копіюємо директорію Users до контейнера для того, щоб там зберігалися аватари користувачів, крім того, задаємо доступ до бекенду через порт 80, тобто адреса буде виглядати як http://localhost:80 . У рядках 8-17 ми копіюємо всі файли проєктів до контейнера, а також

через файл проєкту отримуємо список залежностей, що мають бути встановлені, після чого збираємо проєкт. У рядках 19, 20 ми збираємо все рішення в конфігурації Release. І нарешті, в останніх рядках ми задаємо вхідну точку до нашого додатку.

Рисунок 3.3. Конфігурація Production та рядок з’єднання з базою даних

Тепер задамо конфігурацію для запуску додатку з-під контейнера. Для цього ми маємо додати ще одне середовище – Production. Створимо файл appsettings.Production.json та додамо до нього рядок з’єднання з базою даних (див. рис. 3.3). Окрім цього, у файлі launchSettings.json також

```
"Docker": {
  "commandName": "Docker",
  "launchUrl": "{Scheme}://{ServiceHost}:{ServicePort}/swagger",
  "publishAllPorts": true,
  "useSSL": true,
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Production"
  }
```

потрібно внести зміни для Docker конфігурації. Весь потрібний код уже

Рисунок 3.4. Конфігурація для запуску бекенду під Docker

написаний у файлі, потрібно лише замінити значення ASPNETCORE_ENVIRONMENT (це змінна середовища в .NET) на Production. Кінцевий вигляд цієї частини коду зображено на рис. 3.4.

Наступним кроком є задання налаштувань для фронтенду. По-перше, у проєкті prepod-portal-webapp потрібно до кореневої директорії проєкту додати файл nginx.conf (файл конфігурації Nginx – це вільний для розповсюдження веб та проксі сервер, створений для Windows та Unix [19]), у якому треба розмістити налаштування для запуску Angular серверу (див. Додаток Г). Окрім цього, потрібно створити Dockerfile, але вже налаштований для Angular (див. рис. 3.5). У ньому потрібно вказати

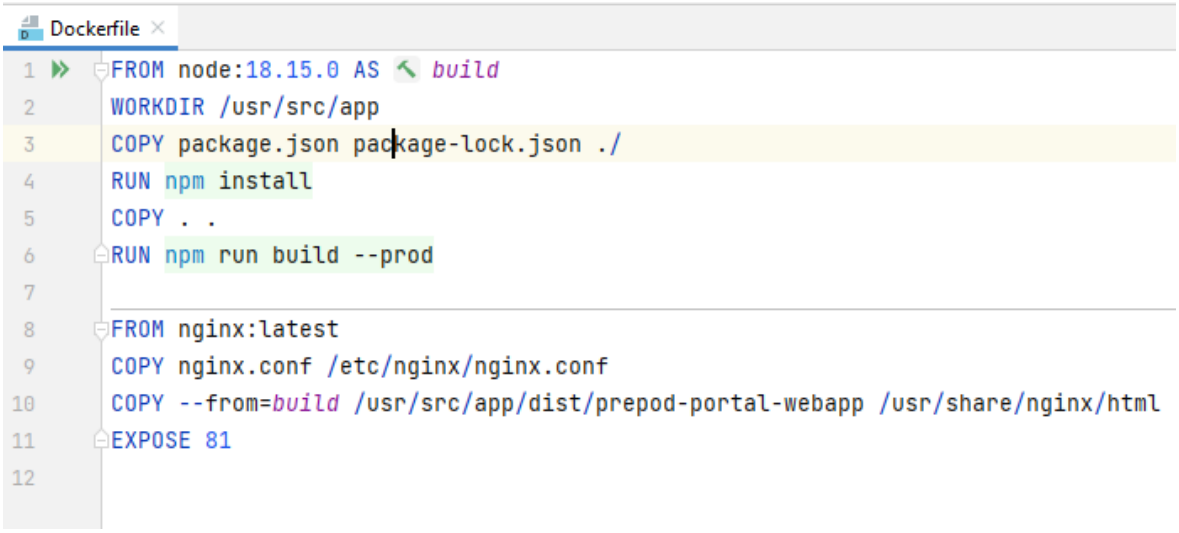


Рисунок 3.5. Dockerfile для Angular додатку

LTS-версію node, як і було вирішено раніше, потім скопіювати файли package.json та package.lock.json, де зберігаються проєктні залежності, після чого задати команду збирання, скопіювати інші файли та зібрати проєкт. Останніми налаштуваннями в файлі будуть nginx та задання його конфігурації з доданого вище файлу nginx.conf, а також прослуховування порту 81 для цього додатку, тобто адреса фронтенд частини буде мати вигляд http://localhost:81. Оптимальність такого рішення полягає в тому, що перед виконанням досить важкої та довгої команди npm install ми спочатку копіюємо залежності, що значно оптимізує набір даних, що скачується при збиранні контейнера.

Після зазначених налаштувань треба додати середовище prod у фронтенд додатку. Для цього треба у папці environments додати файл environment.prod.ts, а в ньому задати об’єкт, де вказати production = true та задати значення шляхів до API як http://localhost:80; крім цього, у файлі angular.json треба додати до об’єкта configurations об’єкт prod, у якому помістити налаштування обмежень по розміру стилів та інших файлів, а також задати заміни файлів environment.ts на environment.prod.ts. Повний варіант цих налаштувань див. Додаток Д.

Фінальним налаштуванням Docker є компонування всіх частин у єдину систему за допомогою Docker Compose. Docker Compose — це інструмент, розроблений для визначення й спільного використання багатоконтейнерних програм. За допомогою Compose можна створити файл YAML для визначення служб і за допомогою однієї команди запустити або зупинити все одразу.

Великою перевагою використання Compose є те, що можливо визначити свій стек додатків у файлі, зберегти його в корені репозиторію проекту (тепер воно контролюється версіями) і легко дозволити комусь іншому зробити свій внесок у проєкт. Комусь потрібно лише клонувати репозиторій та запустити програму створення. [20]

Отже, файл має називатися docker-compose.yml та знаходитися вище рівнем, аніж фронтенд та бекенд додатки. Його повну конфігурацію можна побачити в додатку Е. Якщо коротко зазначити його зміст, то спочатку задається контейнер бази даних із образом SQL Server, потім контейнер додатку ASP.NET Core, що залежить від бази даних, та останнім є контейнер для фронтенд додатку. У двох останніх контейнерів також задано шлях до Dockerfile та початковий робочий каталог.

Таким чином, сконфігурувавши файл, ми переходимо за допомогою терміналу або PowerShell до каталогу, де лежить docker-compose.yml, та запускаємо команду docker-compose up. Це створить контейнери та запустить кожен із них у порядку слідування в файлі. Щоб видалити контейнери та зупинити виконання, потрібно натиснути Ctrl-C (переривання термінальної команди), потім прописати docker-compose down.

ВИСНОВКИ

Розроблений в ході роботи сервіс «еНауковаДекларація» допоможе співробітникам Херсонського державного університету робити звіти з науково-дослідницької роботи кафедри на основі праць викладачів та студентів.

У результаті дослідження були проаналізовані аналоги систем наукового звітування, такі як Pure (Elsevier), Symplectic, Converis, OpenScholar, а також вітчизняні розробки «Інтелект» (НУ «КПІ ім. І. Сікорського») та «Е-Портфоліо» (Київський університет ім. Б. Грінченка), визначені їхні основні можливості, до того ж, проведений SWOT-аналіз показав, що найкращим рішенням для навчальних закладів є створення власної системи наукового звітування для внутрішнього користування.

Також було зібрано реєстр вимог сервісу і побудовано відповідні UML-діаграми:

- діаграма прецедентів
- діаграма активностей
- діаграма послідовностей

Окрім цього, було розроблено бекенд, фронтенд частини та базу даних сервісу, використовуючи досліджені й обрані в ході роботи технології, а саме ASP.NET Core 6, Angular 15, Microsoft SQL Server.

Під час розробки було виявлено певні недоліки, що будуть усунені з часом, а саме:

- неможливість відправки email повідомлень через нову політику Gmail щодо використання third-party applications, потрібно шукати інший

поштовий сервіс, можливо навіть вийде використовувати корпоративну пошту Херсонського державного університету;

- під час додавання нового викладача має бути інтеграція з сервісом Publication [17], де має бути створений акаунт науковця з тими самими даними, що й на «еНауковаДекларація», але поки що функціонал не реалізований з боку Publication, тому це відбудеться пізніше.

Варто також зазначити, що всі компоненти сервісу були належним чином сконфігуровані та налаштовані для розміщення за допомогою засобу контейнеризації Docker, використовуючи засоби Docker Compose для поєднання частин в єдине ціле та швидкого запуску/згортання системи цілком. Це дозволить, по-перше, опублікувати додаток на будь-якому сервері (університетському або хмарному), а по-друге, дозволить швидко імплементувати нові зміни в сервіс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

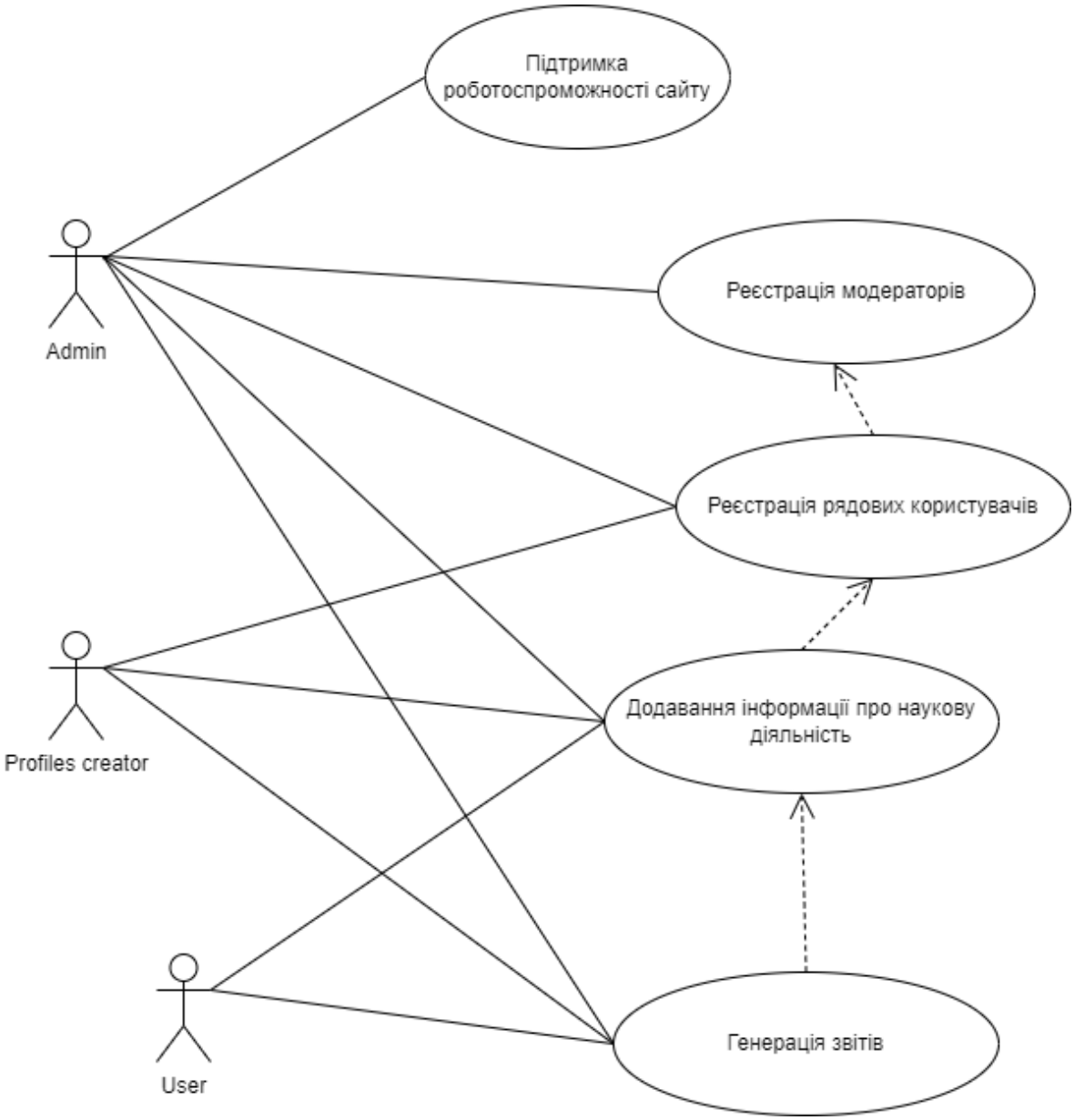
1. About Elsevier. URL: https://www.elsevier.com/about (дата звернення: 07.04.2023)
2. Client portals. URL: https://www.elsevier.com/solutions/pure/pure-in-action (дата звернення: 07.04.2023)
3. How Pure works. URL: https://www.elsevier.com/solutions/pure/how-it-works (дата звернення: 07.04.2023)
4. Symplectic – Digital Science. URL: https://www.digital-science.com/product/symplectic/ (дата звернення: 07.04.2023)
5. The Elements Platform – Symplectic. URL: https://www.symplectic.co.uk/theelementsplatform (дата звернення: 07.04.2023)
6. Converis – Clarivate – CIS. URL: https://clarivate.com/cis/solutions/converis/ (дата звернення: 07.04.2023)
7. Introducing OpenScholar LYNX: Search Research & Find Collaborators. URL: https://lp.theopenscholar.com/lynx (дата звернення: 07.04.2023)
8. Platform | OpenScholar. URL: https://theopenscholar.com/platform (дата звернення: 07.04.2023)
9. Інтеллект | КПІ ім. Ігоря Сікорського. URL: https://intellect.kpi.ua (дата звернення: 07.04.2023)
10. Е-Портфоліо. URL: https://eportfolio.kubg.edu.ua/rating (дата звернення: 07.04.2023)
11. What's new in .NET 6 | Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-6 (дата звернення: 07.04.2023)

12. 8 Benefits of Angular for Your Project | LIGHT-IT. URL: https://light-it.net/blog/8-advantages-of-angular-for-businesses-and-developers/ (дата звернення: 07.04.2023)
13. MS SQL Server History and Advantages – ByteScout. URL: https://bytescout.com/blog/2014/09/ms-sql-server-history-and-advantages.html (дата звернення: 07.04.2023)
14. What Is and What Are the Benefits of Docker Container? URL: https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker-container (дата звернення: 07.04.2023)
15. elkprostoelk/prepod-portal-api: eNaukovaDeklaracia Web API. URL: https://github.com/elkprostoelk/prepod-portal-api (дата звернення: 07.04.2023)
16. What is Swagger. URL: https://swagger.io/docs/specification/2-0/what-is-swagger/ (дата звернення: 07.04.2023)
17. Рейтинг. URL: http://publication.kspu.edu (дата звернення: 07.04.2023)
18. elkprostoelk/prepod-portal-webapp: eScientificDeclaration frontend Angular app. URL: https://github.com/elkprostoelk/prepod-portal-webapp (дата звернення: 07.04.2023)
19. nginx – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Nginx (дата звернення: 07.04.2023)
20. Use Docker Compose. URL: https://docs.docker.com/get-started/08_using_compose/#:~:text=Docker%20Compose%20is%20a%20tool,or%20tear%20it%20all%20down. (дата звернення: 07.04.2023)

ДОДАТКИ

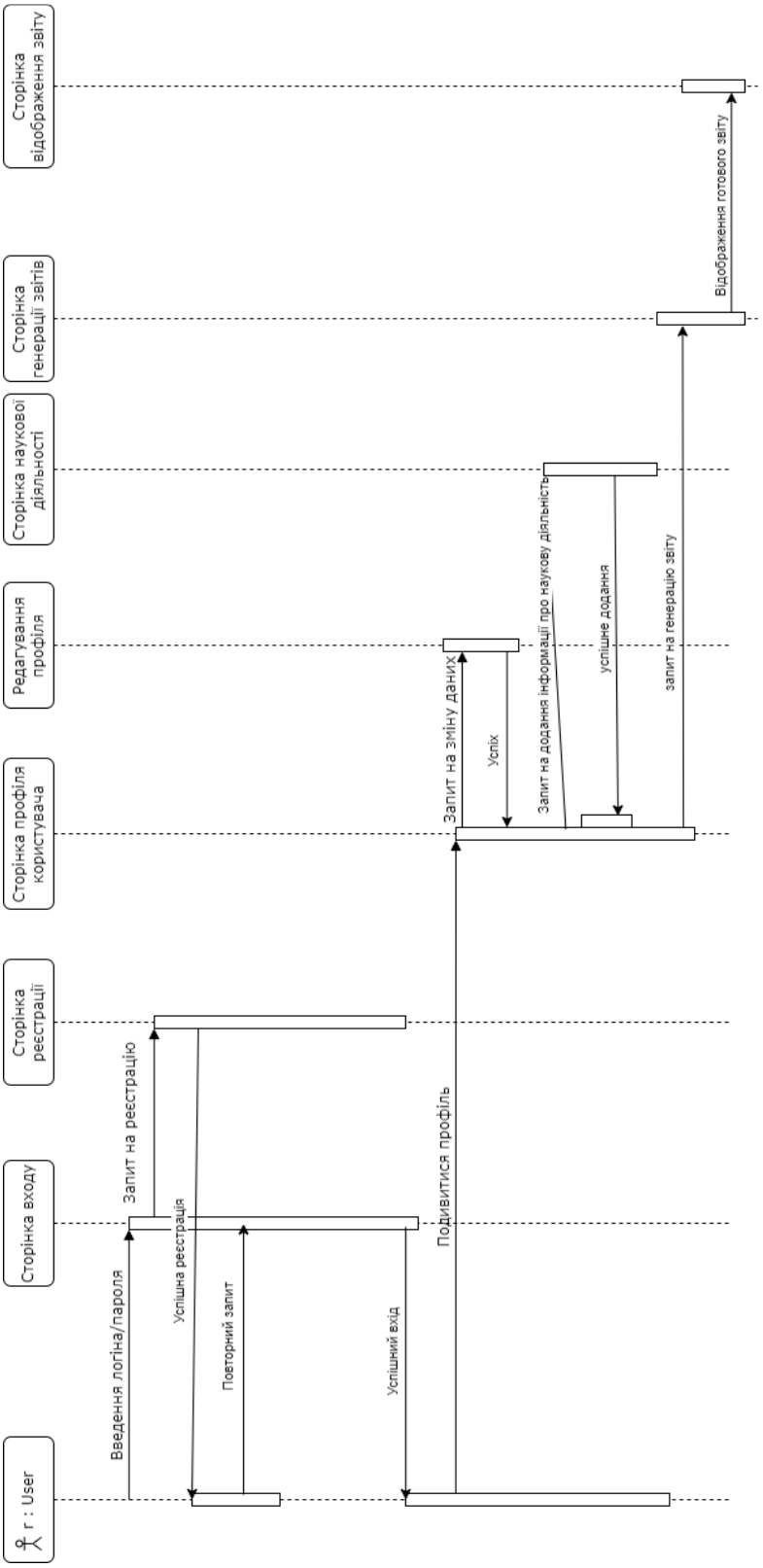
Додаток А

Діаграма прецедентів



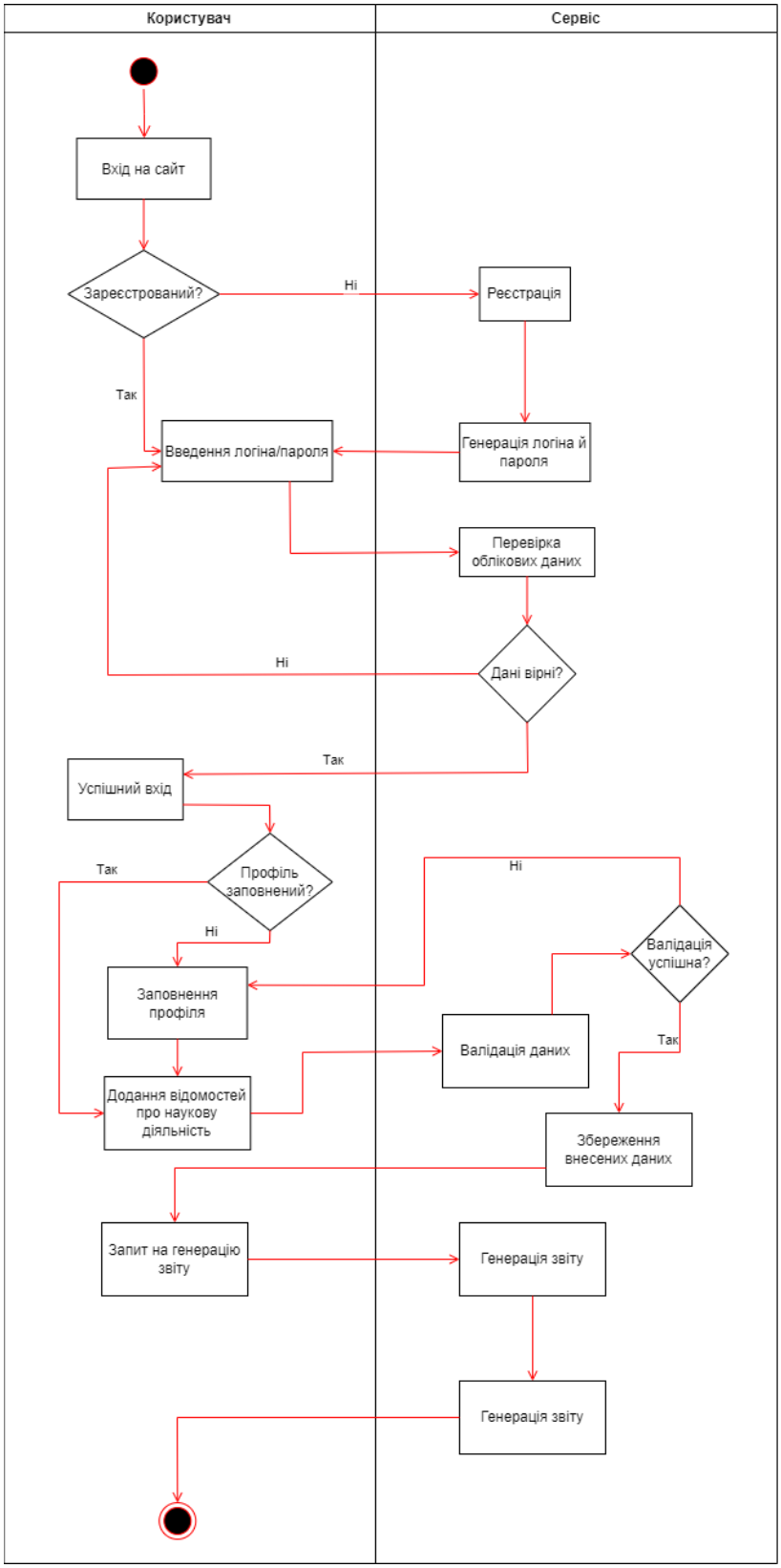
Додаток Б

Діаграма послідовностей



Додаток В

Діаграма активностей



Додаток Г

Конфігурація nginx для фронтенду

nginx.conf

```
worker_processes 1;

events {
  worker_connections 1024;
}

http {
  include mime.types;
  default_type application/octet-stream;
  sendfile on;
  keepalive_timeout 65;

  server {
    listen 81;
    root /usr/share/nginx/html;
    index index.html index.htm;

    location / {
      try_files $uri $uri/ /index.html;
    }

    error_page 404 /404.html;
    error_page 500 502 503 504 /50x.html;

    location = /50x.html {
      root /usr/share/nginx/html;
    }

    location ~* \.(jpg|jpeg|png|gif|css|js|ico)$ {
      expires 365d;
    }
  }
}
```

Додаток Д

Налаштування середовища для фронтенду

angular.json | environment.prod.ts

```
28            "node_modules/bootstrap/dist/css/bootstrap.min.css",
29            "src/styles.css"
30          ],
31          "scripts": []
32        },
33        "configurations": {
34          "prod": {
35            "budgets": [
36              {
37                "type": "initial",
38                "maximumWarning": "500kb",
39                "maximumError": "1mb"
40              },
41              {
42                "type": "anyComponentStyle",
43                "maximumWarning": "10kb",
44                "maximumError": "15kb"
45              }
46            ],
47            "fileReplacements": [
48              {
49                "replace": "src/app/environments/environment.ts",
50                "with": "src/app/environments/environment.prod.ts"
51              }
52            ],
53            "outputHashing": "all"
54          },
```

angular.json | environment.prod.ts

no usages elkprostoelk

```
1  export const environment = {
2    production: true,
3    apiPath: 'http://localhost:80/api/',
4    serverPath: 'http://localhost:80/'
5  }
6
```

Додаток Е

Конфігурація Docker Compose файлу

docker-compose.yml

C: > Users > Mykhailo Novikov > Documents > GitHub > docker-compose.yml

```yaml
version: '3.9'

services:
  db:
    image: mcr.microsoft.com/mssql/server:2019-latest
    container_name: sql_server
    environment:
      SA_PASSWORD: "prepodPortalDbPassword2023"
      ACCEPT_EULA: "Y"
    ports:
      - "1433:1433"
    networks:
      - app-network
  webapi:
    build:
      context: prepod-portal-api/PrepodPortal
      dockerfile: PrepodPortal.WebAPI/Dockerfile
    ports:
      - "80:80"
    depends_on:
      - db
    networks:
      - app-network
  angular:
    build:
      context: prepod-portal-webapp
      dockerfile: Dockerfile
    ports:
      - "81:81"
    networks:
      - app-network

networks:
  app-network:
    driver: bridge
```

Додаток Ж

Кодекс академічної доброчесності здобувача вищої освіти Херсонського державного університету

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Новіков Михайло Миколайович,
учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:
– дотримуватися:
· вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
· принципів та правил академічної доброчесності;
· нульової толерантності до академічного плагіату;
· моральних норм та правил етичної поведінки;
· толерантного ставлення до інших;
· дотримуватися високого рівня культури спілкування;
– надавати згоду на:
· безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
· оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
· використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;
– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;
– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;
– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;
– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;
– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;
– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;
– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;
– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;
– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науководослідницькі завдання;
– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;
– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;
– не підроблювати документи;
– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;
– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;
– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;
– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;
– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;
– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;
– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.
УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

12.09.2019		Михайло Новіков
(дата)	(підпис)	(ім'я, прізвище)
