МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МЕЛЬНИК ДМИТРО ВОЛОДИМИРОВИЧ

Допускається до захисту:
в. о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____О.В. ЗЕЛІНСЬКА
«____»_____ 2024 р.

ІНФОРМАЦІЙНА СИСТЕМА СУПРОВОДУ ОСВІТНЬОГО ПРОЦЕСУ ЗАКЛАДІВ ВИЩОЇ ОСВІТИ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
О.В. Зелінська, доцент кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____

Оцінка: ___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2024

АНОТАЦІЯ

Мельник Д. В. Інформаційна система супроводу освітнього процесу закладів вищої освіти. Спеціальність 122 «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (магістерській) роботі досліджено розробку та процеси обробки даних клієнт-серверного веб-додатку, який автоматизує процес оформлення індивідуального графіку здобувача для поліпшення організації робочого процесу та зручності користувачів. У ході дослідження розроблено та реалізовано багаторівневу систему, яка забезпечує автоматизацію навчальних і адміністративних процесів, включаючи управління користувачами, створення документів, моніторинг успішності студентів та доступ до освітніх ресурсів.

Ключові слова: автоматизація, архітектура клієнт-сервер, база даних, C#, .NET Core, Angular, TypeScript, PostgreSQL

79 с., 0 табл., 23 рис., 6 дод., 51 джерело.

ANOTATION

Melnyk D. V. Information system for supporting the educational process in higher education institutions. Specialty 122 "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia, 2024.

In the qualification (master's) thesis, the development and data processing processes of a client-server web application that automates the creation of individual schedules for students to improve workflow organization and user convenience were investigated. During the research, a multi-level system was developed and implemented, providing automation of educational and administrative processes, including user management, document generation, student performance monitoring, and access to educational resources.

Keywords: automation, client-server architecture, database, C#, .NET Core, Angular, TypeScript, PostgreSQL

Зміст

ВСТУП

Актуальність теми дослідження визначається сучасними тенденціями автоматизації, які охоплюють усі аспекти людської діяльності, включаючи сферу освіти. У зв'язку з цим розробка інформаційної системи супроводу освітнього процесу є важливим кроком у забезпеченні доступності освітніх ресурсів, підвищенні якості навчання та ефективності управління закладом вищої освіти.

Об’єктом дослідження є інформаційна система для управління освітнім процесом у закладах вищої освіти. Вивчення цього дозволяє виявити ключові переваги, ризики та обмеження, які виникають у ході інтеграції цифрових рішень в освітню інфраструктуру.

Предметом дослідження є конкретні компоненти та технології, що використовуються у розробці інформаційної системи супроводу освітнього процесу. Це включає в себе розробку клієнт-серверної архітектури, створення веб-інтерфейсу для управління освітнім контентом та забезпечення ефективного зберігання даних у базі даних.

Мета дослідження полягає у вивченні процесу обробки даних та створенні інформаційної системи, яка дозволяє автоматизувати адміністративні та навчальні процеси, підвищити їх ефективність, а також забезпечити зручний доступ до освітніх ресурсів усім учасникам навчального процесу.

Було визначено наступні завдання дослідження: вивчити сучасні підходи до обробки даних та розробки інформаційних систем для освітніх установ, розробити архітектуру клієнт-серверної системи, що відповідає потребам користувачів, реалізувати програмні модулі для управління освітнім контентом та збереження даних, забезпечити захист інформації від несанкціонованого доступу, оцінити ефективність розробленої системи в реальних умовах закладу вищої освіти.

РОЗДІЛ 1
ОГЛЯД ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Існуючі інформаційні системи для супроводу освітнього процесу

Існуючі інформаційні системи для супроводу освітнього процесу охоплюють широкий спектр рішень, призначених для автоматизації та покращення різних аспектів навчальної діяльності. Вони забезпечують ефективне управління навчальним процесом, підтримку комунікації, доступ до навчальних ресурсів, моніторинг успішності та багато іншого. Нижче розглянемо три найпопулярніші і найбільш поширені інформаційні системи, а також вебсайти університетів України як важливий компонент інформаційних систем.

Moodle (Modular Object-Oriented Dynamic Learning Environment) є однією з найпопулярніших платформ для управління навчальним процесом. Це відкрите програмне забезпечення, яке використовується у багатьох університетах та коледжах по всьому світу[1]. Основні характеристики і переваги Moodle включають:

1. Відкритий вихідний код: Moodle є безкоштовною платформою з відкритим вихідним кодом, що дозволяє користувачам адаптувати і розширювати систему відповідно до своїх потреб.
2. Гнучкість і масштабованість: Moodle підтримує широкий спектр функцій і може бути налаштована для будь-яких навчальних потреб – від невеликих курсів до масштабних освітніх програм.
3. Інтерактивність: Moodle забезпечує інтерактивні інструменти для комунікації та співпраці, такі як форуми, чати, вікі, глосарії та багато іншого.
4. Підтримка різних типів навчальних ресурсів: платформа дозволяє завантажувати і організовувати різноманітні навчальні матеріали, включаючи текстові документи, відео, аудіо, зображення та інтерактивні вправи.

5. Тестування і оцінювання: Moodle надає інструменти для створення тестів і опитувань, автоматичного оцінювання та аналізу результатів.
6. Інтеграція з іншими системами: Moodle може бути інтегрована з іншими освітніми системами та інструментами, такими як системи управління студентами (SIS), бібліотечні системи та інші.

Blackboard є комерційною платформою для управління навчальним процесом, яка широко використовується у вищих навчальних закладах[2]. Основні характеристики і переваги Blackboard включають:

1. Повний набір інструментів для навчання: Blackboard надає широкий спектр інструментів для створення і управління курсами, тестування, оцінювання, комунікації та співпраці.
2. Підтримка мобільних пристроїв: Blackboard має мобільний додаток, який дозволяє студентам і викладачам отримувати доступ до курсів і ресурсів з будь-якого місця.
3. Аналітика і звітність: платформа забезпечує потужні аналітичні інструменти для моніторингу успішності студентів, аналізу даних і створення звітів.
4. Інтеграція з іншими системами: Blackboard підтримує інтеграцію з різними системами і сервісами, такими як бібліотечні системи, системи управління студентами, інструменти для відеоконференцій та інші.
5. Безпека і відповідність стандартам: Blackboard забезпечує високий рівень безпеки даних і відповідає міжнародним стандартам і вимогам щодо захисту даних.

Canvas є хмарною платформою для управління навчальним процесом, яка швидко завойовує популярність завдяки своїй зручності використання і широким можливостям[3]. Основні характеристики і переваги Canvas включають:

1. Інтуїтивний інтерфейс: Canvas має зручний і інтуїтивно зрозумілий інтерфейс, що полегшує роботу з платформою для студентів і викладачів.

2. Підтримка мобільних пристроїв: Canvas має повнофункціональні мобільні додатки для iOS і Android, що дозволяє користувачам отримувати доступ до курсів і ресурсів з будь-якого місця.
3. Інтеграція з іншими системами: Canvas підтримує інтеграцію з багатьма освітніми інструментами і сервісами, такими як Google Apps, Microsoft Office 365, системи управління студентами і інші.
4. Аналітика і звітність: Canvas надає потужні інструменти для аналізу даних і створення звітів, що допомагає викладачам і адміністраторам моніторити успішність студентів і приймати обґрунтовані рішення.
5. Підтримка інноваційних методик навчання: Canvas підтримує різні інноваційні методики навчання, такі як змішане навчання, перевернуті класи, адаптивне навчання і інші.

Вебсайти університетів України є важливими інформаційними системами, які виконують численні функції, спрямовані на підтримку освітнього процесу. Основні характеристики і переваги вебсайтів університетів включають:

1. Інформаційний ресурс: вебсайти надають детальну інформацію про університет, включаючи історію, місію, структуру, академічні програми, викладачів, новини та події. Це допомагає студентам, викладачам і громадськості отримувати актуальну інформацію про діяльність університету. Прикладом є вебсайт Вінницького національного технічного університету (ВНТУ), який надає вичерпну інформацію про факультети, спеціальності, новини та події університету.
2. Портал для абітурієнтів: вебсайти університетів містять інформацію про вступні вимоги, правила прийому, спеціальності, програми підготовки, що є важливим ресурсом для абітурієнтів. Вони також можуть містити онлайн-форми для подання заявок на вступ. Наприклад, вебсайт Донецького національного університету (ДонНУ) надає детальну інформацію про вступні кампанії, спеціальності та умови вступу.

3. Доступ до навчальних ресурсів: багато університетських вебсайтів надають доступ до електронних бібліотек, наукових статей, підручників, лекцій і інших навчальних матеріалів. Це сприяє зручності доступу до знань і підтримує навчальний процес. Наприклад, вебсайт Київського національного університету імені Тараса Шевченка пропонує доступ до численних навчальних ресурсів і бібліотечних фондів.
4. Система управління навчальним процесом: деякі університетські вебсайти інтегровані з LMS, що дозволяє студентам і викладачам використовувати вебсайт як портал для доступу до онлайн-курсів, завдань, тестів, оцінок та іншої навчальної діяльності.
5. Комунікаційний канал: вебсайти університетів забезпечують ефективну комунікацію між студентами, викладачами і адміністрацією через оголошення, новини, контактні форми, форуми та інші інструменти для взаємодії.
6. Адміністративні функції: вебсайти можуть підтримувати адміністративні процеси, такі як реєстрація на курси, управління академічними записами, подання заявок на стипендії і гранти, що полегшує роботу адміністративного персоналу і підвищує ефективність управління.

1.2 Методи розробки інформаційних систем

Розробка інформаційних систем є складним і багатоетапним процесом, який включає в себе проектування, розробку, тестування та впровадження системи. Існує декілька методів та підходів до розробки інформаційних систем, кожен з яких має свої особливості, переваги та недоліки. Нижче наведені найпоширеніші методи розробки інформаційних систем.

Каскадний метод (Waterfall) є одним з найстаріших і найбільш відомих підходів до розробки програмного забезпечення[4]. Він передбачає послідовне

виконання етапів проекту, де кожен етап повинен бути завершений перед переходом до наступного. Основні етапи каскадного методу включають:

1. Збір і аналіз вимог: на цьому етапі визначаються вимоги до системи, включаючи функціональні та нефункціональні вимоги. Всі вимоги документуються і погоджуються з замовником.
2. Проектування системи: на основі вимог розробляється архітектура системи, включаючи детальні технічні специфікації. Це включає проектування бази даних, інтерфейсу користувача, модулів системи та їх взаємодії.
3. Реалізація: на цьому етапі розробники пишуть код відповідно до технічних специфікацій, створених на етапі проектування. Реалізація може включати написання нових програм або модулів, а також інтеграцію з існуючими системами.
4. Тестування: після завершення реалізації система проходить різноманітні види тестування, включаючи модульне, інтеграційне, системне і приймальне тестування. Метою тестування є виявлення і виправлення помилок, а також перевірка відповідності системи вимогам.
5. Впровадження: після успішного тестування система впроваджується в експлуатацію. Це може включати встановлення програмного забезпечення на користувацьких комп'ютерах, навчання користувачів та надання підтримки на початкових етапах експлуатації.
6. Експлуатація і супровід: після впровадження система переходить у фазу експлуатації, де вона використовується користувачами. У разі виникнення проблем або необхідності внесення змін, проводиться супровід системи.

Спіральна модель (Spiral Model) є ітеративним підходом до розробки програмного забезпечення, який поєднує в собі елементи каскадного методу і прототипування[5]. Спіральна модель передбачає розбиття проекту на серію

циклів, кожен з яких включає планування, аналіз ризиків, розробку і тестування. Основні етапи спіральної моделі включають:

1. Планування: на цьому етапі визначаються цілі і завдання для поточної ітерації, а також ресурси і терміни виконання.
2. Аналіз ризиків: на кожній ітерації проводиться аналіз можливих ризиків, які можуть вплинути на проект. Розробляються стратегії для мінімізації або усунення цих ризиків.
3. Розробка: виконується проектування і реалізація компонентів системи, передбачених для поточної ітерації. Це може включати створення прототипів для оцінки концепцій і отримання зворотного зв’язку від користувачів.
4. Тестування: на цьому етапі виконуються тестування компонентів, розроблених на поточній ітерації. Виявлені помилки виправляються, а система вдосконалюється.
5. Оцінка: після завершення кожної ітерації проводиться оцінка результатів і приймається рішення про перехід до наступної ітерації або завершення проекту.

Аджайл (Agile) є гнучким підходом до розробки програмного забезпечення, який фокусується на ітеративній розробці, інкрементальному впровадженні функціоналу і регулярному отриманні зворотного зв’язку від користувачів[6]. Основні принципи Аджайл включають:

1. Ітеративний підхід: розробка програмного забезпечення розбивається на короткі ітерації, зазвичай тривалістю від двох до чотирьох тижнів. Кожна ітерація завершується створенням робочого продукту.
2. Інкрементальна розробка: на кожній ітерації додається новий функціонал до вже існуючої системи. Це дозволяє поступово нарощувати функціональність і отримувати зворотний зв’язок від користувачів на ранніх етапах розробки.
3. Активна взаємодія з користувачами: команда розробників регулярно взаємодіє з користувачами і замовниками для уточнення вимог і

отримання зворотного зв’язку. Це дозволяє швидко реагувати на зміни вимог і адаптувати систему під потреби користувачів.

4. Самоорганізовані команди: команди розробників в Аджайл є самоорганізованими і відповідальними за виконання завдань. Це сприяє підвищенню мотивації і ефективності роботи команди.
5. Постійне вдосконалення: команди регулярно аналізують свою роботу і впроваджують зміни для підвищення продуктивності і якості розробки.

DevOps – це підхід, який поєднує розробку (Development) і операції (Operations) для забезпечення безперервної інтеграції і доставки програмного забезпечення[7]. Основні принципи DevOps включають:

1. Автоматизація процесів: автоматизація процесів розгортання, тестування і моніторингу дозволяє швидко і надійно впроваджувати зміни і оновлення програмного забезпечення.
2. Безперервна інтеграція (CI): зміни в коді регулярно інтегруються в основну гілку розробки, що дозволяє виявляти і виправляти помилки на ранніх етапах.
3. Безперервна доставка (CD): нові версії програмного забезпечення автоматично проходять всі етапи розгортання і тестування, що дозволяє швидко впроваджувати їх в експлуатацію.
4. Співпраця між командами: DevOps сприяє тісній співпраці між командами розробників, тестувальників і операційних інженерів, що забезпечує більш ефективне і швидке вирішення проблем.
5. Моніторинг і зворотний зв’язок: постійний моніторинг роботи системи і отримання зворотного зв’язку від користувачів дозволяють оперативно виявляти і усувати проблеми, підвищуючи якість і стабільність програмного забезпечення.

RAD (Rapid Application Development) – це підхід до швидкої розробки програмного забезпечення, який акцентує увагу на швидкому створенні прототипів і інтенсивній взаємодії з користувачами[8]. Основні принципи RAD включають:

1. Швидке створення прототипів: розробка програмного забезпечення починається з швидкого створення прототипів, які демонструють основні функціональні можливості системи. Це дозволяє отримати зворотний зв’язок від користувачів на ранніх етапах розробки.
2. Інтенсивна взаємодія з користувачами: розробники тісно співпрацюють з користувачами протягом всього циклу розробки, що дозволяє швидко реагувати на зміни вимог і вдосконалювати систему відповідно до потреб користувачів.
3. Ітеративний процес: розробка здійснюється ітеративно, з поступовим додаванням нового функціоналу і поліпшенням вже існуючих компонентів системи.
4. Використання готових компонентів: RAD передбачає використання готових програмних компонентів і інструментів для прискорення процесу розробки і зниження витрат.

Метод прототипування передбачає створення прототипів системи на ранніх етапах розробки для виявлення і уточнення вимог[9]. Основні етапи прототипування включають:

1. Збір вимог: розробники збирають початкові вимоги до системи від користувачів і замовників.
2. Створення прототипу: на основі зібраних вимог створюється початковий прототип системи, який демонструє основні функціональні можливості.
3. Оцінка прототипу: користувачі оцінюють прототип і надають зворотний зв’язок, що дозволяє виявити недоліки і уточнити вимоги.
4. Удосконалення прототипу: на основі зворотного зв’язку прототип доопрацьовується і вдосконалюється.
5. Розробка остаточної системи: після доопрацювання прототипу розпочинається розробка остаточної версії системи, з урахуванням всіх уточнених вимог.

Scrum є однією з популярних методологій Agile, яка фокусується на ітеративній розробці і постійному вдосконаленні продукту[10]. Основні компоненти Scrum включають:

1. Спринти: короткі, зазвичай дво-три тижневі ітерації, на яких команда працює над досягненням конкретних цілей.
2. Ролі в Scrum:
    a. Scrum Master: відповідальний за підтримку процесу Scrum і усунення перешкод для команди.
    b. Product Owner: представляє інтереси замовника і відповідає за управління продуктом.
    c. Розробницька команда: група фахівців, яка працює над реалізацією продукту.
3. Зустрічі в Scrum:
    a. Планування спринту: визначення цілей і завдань для поточного спринту.
    b. Щоденні зустрічі (Daily Standup): короткі щоденні зустрічі для обговорення прогресу і проблем.
    c. Огляд спринту (Sprint Review): демонстрація результатів спринту замовнику.
    d. Ретроспектива спринту (Sprint Retrospective): обговорення підсумків спринту і пошук шляхів для покращення процесу.

XP (Extreme Programming) є ще однією методологією Agile, яка акцентує увагу на високій якості програмного забезпечення і швидкій адаптації до змін[11]. Основні принципи XP включають:

1. Парне програмування: розробники працюють парами, що підвищує якість коду і дозволяє швидко вирішувати проблеми.
2. Постійне тестування: тестування проводиться на всіх етапах розробки, що дозволяє виявляти і виправляти помилки на ранніх стадіях.
3. Простий дизайн: проектування системи здійснюється з урахуванням максимальної простоти і гнучкості.

4. Регулярні релізи: програмне забезпечення регулярно випускається в експлуатацію, що дозволяє швидко отримувати зворотний зв’язок від користувачів.
5. Спільне володіння кодом: всі члени команди мають доступ до коду і можуть вносити зміни, що забезпечує швидку адаптацію до нових вимог.

1.3 Важливість інформаційних систем для освітнього процесу

Інформаційні системи (ІС) відіграють ключову роль у сучасному освітньому процесі. Вони сприяють підвищенню ефективності навчання, оптимізації управлінських процесів, забезпеченню доступу до знань та покращенню якості освіти в цілому[12]. В цій частині розглянемо детальніше, чому інформаційні системи є такими важливими для освітнього процесу, як вони впливають на різні аспекти освіти та які переваги вони надають.

Інформаційні системи значно спрощують і автоматизують адміністративні процеси в освітніх установах. Завдяки впровадженню ІС, адміністративний персонал може ефективніше управляти реєстрацією студентів, складанням розкладів, веденням документації, управлінням навчальними планами та іншими завданнями. Основні переваги оптимізації адміністративних процесів включають:

1. Зниження навантаження на адміністративний персонал: автоматизація рутинних завдань дозволяє зменшити обсяг ручної роботи і звільнити час для виконання більш складних і творчих завдань.
2. Зменшення кількості помилок: використання ІС для автоматизації процесів допомагає уникнути помилок, пов'язаних з ручним введенням даних, що підвищує точність і надійність адміністративних операцій.
3. Підвищення прозорості та контрольованості: ІС забезпечують зручний доступ до актуальної інформації про студентів, викладачів, розклади та інші аспекти освітнього процесу, що полегшує контроль і управління.

Інформаційні системи роблять навчальні ресурси більш доступними для студентів та викладачів. Системи управління навчальним процесом (LMS) дозволяють зберігати і організовувати навчальні матеріали в електронному форматі, що забезпечує зручний доступ до них у будь-який час і з будь-якого місця[13]. Основні переваги підвищення доступності навчальних ресурсів включають:

1. Доступність 24/7: студенти можуть отримувати доступ до лекцій, підручників, презентацій, відеоматеріалів та інших ресурсів у будь-який час, що сприяє гнучкості навчання і дозволяє їм вчитися у власному темпі.
2. Зручний пошук і організація матеріалів: ІС дозволяють легко знаходити необхідні навчальні матеріали, організовувати їх за темами, курсами, датами тощо, що сприяє ефективнішому навчанню.
3. Інтерактивність і мультимедійність: ІС підтримують використання інтерактивних і мультимедійних ресурсів, таких як відео, аудіо, анімації, інтерактивні вправи, що робить навчання більш цікавим і ефективним.

Інформаційні системи значно покращують комунікацію між студентами, викладачами та адміністративним персоналом. Вони забезпечують різні інструменти для обміну інформацією і співпраці, включаючи електронну пошту, чати, форуми, відеоконференції та інші. Основні переваги покращення комунікації та співпраці включають:

1. Швидка і ефективна комунікація: ІС дозволяють швидко обмінюватися інформацією, задавати питання і отримувати відповіді, що сприяє оперативному вирішенню проблем і питань.
2. Підтримка дистанційного навчання: ІС забезпечують проведення онлайн-лекцій, вебінарів, семінарів і відеоконференцій, що дозволяє організувати навчання незалежно від місця знаходження студентів і викладачів.

3. Спільна робота над проектами: ІС надають інструменти для спільної роботи над проектами, що дозволяє студентам і викладачам ефективно співпрацювати і обмінюватися ідеями.

Інформаційні системи дозволяють забезпечити індивідуальний підхід до навчання кожного студента. Вони надають можливість створення індивідуальних навчальних планів, відстеження успішності і прогресу студентів, а також надання персоналізованих рекомендацій і зворотного зв'язку. Основні переваги підтримки індивідуального навчання включають:

1. Адаптація навчальних матеріалів: ІС дозволяють адаптувати навчальні матеріали і завдання під рівень знань і потреби кожного студента, що сприяє ефективнішому засвоєнню матеріалу.
2. Відстеження прогресу: ІС надають можливість викладачам і студентам відстежувати прогрес у навчанні, аналізувати результати тестів, оцінки і відвідуваність, що дозволяє вчасно виявляти проблеми і вживати заходів для їх усунення.
3. Персоналізовані рекомендації: на основі аналізу даних про успішність і прогрес студентів, ІС можуть надавати персоналізовані рекомендації щодо подальшого навчання, що допомагає студентам досягати кращих результатів.

Інформаційні системи надають важливу підтримку для прийняття управлінських рішень в освітніх установах. Вони забезпечують доступ до актуальної і точної інформації, а також аналітичні інструменти для аналізу даних і оцінки ефективності навчального процесу. Основні переваги підтримки прийняття управлінських рішень включають:

1. Оперативний доступ до даних: ІС надають керівництву закладів освіти оперативний доступ до даних про студентів, викладачів, навчальні плани, ресурси та інші аспекти освітнього процесу, що дозволяє швидко отримувати необхідну інформацію для прийняття рішень.
2. Аналітика і звітність: ІС забезпечують інструменти для аналізу даних і створення звітів, що допомагає керівництву оцінювати ефективність

навчального процесу, виявляти проблемні області і приймати обґрунтовані рішення.

3. Прогнозування і планування: на основі аналізу даних ІС можуть допомагати керівництву закладів освіти прогнозувати майбутні потреби і планувати ресурси, включаючи кількість студентів, необхідну кількість викладачів, матеріальні ресурси та інше.

Впровадження інформаційних систем сприяє підвищенню якості освіти за рахунок автоматизації процесів, підвищення ефективності навчання і забезпечення доступу до якісних навчальних матеріалів. Основні переваги підвищення якості освіти включають:

1. Покращення методик навчання: ІС дозволяють впроваджувати сучасні методики навчання, включаючи дистанційне навчання, змішане навчання, адаптивне навчання та інші, що сприяє підвищенню якості освіти.
2. Забезпечення якості навчальних матеріалів: ІС забезпечують доступ до якісних і актуальних навчальних матеріалів, що сприяє кращому засвоєнню знань студентами.
3. Підвищення мотивації студентів: використання ІС для надання інтерактивних і мультимедійних навчальних матеріалів, а також персоналізованих рекомендацій і зворотного зв’язку, підвищує мотивацію студентів до навчання.

Інформаційні системи забезпечують захист інформаційних ресурсів від несанкціонованого доступу, втрати або пошкодження. Це особливо важливо в умовах зростаючих кіберзагроз і необхідності захисту персональних даних студентів і викладачів. Основні переваги забезпечення безпеки даних включають:

1. Захист персональних даних: ІС забезпечують захист персональних даних студентів і викладачів, включаючи їхні особисті дані, результати навчання, фінансову інформацію та інше.

2. Запобігання втраті даних: використання ІС дозволяє зменшити ризик втрати даних за рахунок регулярного резервного копіювання і відновлення даних у разі аварійних ситуацій.
3. Контроль доступу: ІС забезпечують контроль доступу до даних і ресурсів, що дозволяє обмежити доступ до конфіденційної інформації тільки авторизованим користувачам.

1.4 Постановка задачі роботи

Розробка веб-додатку для інформаційної системи закладу вищої освіти включає використання Angular для фронтенду, .NET (ASP.NET Core) для бекенду та PostgreSQL для бази даних. Основними функціями системи є автоматизація процесу оформлення індивідуального графіку студентів, що дозволяє студентам подавати запити на створення графіків. Крім того, система полегшує ознайомлення з інформацією кафедри, включаючи відображення новин та оголошень, списки викладачів із контактною інформацією та розкладом консультацій, а також інформацію про курси. Модуль для адміністрування включає управління користувачами, права доступу та діяльність користувачів. Архітектура додатку передбачає використання компонентів Angular для відображення модулів системи, сервісів для взаємодії з бекендом через API, контролерів .NET Core для обробки запитів від фронтенду, сервісів для бізнес-логіки та репозиторіїв для взаємодії з базою даних PostgreSQL, яка міститиме таблиці для зберігання інформації про студентів, викладачів, курси, розклади та інші дані.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ СУПРОВОДУ ОСВІТНЬОГО ПРОЦЕСУ ЗАКЛАДІВ ВИЩОЇ ОСВІТИ

2.1 Загальні положення про розроблювану ІС

Інформаційна система супроводу освітнього процесу закладів вищої освіти призначена для автоматизації та покращення управління ключовими аспектами освітнього процесу. Основною метою системи є забезпечення ефективного управління студентськими даними, документообігом, аналітикою навчального процесу та взаємодією між учасниками освітнього процесу — адміністрацією, викладачами та студентами. Розробка цієї системи спирається на сучасні технології, такі як .NET Web API[14], PostgreSQL[15] та Angular[16], що забезпечують високий рівень продуктивності, надійності та можливість масштабування для розширення функціональності в майбутньому. Система спрямована на покращення організаційної роботи закладу вищої освіти через автоматизацію рутинних завдань, полегшення доступу до інформації та зменшення помилок, викликаних людським фактором.

Основними завданнями інформаційної системи є:

- Автоматизація адміністративних процесів: система повинна допомагати автоматизувати рутинні завдання, такі як управління розкладами, реєстрація студентів, обробка документів.
- Підтримка навчального процесу: забезпечення студентам і викладачам доступу до інформації про курси, оцінки, академічні звіти, навчальні матеріали та інші ресурси.
- Забезпечення аналітики та моніторингу: система повинна дозволяти адміністрації ЗВО аналізувати академічні результати, успішність студентів, ефективність викладання, а також моніторити діяльність кафедр.

Інформаційна система охоплює кілька ключових модулів, які забезпечують реалізацію її функціональних можливостей:

- Модуль управління студентськими даними: зберігає інформацію про студентів, включаючи їхні особисті дані, академічну історію, відвідуваність та успішність.
- Модуль документообігу: дозволяє зберігати, шукати та обробляти офіційні документи, такі як накази, звіти, навчальні плани.
- Модуль аналітики та звітності: забезпечує збирання й аналіз даних для створення статистичних звітів про успішність студентів, діяльність викладачів і загальний стан навчального процесу.

Інформаційна система складається з кількох основних компонентів, кожен з яких відіграє важливу роль в її функціонуванні:

- Клієнтська частина: представлена веб-інтерфейсом, за допомогою якого користувачі (студенти, викладачі, адміністрація) взаємодіють із системою. Вона забезпечує доступ до даних та функцій системи через браузер.
- Серверна частина: відповідає за обробку запитів від клієнтської частини, управління бізнес-логікою та роботу з базою даних. Бекенд реалізований на основі архітектури REST API[17], яка дозволяє взаємодіяти з клієнтами через HTTP-запити[18].
- База даних: це центральне сховище всіх даних системи. Вона зберігає інформацію про студентів, викладачів, курси, оцінки, документи та інші ключові елементи. Використовується реляційна база даних PostgreSQL, яка забезпечує високий рівень надійності, масштабованості та продуктивності.

Інформаційна система повинна відповідати низці вимог, що забезпечують її ефективну роботу:

- Продуктивність: система повинна обробляти великі обсяги даних і запитів в реальному часі без затримок.

- Масштабованість: можливість збільшення навантаження на систему (зростання кількості користувачів, даних тощо) без погіршення її продуктивності.
- Безпека: захист персональних даних студентів та викладачів від несанкціонованого доступу, забезпечення конфіденційності інформації, а також впровадження механізмів шифрування та багаторівневої авторизації.
- Надійність: система повинна бути стабільною і працювати без збоїв. Важливо забезпечити регулярне резервне копіювання даних і можливість швидкого відновлення після аварій.
- Доступність: користувачі повинні мати доступ до системи 24/7 з будь-яких пристроїв (комп'ютери, планшети, смартфони).

Основними користувачами інформаційної системи є:

- Студенти: отримують доступ до своїх академічних записів, успішності, розкладу занять, а також можуть переглядати повідомлення і отримувати сповіщення від викладачів.
- Викладачі: керують навчальними курсами, завантажують навчальні матеріали, вводять оцінки та формують звіти про успішність студентів.
- Адміністрація: має доступ до аналітичних даних, займається управлінням академічним процесом, підготовкою звітів, а також обробкою і керуванням документами.

Інформаційна система значно спрощує і прискорює процес управління навчальним процесом, забезпечуючи такі переваги:

- Оптимізація часу: завдяки автоматизації рутинних процесів викладачі та адміністрація можуть сконцентруватися на більш важливих завданнях.
- Підвищення ефективності: за рахунок зручного доступу до даних і аналітики можна своєчасно реагувати на зміни в успішності студентів та загальному стані навчального процесу.

- Поліпшення комунікації: система забезпечує швидку і прозору взаємодію між студентами, викладачами та адміністрацією через повідомлення, сповіщення та електронні документи.

2.2 Архітектура розроблюваної інформаційної системи

Архітектура інформаційної системи є однією з ключових частин проектування, оскільки вона визначає, як компоненти системи взаємодіють між собою, як обробляються дані та забезпечується робота з кінцевими користувачами. Інформаційна система для супроводу освітнього процесу у закладах вищої освіти спроектована з використанням багаторівневої архітектури, що дозволяє забезпечити масштабованість, безпеку та ефективність.

Багаторівнева архітектура (multitier architecture) — це структура системи, де функціональні можливості та логіка розділені на кілька рівнів[19]. Це дозволяє зменшити складність і підвищити масштабованість системи. Основними рівнями такої архітектури є:

- Рівень презентації (Presentation Layer): цей рівень відповідає за взаємодію користувача із системою, зокрема за відображення інтерфейсу та отримання введених даних.
- Рівень бізнес-логіки (Business Logic Layer): містить правила та логіку, що визначають поведінку системи відповідно до бізнес-процесів.
- Рівень даних (Data Layer): відповідає за доступ до бази даних, збереження та отримання інформації.

Інформаційна система розробляється за класичною клієнт-серверною моделлю, де клієнтська частина виконується на комп'ютері або іншому пристрої користувача, а серверна частина знаходиться на сервері або в хмарі. Така архітектура забезпечує поділ завдань і підвищує продуктивність та масштабованість системи. Клієнт (студенти, викладачі, адміністрація) відправляє запити на сервер для отримання або відправки даних (наприклад,

перегляд розкладу занять, введення оцінок, завантаження документів). Сервер обробляє ці запити, взаємодіє з базою даних і повертає необхідну інформацію.

Клієнтська частина інформаційної системи відповідає за відображення інтерфейсу користувачам і отримання введених даних. Вона реалізована за допомогою Angular, сучасного фреймворку для створення веб-застосунків. Основні завдання клієнтської частини включають:

- Відображення даних: клієнтська частина отримує інформацію з сервера через REST API і відображає її у відповідному вигляді (таблиці, форми, графіки).
- Введення даних: користувачі можуть вводити дані (наприклад, додавати нові документи, оцінки, інформацію про курси) через форми, які надсилають запити на сервер для збереження в базі даних.
- Адаптивний дизайн: Angular забезпечує створення інтерфейсів, які адаптуються під різні пристрої, зокрема комп'ютери, планшети та смартфони. Це дозволяє студентам і викладачам працювати із системою з будь-якого місця.

Рівень бізнес-логіки (серверна частина) відповідає за реалізацію бізнес-правил і обробку запитів від клієнтської частини. Він реалізований за допомогою ASP.NET Core[20], що є потужним серверним фреймворком для створення веб-додатків і REST API. Основні компоненти рівня бізнес-логіки:

- Контролери: відповідають за прийом HTTP-запитів від клієнта. Контролери передають запити до сервісів, які реалізують бізнес-логіку.
- Сервіси: реалізують основні функції та правила системи. Наприклад, якщо система отримує запит на оновлення оцінок студента, сервіс відповідає за перевірку прав доступу, валідацію даних і передачу запиту до бази даних через репозиторій.
- Авторизація та аутентифікація: використання JWT (JSON Web Token)[21] для аутентифікації користувачів і перевірки прав доступу

до певних ресурсів (наприклад, студенти можуть переглядати свої оцінки, але не змінювати їх).

Серверна частина також відповідає за:

- Обробку запитів та генерацію відповідей: сервер отримує запити від клієнта через HTTP, обробляє їх і повертає відповідь у форматі JSON[22].
- Обробку даних та виконання бізнес-логіки: це може включати перевірку коректності даних, виконання обчислень та прийняття управлінських рішень на основі отриманих запитів.

Рівень даних відповідає за зберігання та управління даними. База даних, яку використовує система, — це реляційна система управління базами даних PostgreSQL. Основні завдання рівня даних включають:

- Зберігання даних: зберігається інформація про студентів, курси, викладачів, розклад, документи та інші дані.
- Запити до бази даних: серверна частина взаємодіє з базою даних через Entity Framework Core[23], що дозволяє виконувати SQL-запити за допомогою об'єктно-орієнтованих підходів.
- Цілісність і безпека даних: база даних підтримує механізми цілісності (наприклад, зовнішні ключі) для запобігання некоректним змінам, а також доступність даних лише для авторизованих користувачів.

Комунікація між рівнями системи здійснюється за допомогою протоколу HTTP і REST API. Клієнт надсилає запити (наприклад, на отримання інформації про курси або оцінки) до серверної частини. Сервер обробляє запит, звертається до бази даних для отримання або запису інформації, після чого повертає відповідь у форматі JSON. REST API забезпечує простоту інтеграції та взаємодії між різними компонентами системи, а також дозволяє масштабувати систему. Завдяки цьому клієнтська та серверна частини можуть працювати незалежно одна від одної, що підвищує гнучкість і продуктивність системи. Перевагами такої архітектури є:

- Масштабованість: багаторівнева архітектура дозволяє розподілити навантаження між різними компонентами системи. Сервери можуть бути збільшені залежно від потреб.
- Гнучкість: завдяки розділенню компонентів можна легко оновлювати або змінювати окремі частини системи без впливу на інші.
- Безпека: використання рівня авторизації та аутентифікації забезпечує контроль доступу до даних.
- Модульність: кожен рівень системи виконує свою чітко визначену функцію, що спрощує підтримку та розширення системи в майбутньому.

Архітектура системи є фундаментом для розробки ефективної інформаційної системи, яка здатна задовольнити потреби різних користувачів — від студентів до адміністраторів, забезпечуючи безперервний доступ до інформації та ресурсів.

2.3 Проектування бази даних розроблюваної інформаційної системи

Проектування бази даних є важливим етапом розробки інформаційної системи, оскільки база даних є центральним сховищем всієї інформації та забезпечує доступ до даних для всіх користувачів системи. У випадку інформаційної системи для супроводу освітнього процесу база даних повинна підтримувати зберігання великих обсягів інформації про студентів, викладачів, курси, документи, результати навчання тощо.

База даних повинна відповідати таким вимогам:

- Цілісність даних: збереження даних має забезпечувати логічну цілісність і коректність записів. Наприклад, кожен студент повинен мати унікальний ідентифікатор, і кожен документ повинен бути правильно пов'язаний із студентом або викладачем.

- Продуктивність: база даних повинна бути оптимізована для швидкого виконання запитів, таких як отримання списку студентів, інформації про курси та оцінки.
- Масштабованість: система повинна підтримувати розширення без втрати продуктивності, оскільки кількість даних (студенти, курси, документи) постійно зростає.
- Безпека: база даних повинна забезпечувати захист персональних даних, таких як результати оцінювання студентів або приватна інформація про викладачів.

Для реалізації бази даних обрано PostgreSQL — потужну реляційну систему управління базами даних з відкритим кодом, яка забезпечує високу надійність, продуктивність і підтримує складні транзакції. PostgreSQL дозволяє:

- Створювати складні структури баз даних з використанням зв'язків між таблицями.
- Підтримувати масштабування і високу продуктивність через оптимізовані індекси та механізми кешування.
- Використовувати тригери, збережені процедури та складні запити для більш ефективної обробки даних.

База даних інформаційної системи складається з декількох основних таблиць, кожна з яких зберігає певний набір даних. Основні таблиці включають:

1. Таблиця студентів (Students):
    - ID (PRIMARY KEY): унікальний ідентифікатор студента.
    - Full_Name: повне ім'я студента.
    - Birth_Date: дата народження студента.
    - Enrollment_Date: дата зарахування до закладу.
    - Major: спеціальність студента.
    - Email: електронна пошта для контактів.
2. Таблиця викладачів (Teachers):
    - ID (PRIMARY KEY): унікальний ідентифікатор викладача.
    - Full_Name: повне ім'я викладача.

- Department: кафедра, до якої належить викладач.
- Position: посада (наприклад, професор, доцент).
- Email: електронна пошта для контактів.

3. Таблиця курсів (Courses):
   - ID (PRIMARY KEY): унікальний ідентифікатор курсу.
   - Course_Name: назва курсу.
   - Teacher_ID (FOREIGN KEY): ідентифікатор викладача, який проводить курс (посилання на таблицю Teachers).
   - Credits: кількість кредитів за курс.
   - Semester: семестр, у якому проводиться курс.
4. Таблиця оцінок (Grades):
   - ID (PRIMARY KEY): унікальний ідентифікатор оцінки.
   - Student_ID (FOREIGN KEY): ідентифікатор студента (посилання на таблицю Students).
   - Course_ID (FOREIGN KEY): ідентифікатор курсу (посилання на таблицю Courses).
   - Grade: оцінка, отримана студентом за курс.
   - Exam_Date: дата проведення іспиту.
5. Таблиця документів (Documents):
   - ID (PRIMARY KEY): унікальний ідентифікатор документа.
   - Document_Type: тип документа (наприклад, звіт, наказ).
   - Document_Name: назва документа.
   - Student_ID (FOREIGN KEY): посилання на студента, якого стосується документ.
   - Upload_Date: дата завантаження документа.

У базі даних передбачені кілька типів зв'язків:

- Один-до-багатьох (One-to-Many):
  - Студент — Оцінки: кожен студент може мати багато оцінок, але кожна оцінка належить одному студенту.

- Викладач — Курси: один викладач може викладати кілька курсів, але кожен курс прив'язаний до одного викладача.

- Багато-до-багатьох (Many-to-Many):
  - Студенти — Курси: студент може бути зарахований на кілька курсів, а кожен курс має кілька студентів. Це досягається через проміжну таблицю "Оцінки", яка пов'язує студентів з курсами.

Для підвищення продуктивності бази даних використовуються індекси, які допомагають швидко знаходити записи без необхідності проходження всієї таблиці[24]. Найчастіше індекси створюються на первинних ключах (ID), а також на полях, які часто використовуються у запитах (наприклад, Email). Додатково використовуються кешування запитів і оптимізація транзакцій для того, щоб мінімізувати час затримки під час звернення до бази даних при великих обсягах даних.

Для забезпечення цілісності даних використовуються такі механізми:

- Первинні ключі (Primary Keys): кожна таблиця має унікальний ідентифікатор (ID), який не може бути порожнім[25].
- Зовнішні ключі (Foreign Keys): забезпечують зв'язки між таблицями і гарантують, що запис у одній таблиці пов'язаний з існуючим записом в іншій[26]. Наприклад, кожна оцінка прив'язана до існуючого студента через поле Student_ID.
- Обмеження (Constraints): використовуються для контролю за допустимістю значень у полях[27]. Наприклад, оцінки мають бути в діапазоні від 0 до 100.

З метою захисту конфіденційної інформації в базі даних передбачені наступні механізми:

- Контроль доступу: доступ до бази даних і операцій з нею надається лише авторизованим користувачам. Кожен користувач має свої права (читання, запис, видалення).

- Шифрування даних: конфіденційні дані, такі як паролі або персональна інформація студентів, шифруються з використанням алгоритмів шифрування.
- Резервне копіювання: система повинна регулярно виконувати резервне копіювання даних, щоб запобігти втратам у разі збоїв або атак.

Транзакції використовуються для того, щоб забезпечити виконання кількох операцій над базою даних як єдиного атомарного процесу[28]. Це означає, що всі операції в межах однієї транзакції або успішно виконуються разом, або відхиляються в разі помилки, забезпечуючи цілісність даних. PostgreSQL підтримує ACID-транзакції, які забезпечують:

- Атомарність (Atomicity): або всі зміни в межах транзакції застосовуються, або жодна[29].
- Цілісність (Consistency): після кожної транзакції база даних залишається в коректному стані[30].
- Ізольованість (Isolation): жодна транзакція не може бачити зміни іншої транзакції, доки та не завершена[31].
- Довговічність (Durability): після завершення транзакції її зміни стають постійними[32].

Проектування бази даних для інформаційної системи є складним процесом, що включає аналіз потреб користувачів, розробку структури таблиць та зв'язків між ними, а також забезпечення безпеки та оптимізації продуктивності. PostgreSQL, як обрана СУБД, надає широкі можливості для реалізації масштабованої та надійної системи, здатної обробляти великі обсяги даних і забезпечувати їхню цілісність.

2.4 Проектування інтерфейсу користувача розроблюваної інформаційної системи

Інтерфейс користувача (UI) є ключовим елементом інформаційної системи, оскільки саме через нього користувачі взаємодіють із системою. Грамотно

спроектований інтерфейс не лише підвищує зручність користування системою, але й позитивно впливає на загальну ефективність її роботи.

Інтерфейс користувача повинен відповідати кільком ключовим вимогам:

- Зручність використання (Usability): інтерфейс повинен бути інтуїтивно зрозумілим для користувачів різних рівнів підготовки (студентів, викладачів, адміністрації). Основна увага приділяється простоті навігації та доступності функцій[33].
- Доступність (Accessibility): інтерфейс повинен бути доступним для людей з обмеженими можливостями, враховуючи стандартні рекомендації доступності (наприклад, WCAG — Web Content Accessibility Guidelines)[34].
- Адаптивність (Responsiveness): інтерфейс повинен адаптуватися під різні розміри екранів і пристрої (комп'ютери, планшети, смартфони), забезпечуючи однаково комфортне користування незалежно від платформи[35].
- Естетичний дизайн (Aesthetic Design): інтерфейс повинен бути візуально привабливим і відповідати сучасним тенденціям дизайну[36].

Проектування інтерфейсу користувача базується на кількох теоретичних концепціях:

- Модель користувача (User Model): під час проектування інтерфейсу враховується модель користувача — тобто уявлення про те, як користувачі сприймають систему та її функції[37]. Важливо розуміти, що користувачі не завжди володіють технічними знаннями, тому інтерфейс має бути простим і зрозумілим для широкого кола людей.
- Ментальна модель (Mental Model): користувачі підходять до нових систем із певними очікуваннями, заснованими на попередньому досвіді користування іншими програмами[38]. Ментальна модель

допомагає передбачити, як користувачі будуть взаємодіяти з інтерфейсом і яких дій від нього очікувати.

- Принципи Гештальта: ці принципи допомагають структурувати інформацію в інтерфейсі таким чином, щоб користувачі легко її сприймали[29]. Основні принципи включають близькість, схожість, замкненість і продовженість. Наприклад, схожі елементи (кнопки одного типу) повинні виглядати однаково.

Інтерфейс користувача складається з кількох ключових сторінок, кожна з яких виконує певні функції:

- Головна сторінка (Dashboard):
  - Відображає оголошення, останні події, інформаційні блоки та швидкий доступ до основних розділів системи.
- Студентський модуль:
  - Відображає інформацію про курси, оцінки, навчальні матеріали та розклад занять.
- Модуль документообігу:
  - Забезпечує можливість пошуку, завантаження та управління документами, такими як навчальні плани, звіти та накази.
- Модуль аналітики:
  - Включає графіки, діаграми та звіти для адміністрації та викладачів.
- Сторінка "Історія кафедри":
  - Містить текстову та візуальну інформацію про історію кафедри, важливі події та розвиток наукової діяльності.
- Сторінка "Колектив кафедри":
  - Відображає перелік викладачів та співробітників кафедри з фотографіями, контактною інформацією та коротким описом їхньої наукової діяльності.
- Сторінка "Абітурієнту":

- Інформація для вступників: перелік спеціальностей, умови вступу, вимоги до документів, важливі дати та контактні дані приймальної комісії.

- Сторінка "Освітні програми":
  - Опис програм навчання, доступні спеціальності, плани курсів та можливості кар’єрного росту для студентів.
- Сторінка "Наукова діяльність":
  - Інформація про наукові дослідження, проекти кафедри, публікації викладачів та участь у наукових конференціях.
- Сторінка "Акредитація":
  - Відомості про акредитацію освітніх програм, сертифікати та відповідність стандартам вищої освіти.
- Сторінка "Співробітництво"
  - Інформація про партнерства кафедри з іншими університетами та організаціями, спільні наукові проекти та програми обміну.
- Сторінка "Контакти":
  - Контактна інформація кафедри, адреса, електронна пошта та телефон для зв'язку з адміністрацією та викладачами.

Оскільки інформаційна система повинна бути доступною для різних пристроїв, важливо забезпечити адаптивність інтерфейсу. Адаптивний дизайн (Responsive Design) забезпечує коректне відображення інтерфейсу на різних розмірах екранів[40], зокрема на:

- Смартфонах: менший екран потребує зменшення кількості елементів на одному екрані, а також використання гнучких сіток (grid) та адаптивних зображень.
- Планшетах: інтерфейс повинен адаптуватися до середніх розмірів екранів, забезпечуючи зручну навігацію.
- Комп'ютерах: повна версія інтерфейсу з усіма функціями повинна бути доступною для користувачів настільних ПК та ноутбуків.

Для цього використовуються такі підходи:

- Медіа-запити (media queries): CSS-правила, що змінюють вигляд інтерфейсу залежно від розміру екрана[41].
- Гнучкі сітки (flexible grids): система сіток, яка дозволяє елементам інтерфейсу автоматично підлаштовуватися під доступний простір[42].
- Мобільна версія інтерфейсу: окремий дизайн для смартфонів, який оптимізує інтерфейс для роботи на невеликих екранах.

Щоб забезпечити високу якість інтерфейсу користувача, необхідно проводити тестування UI/UX. Основні методи тестування включають:

- Юзабіліті-тестування: залучення кінцевих користувачів до використання системи для оцінки її зручності[43]. Це допомагає виявити проблеми на ранніх стадіях розробки.
- А/Б-тестування: порівняння двох версій інтерфейсу для визначення, яка з них є більш ефективною[44].
- Тестування доступності: перевірка інтерфейсу на відповідність стандартам доступності, зокрема для людей із порушеннями зору або рухових функцій[45].

Важливим аспектом проектування інтерфейсу є створення візуальних прототипів та макетів для кращого розуміння того, як виглядатиме і функціонуватиме система. Для цього використовують інструменти для дизайну інтерфейсів, і одним із найпопулярніших є Figma. Figma — це хмарний інструмент для дизайну та прототипування, який дозволяє команді спільно працювати над проектами в реальному часі, тестувати інтерфейси та отримувати зворотний зв’язок на етапі розробки[46]. Figma надає широкий набір функцій для створення ітерацій дизайну та прототипів інтерфейсів:

- Хмарна платформа: усі елементи дизайну зберігаються в хмарі, що дозволяє легко обмінюватися макетами з командою та інтегрувати відгуки від замовників у реальному часі.

- Прототипування: можливість створення інтерактивних прототипів, що імітують поведінку справжнього інтерфейсу без необхідності написання коду.
- Командна робота: кілька дизайнерів і розробників можуть одночасно працювати над макетом і вносити свої зміни.
- Гнучкість у розробці дизайну для різних пристроїв: Figma дозволяє легко адаптувати інтерфейс для різних розмірів екранів — настільних ПК, планшетів і мобільних телефонів.

Процес проектування інтерфейсу в Figma:

1. Використання стилів та компонентів:
    - Компоненти: у Figma можна створювати повторно використовувані компоненти, такі як кнопки, поля введення, картки курсів, що забезпечує єдиний стиль по всій системі та зменшує час на редагування.
    - Стиль шрифтів і кольорів: визначаються глобальні стилі для шрифтів і кольорів, що допомагає підтримувати послідовність в усьому дизайні. Наприклад, всі кнопки можуть бути одного кольору та стилю, що полегшує їх розпізнавання користувачами.
2. Прототипування:
    - Після створення макетів Figma дозволяє додати інтерактивність за допомогою прототипування. Користувачі можуть натискати на кнопки, заповнювати форми й отримувати візуальні відгуки у вигляді переходів між сторінками або спливаючих вікон. Це допомагає створити динамічний прототип, який можна демонструвати замовникам або використовувати для тестування з кінцевими користувачами.
3. Адаптація для різних пристроїв:
    - Mobile-First підхід: важливо, щоб дизайн інтерфейсу був адаптований під різні екрани. Figma дозволяє створювати макети для мобільних пристроїв, використовуючи фрейми різних

розмірів. Наприклад, окремий макет для мобільних телефонів, де деякі елементи можуть бути спрощені або змінені для покращення користувацького досвіду.

- Адаптивність: для настільних ПК, планшетів і смартфонів макети можуть бути адаптовані так, щоб елементи інтерфейсу правильно розміщувалися й залишалися зручними для користування на всіх пристроях.

Етапи розробки дизайну в Figma:

- Збір вимог: Перед початком дизайну необхідно зібрати вимоги від замовників і користувачів, щоб зрозуміти їхні очікування та проблеми, які має вирішити система.
- Створення каркасів (Wireframes): Каркаси є простими макетами, що показують структуру сторінок і розміщення елементів інтерфейсу без деталей дизайну (кольорів, шрифтів). Це дозволяє швидко затвердити основну структуру інтерфейсу.
- Створення детальних макетів: Після затвердження каркасів дизайнер створює детальні макети з повним набором візуальних елементів: шрифтів, кольорів, іконок, зображень тощо.
- Прототипування та тестування: Додавання інтерактивності до макетів і тестування прототипів для отримання відгуків від користувачів та замовників.
- Внесення коригувань та підготовка до розробки: Після тестування макети редагуються, враховуючи відгуки, і передаються до розробників.

Проектування інтерфейсу користувача є складним процесом, що включає в себе врахування вимог до зручності, адаптивності та доступності. Застосування сучасних теоретичних основ і підходів до UI/UX забезпечує створення інтерфейсу, який буде інтуїтивно зрозумілим, зручним у використанні та привабливим для різних категорій користувачів.

2.5 Проектування серверної частини та API розроблюваної інформаційної системи

Серверна частина інформаційної системи є важливою складовою, оскільки відповідає за обробку даних, виконання бізнес-логіки та управління запитами від клієнтської частини. Розглянемо теоретичні аспекти серверної частини, її архітектуру, принципи проектування API та підходи до забезпечення ефективної взаємодії між клієнтом і сервером.

Серверна частина є центром інформаційної системи і забезпечує виконання основних функцій:

- Обробка запитів: сервер отримує запити від клієнтів (фронтенд), обробляє їх і надсилає відповідь. Запити можуть бути HTTP-запитами на отримання, додавання, редагування або видалення даних.
- Управління даними: сервер взаємодіє з базою даних для збереження і отримання інформації, забезпечуючи зберігання і обробку даних.
- Виконання бізнес-логіки: це процес виконання певних дій відповідно до вимог бізнесу. Наприклад, перевірка умов для створення документів, надання доступу на основі прав користувача, обробка транзакцій.

Архітектура серверної частини побудована на трьох основних компонентах:

- Контролери (Controllers): приймають запити від клієнтської частини і передають їх до відповідних сервісів для обробки.
- Сервіси (Services): реалізують бізнес-логіку і виконують основні функції, такі як обробка даних, перевірка правил або розрахунки.
- Репозиторії (Repositories): забезпечують взаємодію з базою даних, абстрагуючи роботу з даними від бізнес-логіки.

REST (Representational State Transfer) — це архітектурний стиль для побудови веб-сервісів, що забезпечує ефективну взаємодію між клієнтом і сервером через протокол HTTP. API (Application Programming Interface) дозволяє

фронтенду взаємодіяти із серверною частиною для отримання та обробки даних. Основні принципи REST API:

- Безстанова взаємодія (Statelessness): кожен запит від клієнта до сервера містить усю необхідну інформацію для обробки цього запиту. Сервер не зберігає стан між запитами, що робить REST більш гнучким і масштабованим.
- Єдність інтерфейсу (Uniform Interface): всі ресурси сервера доступні через стандартизовані методи HTTP (GET, POST, PUT, DELETE). Наприклад:
  - GET: отримання даних (наприклад, отримати список студентів).
  - POST: створення нових даних (наприклад, додати новий документ).
  - PUT: оновлення даних (наприклад, редагувати інформацію про курс).
  - DELETE: видалення даних (наприклад, видалити документ).
- Використання ресурсів (Resources): кожен ресурс (наприклад, студент, курс, документ) має унікальний URL, що дозволяє легко отримати до нього доступ. Наприклад:
  - /students/123: доступ до інформації про студента з ідентифікатором 123.
- Клієнт-серверна архітектура (Client-Server Architecture): клієнтська частина (фронтенд) і серверна частина (бекенд) чітко розділені. Клієнт лише відправляє запити, а сервер обробляє їх та повертає відповіді.
- Кешування: сервер може вказати, що певні відповіді можна кешувати, що зменшує навантаження на сервер та прискорює роботу системи.

Контролери є важливим компонентом серверної архітектури, оскільки саме вони приймають HTTP-запити і відповідають за їх первинну обробку. Кожен

контролер відповідає за певний ресурс або набір дій у системі. Наприклад, можна створити контролери для управління студентами, курсами, документами та іншими ресурсами.

Сервіси відповідають за виконання бізнес-логіки, тобто тієї частини системи, яка обробляє дані перед їхньою передачею або після отримання. Вони виконують операції з даними, обробляють валідації, роблять розрахунки або забезпечують правила доступу. Наприклад, сервіс для управління студентами може мати наступні методи:

- GetAllStudentsAsync(): отримує список всіх студентів.
- GetStudentByIdAsync(int id): отримує дані про студента за його ID.
- AddStudentAsync(Student student): додає нового студента.
- UpdateStudentAsync(Student student): оновлює інформацію про студента.
- DeleteStudentAsync(int id): видаляє студента з системи.

Ці методи можуть включати логіку перевірки наявності студента в системі, валідації даних перед додаванням, а також інші операції, необхідні для коректної роботи системи.

Репозиторії — це компонент серверної частини, який взаємодіє безпосередньо з базою даних. Основне завдання репозиторію — абстрагувати деталі роботи з базою даних від решти системи. Це дозволяє змінювати або оптимізувати взаємодію з базою даних без необхідності змін у бізнес-логіці.
Репозиторій може містити такі методи, як:

- GetAllAsync(): отримує всі записи з бази даних.
- GetByIdAsync(int id): отримує запис за унікальним ідентифікатором.
- AddAsync(T entity): додає новий запис.
- UpdateAsync(T entity): оновлює існуючий запис.
- DeleteAsync(int id): видаляє запис за ID.

Безпека є одним із ключових аспектів у проектуванні серверної частини. Для захисту даних та запобігання несанкціонованого доступу використовуються різні методи:

- JWT (JSON Web Token): використовується для аутентифікації користувачів. Кожен запит до API містить токен, який підтверджує, що користувач авторизований і має право доступу до певних ресурсів.
- HTTPS: забезпечує шифрування всіх даних, що передаються між клієнтом і сервером.
- Авторизація на основі ролей: система повинна підтримувати різні рівні доступу для різних типів користувачів (наприклад, студенти, викладачі, адміністрація).

Логування допомагає відслідковувати роботу системи та виявляти можливі проблеми на сервері. Для цього можна використовувати вбудовані механізми логування в ASP.NET Core або сторонні бібліотеки, такі як Serilog.

Обробка помилок — це ще один важливий аспект серверної частини. Для цього використовуються глобальні механізми перехоплення винятків, які забезпечують обробку критичних помилок і відправлення коректних відповідей клієнту.

Проектування серверної частини та API включає в себе розробку контролерів для обробки запитів, реалізацію бізнес-логіки в сервісах і абстрагування роботи з базою даних за допомогою репозиторіїв. Використання REST API забезпечує зручну і зрозумілу взаємодію між клієнтом і сервером, а забезпечення безпеки та ефективна обробка помилок роблять систему надійною та стійкою до можливих загроз.

2.6 Проектування безпеки та управління доступом

Безпека інформаційних систем є критично важливим аспектом проектування, особливо коли система обробляє чутливі дані, такі як особиста інформація студентів, викладачів і адміністрації. Розробка безпечної інформаційної системи передбачає кілька рівнів захисту: від аутентифікації користувачів до шифрування даних і налаштування прав доступу. Розглянемо теоретичні основи безпеки в інформаційних системах, механізми аутентифікації

та авторизації, а також методи забезпечення цілісності і конфіденційності даних. Безпека інформаційних систем охоплює набір заходів, спрямованих на захист інформації від загроз, таких як несанкціонований доступ, зміна даних або їх втрата.

Основні принципи забезпечення безпеки:

- Конфіденційність (Confidentiality): забезпечення того, щоб доступ до даних отримували лише авторизовані користувачі.
- Цілісність (Integrity): гарантія того, що дані не будуть змінені або зіпсовані в процесі зберігання або передачі.
- Доступність (Availability): забезпечення доступу до даних для авторизованих користувачів тоді, коли це необхідно.
- Ідентифікація та автентифікація (Authentication): перевірка особи користувача перед тим, як надати йому доступ до системи.
- Авторизація (Authorization): визначення прав і ролей користувачів для доступу до певних ресурсів системи.

Аутентифікація — це процес перевірки особи користувача, який намагається увійти в систему[47]. В інформаційній системі супроводу освітнього процесу важливо розмежовувати доступ між різними типами користувачів, такими як студенти, викладачі та адміністрація. Є наступні типи аутентифікації:

- Базова аутентифікація (Basic Authentication): найпростіший метод, який передбачає відправку логіна та пароля з кожним запитом. Однак цей метод вважається недостатньо безпечним через можливість перехоплення даних.
- JWT (JSON Web Token): сучасний підхід до аутентифікації, який широко використовується в веб-додатках. JWT-токен створюється після успішної аутентифікації і передається в заголовку кожного наступного запиту. Токен містить зашифровану інформацію про користувача і його права доступу, тому немає необхідності постійно звертатися до бази даних для перевірки авторизації. Використання JWT має такі переваги:

- Безстанова природа: сервер не зберігає сесії користувачів, що підвищує масштабованість системи.
- Простота інтеграції з різними системами: JWT-токени можна використовувати як у веб-додатках, так і в мобільних.

Сам механізм роботи JWT представляє собою:

- Аутентифікація: користувач вводить свій логін і пароль. Якщо ці дані є коректними, сервер генерує JWT-токен.
- Передача токену: токен передається клієнту (фронтенд) і зберігається в локальному сховищі браузера або у файлі cookie.
- Авторизація запитів: при кожному наступному запиті клієнт додає токен у заголовок запиту, який сервер перевіряє перед обробкою.
- Валідація: сервер розшифровує токен і перевіряє його дійсність (термін дії, коректність підпису). Якщо токен валідний, користувач отримує доступ до ресурсів.

Авторизація — це процес перевірки прав користувача на виконання певних дій або доступ до певних ресурсів[48]. Після аутентифікації система визначає, до яких ресурсів має доступ користувач, виходячи з його ролі або привілеїв.

Модель контролю доступу на основі ролей (RBAC) дозволяє керувати доступом на основі ролей користувачів у системі[49]. Для інформаційної системи підтримуються такі ролі:

- Студент: має доступ до перегляду особистих даних, розкладу, оцінок, навчальних матеріалів, а також до певних документів.
- Викладач: має доступ до управління курсами, завантаження навчальних матеріалів, введення оцінок та перегляду звітів.
- Адміністрація: має доступ до управління всіма модулями системи, включаючи студентів, викладачів, курси, документи і звіти.

Переваги RBAC:

- Гнучкість: можна легко додавати нові ролі або змінювати права доступу для існуючих ролей.

- Масштабованість: система підтримує велике число користувачів без необхідності індивідуальної настройки прав для кожного користувача.
- Простота управління: адміністратори можуть легко контролювати права доступу користувачів на основі їхніх ролей.

Захист даних є критично важливим для інформаційної системи, яка обробляє чутливу інформацію. Основні механізми захисту даних включають:

Шифрування забезпечує захист даних як під час їх передачі, так і під час зберігання:

- Шифрування під час передачі: всі дані, що передаються між клієнтом і сервером, повинні бути зашифровані через HTTPS. Це гарантує, що навіть якщо дані будуть перехоплені, вони не зможуть бути прочитані.
- Шифрування даних на сервері: конфіденційна інформація, така як паролі користувачів або особиста інформація, повинна зберігатися в зашифрованому вигляді в базі даних.

Захист паролів: Паролі користувачів не повинні зберігатися в незашифрованому вигляді. Для захисту паролів використовується:

- Хешування: паролі зберігаються у вигляді хешу. Навіть якщо база даних буде зламана, хеші паролів не можна буде легко відновити до оригінальних значень.
- Сіль (Salt): для кожного пароля додається випадкова "сіль" перед хешуванням, що запобігає використанню заздалегідь обчислених хешів (rainbow tables).

Захист від атак: Щоб захистити систему від поширених типів атак, таких як SQL-ін’єкції та XSS (Cross-Site Scripting), потрібно впровадити такі механізми:

- Захист від SQL-ін’єкцій: SQL-ін’єкції виникають, коли користувач може вставити шкідливий SQL-код у запити до бази даних[50]. Для

захисту використовуються підготовлені запити (Parameterized Queries), які не дозволяють виконувати шкідливі SQL-вставки.

- Захист від XSS (Cross-Site Scripting): XSS-атаки дозволяють зловмисникам виконувати шкідливий JavaScript-код у браузері користувача[51]. Для захисту використовується екранування виводу (sanitization) всіх даних, що вводяться користувачем і відображаються в інтерфейсі, щоб запобігти виконанню небажаного коду.

Управління сесіями та таймаут: Щоб запобігти несанкціонованому доступу в результаті зламу сесій або вкрадених токенів, слід впровадити механізм таймаутів сесій. Токени доступу повинні мати термін дії, після якого користувач має пройти повторну аутентифікацію.

Логування та моніторинг: Логування є важливим аспектом безпеки, оскільки дозволяє відслідковувати всі дії, які відбуваються в системі. Це включає:

- Аутентифікаційні події: спроби входу в систему, успішні або неуспішні.
- Дії адміністратора: зміни в правах доступу або видалення даних.
- Інші критичні події: будь-які помилки або незвичайна поведінка системи.

Моніторинг цих подій допомагає швидко виявляти спроби несанкціонованого доступу і реагувати на можливі загрози.

Безпека і управління доступом є важливими елементами розробки інформаційної системи. Впровадження таких методів, як аутентифікація за допомогою JWT, авторизація на основі ролей, шифрування даних та захист від поширених атак, допомагає забезпечити захищеність системи. Крім того, механізми моніторингу та логування дозволяють своєчасно реагувати на загрози та підтримувати високу надійність системи.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ СУПРОВОДУ ОСВІТНЬОГО ПРОЦЕСУ ЗАКЛАДІВ ВИЩОЇ ОСВІТИ

3.1 Опис процесу оперування даними

У створеній інформаційній системі супроводу освітнього процесу закладів вищої освіти використовується багаторівнева архітектура клієнт-серверного додатку. На кожному рівні застосовуються різні моделі для оперування даними. Серверна частина реалізована за допомогою 3-рівневої архітектури, що дозволяє чітко розмежувати рівні клієнта, бізнес-логіки та доступу до даних. Програмний модуль використовує наступні класи для оперування даними:

Рівень клієнта: Моделі, що використовуються на фронтенді для передачі та відображення даних:

- ApplicantPageContentModel
- LoginModelModel
- SignupModelModel
- UserModelModel
- ContactsPageContentModel
- CooperationPageContentModel
- DepartmentMemberModel
- DepartmentMembersPageContentModel
- TeacherCardViewModel
- EducationalProgramsPageContentModel
- FileModelModel
- DepartmentHistoryCardModel
- DepartmentHistoryInfoModel
- HomePageMainFunctionsViewModel
- NewsCardViewModel
- ResearchActivitiesPageContentModel

- TemplateFileAccessUpdateModel
- TemplateFileResponseModel
- TemplateFileUploadModel
- TemplateFileModel

Рівень бізнес-логіки: DTO (Data Transfer Objects) для передачі даних між рівнями та реалізації логіки:

- ApplicantPageContentDto.cs
- AssignFileToUsersDto.cs
- ChangeRoleDto.cs
- ContactsPageContentDto.cs
- CooperationPageContentDto.cs
- DepartmentHistoryCardContentDto.cs
- DepartmentHistoryInfoDto.cs
- DepartmentMemberDto.cs
- DepartmentMembersInfoDto.cs
- EducationalProgramsPageContentDto.cs
- FileDto.cs
- LoginDto.cs
- ResearchActivitiesPageContentDto.cs
- SignUpDto.cs
- TemplateFileDto.cs
- UserDto.cs

Рівень доступу до даних (DAL): Сутності для збереження та взаємодії з базою даних:

- TemplateFileEntity
- UserEntity
- ApplicantPageContentEntity
- ContactsPageContentEntity
- CooperationPageContentEntity
- DepartmentHistoryCardContentEntity

- DepartmentHistoryInfoEntity
- DepartmentMemberEntity
- DepartmentMembersInfoEntity
- EducationalProgramsPageContentEntity
- ResearchActivitiesPageContentEntity
- UserFileEntity

Розроблена інформаційна система супроводу освітнього процесу закладів вищої освіти базується на чітко визначеному процесі обробки даних між різними рівнями клієнт-серверної архітектури. Обробка даних відбувається шляхом їх передачі від клієнтської частини до серверної, де бізнес-логіка забезпечує відповідну обробку, а рівень доступу до даних працює безпосередньо з базою даних. Після цього результат повертається до клієнта у вигляді підготовлених даних. На клієнтському рівні дані вводяться або вибираються користувачем через веб-інтерфейс. Ці дані формуються у вигляді моделей, наприклад, TemplateFileModel або RoleChangeModel, та надсилаються на сервер у форматі HTTP-запиту. На серверному рівні дані обробляються у декілька етапів. Спочатку отримані з клієнта моделі мапляться у DTO (Data Transfer Objects), такі як TemplateFileDto або ChangeRoleDto. DTO адаптуються для подальшої обробки бізнес-логікою, яка перевіряє валідність отриманих даних. Наприклад, перевіряється існування користувача у базі даних або правильність формату файлу. Після успішної перевірки бізнес-логіка формує запит до рівня доступу до даних (DAL), який безпосередньо взаємодіє з базою даних. Рівень DAL працює із сутностями, такими як TemplateFileEntity або UserEntity, для виконання CRUD-операцій (створення, читання, оновлення, видалення). Наприклад, при збереженні шаблону документа сутність TemplateFileEntity зберігає назву файлу, його вміст у форматі Base64 та роль доступу. Результати операцій, виконаних у базі даних, передаються назад на рівень бізнес-логіки, де вони формуються у відповідні DTO, і врешті повертаються клієнту у вигляді моделей для відображення у веб-інтерфейсі. Таким чином, дані проходять через усі три рівні архітектури: клієнтський інтерфейс, бізнес-логіку та рівень доступу до даних, що

забезпечує масштабованість, гнучкість та безпеку системи. Процес обробки даних враховує особливості кожної сутності:

1. TemplateFileEntity — використовується для збереження шаблонів документів із зазначенням ролей доступу. Процеси включають додавання нового шаблону через інтерфейс, редагування ролі доступу до шаблону та видалення шаблонів. Сутність підтримує повний цикл CRUD-операцій.
2. UserEntity — представляє користувачів системи, таких як студенти, викладачі та адміністратори. Обробка даних включає реєстрацію нових користувачів через модель SignUpDto, аутентифікацію через LoginDto та зміну ролей через ChangeRoleDto.
3. ApplicantPageContentEntity — зберігає контент сторінки вступника, зокрема загальну інформацію, інформацію про перспективи працевлаштування та студентські роботи. Редагування контенту здійснюється через DTO ApplicantPageContentDto та збереження у базі.
4. ContactsPageContentEntity — використовується для збереження контактної інформації. Контент оновлюється через клієнтський інтерфейс за допомогою DTO ContactsPageContentDto.
5. CooperationPageContentEntity — містить інформацію про напрями співпраці кафедри з іншими установами. Клієнтська частина надсилає запити для редагування цього контенту через DTO CooperationPageContentDto, які обробляються бізнес-логікою.
6. DepartmentHistoryCardContentEntity — забезпечує короткий контент історії кафедри у вигляді карток. Процес обробки включає додавання нових карток та оновлення існуючих через CRUD-операції.
7. DepartmentHistoryInfoEntity — зберігає повну інформацію про історію кафедри. Обробка даних включає редагування цього контенту через відповідний DTO DepartmentHistoryInfoDto.
8. DepartmentMemberEntity — представляє дані про членів кафедри, таких як викладачі та адміністративний персонал. Через DTO

DepartmentMemberDto можна додавати нових членів кафедри або редагувати інформацію про існуючих.

9. DepartmentMembersInfoEntity — забезпечує загальну інформацію про склад кафедри. Оновлення даних відбувається через DTO DepartmentMembersInfoDto.
10. EducationalProgramsPageContentEntity — містить інформацію про освітні програми кафедри. Редагування цього контенту здійснюється через клієнтський інтерфейс із використанням DTO EducationalProgramsPageContentDto.
11. ResearchActivitiesPageContentEntity — використовується для управління контентом сторінки наукових досліджень, включаючи інформацію про наукові інтереси, публікації та заходи. Редагування контенту відбувається через DTO ResearchActivitiesPageContentDto.
12. UserFileEntity — прив’язує завантажені файли до конкретних користувачів. При завантаженні файлу користувачем він прив’язується до його облікового запису через цю сутність.
13. FileEntity — представляє окремі файли, збережені у базі даних, з вказівкою на їх вміст (у форматі Base64), назву та ролі доступу. Операції з файлами включають завантаження, видалення та оновлення інформації.
14. LoginDto, AuthenticationRequestDto, AuthenticationResponseDto — забезпечують взаємодію між клієнтським інтерфейсом та бізнес-логікою для авторизації користувачів. Дані про логін, пароль та роль передаються через ці DTO.

Кожна з цих сутностей і моделей забезпечує злагоджену роботу системи, де дані проходять через чітко структуровані етапи обробки. Це дозволяє не тільки ефективно працювати з інформацією, але й забезпечує високу надійність та гнучкість при розширенні функціоналу системи. Система забезпечує масштабованість, підтримує чітке розмежування відповідальності між рівнями і гарантує легкість підтримки та розвитку.

3.2 Основні авторські методи класів, використані в реалізації

Нижче наведено основні авторські методи, реалізовані в рамках розробки інформаційної системи супроводу освітнього процесу закладів вищої освіти.

- UploadFileAsync(TemplateFileDto fileDto) – Додає новий файл у базу даних. Кодує вміст файлу у формат Base64 та прив’язує до нього роль доступу.
- GetFilesByRoleAsync(string role) – Отримує всі файли, доступні для вказаної ролі.
- GetAllFilesAsync() – Повертає всі доступні файли для перегляду адміністраторами.
- UpdateFileAccessRoleAsync(TemplateFileDto fileDto) – Оновлює роль доступу для вибраного файлу.
- DeleteFileAsync(int id) – Видаляє файл за унікальним ідентифікатором.
- SignUpAsync(SignUpDto signUpDto) – Реєструє нового користувача. Виконує хешування пароля перед збереженням у базу даних.
- LoginAsync(LoginDto loginDto) – Повертає токен авторизації після успішної перевірки логіну та пароля.
- ChangeUserRoleAsync(ChangeRoleDto changeRoleDto) – Дозволяє змінити роль користувача.
- GetAllUsersAsync() – Повертає список усіх зареєстрованих користувачів.
- GetUserByEmailAsync(string email) – Отримує користувача за email.
- UpdateApplicantPageContentAsync(ApplicantPageContentDto contentDto) – Оновлює розділи сторінки, такі як інформація про вступ або студентські роботи.
- GetApplicantPageContentAsync() – Повертає актуальний контент сторінки для перегляду.
- UpdateContactsPageContentAsync(ContactsPageContentDto contentDto) – Оновлює контактні дані (телефон, email, адреса).
- GetContactsPageContentAsync() – Повертає актуальні контактні дані.

- UpdateCooperationPageContentAsync(CooperationPageContentDto contentDto) – Оновлює інформацію про напрями співпраці.
- GetCooperationPageContentAsync() – Повертає актуальний контент
- AddDepartmentHistoryCardAsync(DepartmentHistoryCardContentDto cardDto) – Додає нову картку в історію.
- UpdateDepartmentHistoryCardAsync(DepartmentHistoryCardContentDto cardDto) – Оновлює існуючу картку.
- GetAllDepartmentHistoryCardsAsync() – Повертає всі картки історії.
- UpdateDepartmentHistoryInfoAsync(DepartmentHistoryInfoDto infoDto) – Оновлює інформацію про історію кафедри.
- GetDepartmentHistoryInfoAsync() – Повертає актуальний текстовий контент історії.
- AddDepartmentMemberAsync(DepartmentMemberDto memberDto) – Додає нового члена кафедри до бази.
- UpdateDepartmentMemberAsync(DepartmentMemberDto memberDto) – Оновлює інформацію про існуючого члена.
- DeleteDepartmentMemberAsync(int id) – Видаляє члена кафедри за його ID.
- GetAllDepartmentMembersAsync() – Повертає список усіх членів кафедри.
- UpdateDepartmentMembersInfoAsync(DepartmentMembersInfoDto infoDto) – Оновлює загальну інформацію про склад кафедри.
- GetDepartmentMembersInfoAsync() – Повертає актуальні дані.
- UpdateEducationalProgramsPageContentAsync(EducationalProgramsPage ContentDto contentDto) – Оновлює інформацію про освітні програми.
- GetEducationalProgramsPageContentAsync() – Повертає актуальний контент.
- UpdateResearchActivitiesPageContentAsync(ResearchActivitiesPageConte ntDto contentDto) – Оновлює розділи наукової діяльності (публікації, конференції, інтереси).

- GetResearchActivitiesPageContentAsync() – Повертає актуальний контент сторінки.
- UploadUserFileAsync(UserFileDto fileDto) – Завантажує новий файл для конкретного користувача.
- GetUserFilesAsync(int userId) – Отримує всі файли, пов’язані з конкретним користувачем.
- DeleteUserFileAsync(int fileId) – Видаляє файл користувача за ID.

Кожна сутність має набір методів, що реалізують її функціонал відповідно до бізнес-логіки системи. Це забезпечує структурованість, масштабованість та легкість у розширенні функціоналу системи.

3.3 Інтерфейс користувача та посібник експлуатації

Весь інтерфейс користувача написаний українською мовою. Інформаційна система має тринадцять сторінок та допоміжні елементи для взаємодії з користувачем. Сторінка історії кафедри(див. додаток А) виконує функцію надання інформації про історію кафедри як у скороченому так і у повному її вигляді. Скорочений вигляд історії кафедри представлений за допомогою інформаційних карток(див. рис. 3.1.).

Рисунок 3.1. Приклад інформаційної картки

Подібні інформаційні картки містять певний заголовок та опис із короткою інформацією. Способом додавання кількох описаних елементів була створена частина сторінки історії кафедри із скороченим вмістом історії кафедри(див. рис. 3.2.).

Рисунок 3.2. Скорочений вміст історії кафедри

Також на сторінці присутній повний вміст історії кафедри, який представлений текстовим елементом(див. рис. 3.3.).

Рисунок 3.3. Повний вміст історії кафедри

Вміст кожного з цих елементів користувач із роллю «Адміністратор» може редагувати, для цього потрібно просто натиснути на елемент і відкриється панель редагування(див. рис. 3.4).

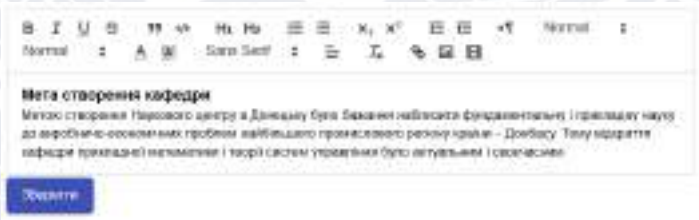


Рисунок 3.4. Приклад панелі редагування текстового елементу

Панель містить розширені функції текстового редактору, які інтуїтивно зрозумілі користувачу через піктограми. Також міститься кнопка «Зберегти» при натисканні якої зміни надсилаються на сервер та зберігаються у базі даних.

Сторінка колективу кафедри(див. додаток Б) виконує функцію надання інформації про колектив кафедри та її кадровий потенціал. Вона складається з переліку членів кафедри та поля з інформаційною виноскою.

Перелік колективу кафедри складається з інформаційних карток(див. рис. 3.5.), які містять інформацію про члена кафедри.

Рисунок 3.5. Приклад інформаційної картки із членом кафедри

Інформаційну виноску(див. рис. 3.6) користувач із роллю «Адміністратор» може редагувати, для цього потрібно просто натиснути на елемент.

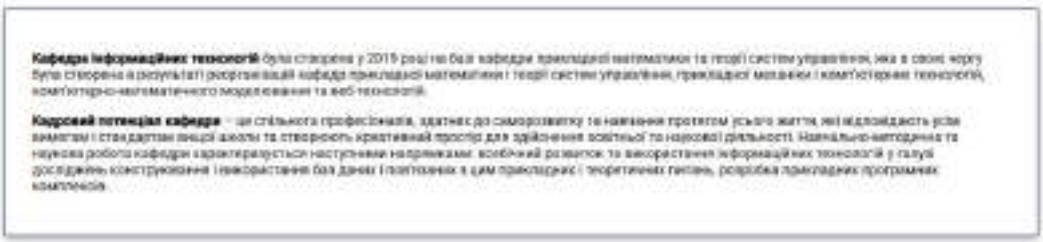


Рисунок 3.6. Виноска із описом кадрового потенціалу кафедри

При натисканні на інформаційну картку відкриється сторінка із описом члена кафедри(див. рис. 3.7.).

Рисунок 3.7. Сторінка із інформацією про члена кафедри

Умовно сторінка поділена на 3 структурні елементи, які відкриті для редагування користувачам із роллю «Адміністратор»: додатковий блок інформації (див. рис. 3.8), фотографія(див. рис. 3.9.), основний блок інформації(див. рис. 3.10.). При натисканні на додатковий блок інформації також буде відриватись панель редагування(див. рис. 3.4.).

Рисунок 3.8. Додатковий блок інформації

Рисунок 3.9. Фото із наведеним курсором

При натисканні на основний блок інформації буде відриватись панель редагування(див. рис. 3.4.).

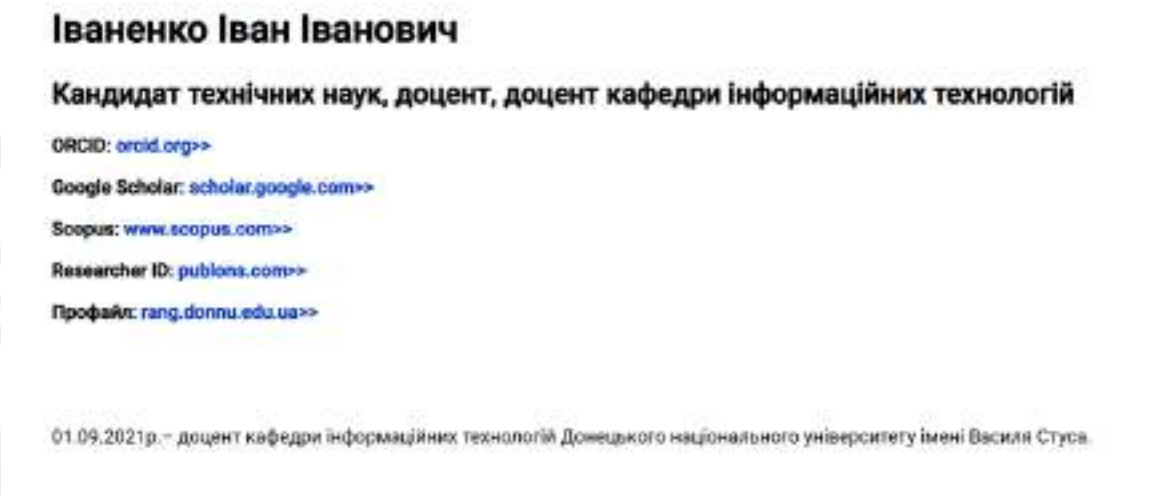


Рисунок 3.10. Основний блок інформації

Сторінка із інформацією для абітурієнтів(див. додаток В) поділена на 3 структурні елементи, які відкриті для редагування користувачам із роллю «Адміністратор»: основний блок, блок із місцями майбутнього працевлаштування, блок із студентськими роботами.

Сторінка із інформацією про освітні програми(див. додаток Г) поділена на 6 структурних елементів, які відкриті для редагування користувачам із роллю «Адміністратор»: основний блок, блок із корисними посиланнями, слайдер(див. рис. 3.11.), блок із інформацією про бакалаврат, блок із інформацією про магістратуру, блок із інформацією про докторат філософії.

Рисунок 3.11. Слайдер із інформацією доступною до редагування

Сторінка із інформацією про наукову діяльність(див. додаток Д) поділена на 4 структурні елементи, які відкриті для редагування користувачам із роллю «Адміністратор»: блок із науковою діяльністю кафедри, блок із науковими інтересами, блок із науковими журналами, блок із науковими подіями.

Сторінка із інформацією про співробітництво(див. додаток Е) містить 1 структурний елемент, який відкритий для редагування користувачам із роллю «Адміністратор».

Сторінка із інформацією про контакти кафедри(див. рис. 3.12.) містить 1 структурний елемент, який відкритий для редагування користувачам із роллю «Адміністратор».

Рисунок 3.12 Сторінка із контактами кафедри

Для того щоб користуватись розширеними функціями системи, потрібно зареєструватись у ній та увійти. Щоб виконати дію реєстрації потрібно натиснути на піктограму користувача, яка розташована на шапці сторінки та обрати опцію «Зареєструватись»(див. рис. 3.13.).

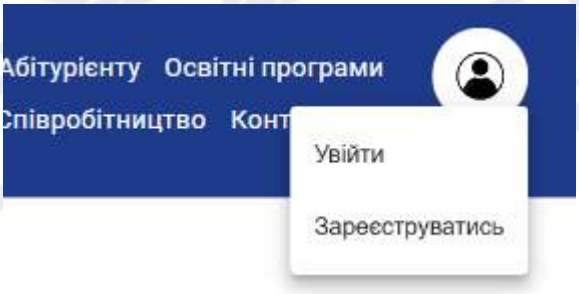


Рисунок 3.13. Піктограма неавторизованого користувача із опціями

Після натискання на опцію відкриється сторінка реєстрації користувача(див. рис. 3.14.). На ній міститься форма, яку потрібно заповнити для створення акаунту.

Рисунок 3.14 Сторінка реєстрації користувача

Якщо форма буде заповнена некоректними даними(невалідний формат email-адреси, невалідна довжина паролю, неспівпадіння паролю із підтвердженням), то користувачу буде показано повідомлення про це(див. рис. 3.15.).

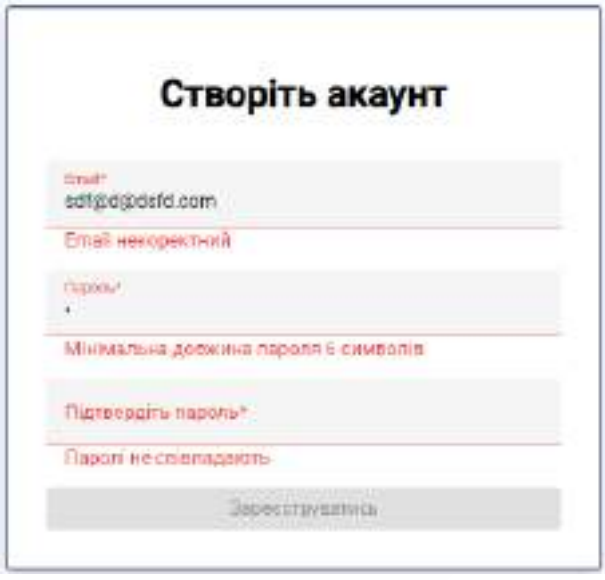


Рисунок 3.15. Форма із некоректними даними

Після створення акаунту потрібно увійти в систему, щоб виконати вхід потрібно натиснути на піктограму користувача, яка розташована на шапці сторінки та обрати опцію «Увійти»(див. рис. 3.13.). Якщо користувач вводить коректні дані, то буде показано сповіщення про це(див. рис. 3.16.). Схоже сповіщення буде також з’являтись, якщо користувач вводить некоректні дані, але у сповіщенні буде інше повідомлення.

Рисунок 3.16. Сповіщення про успішний вхід

Після успішного входу, зробивши клік на піктограму користувача, з’явиться меню із опціями для авторизованих користувачів(див. рис. 3.17.)

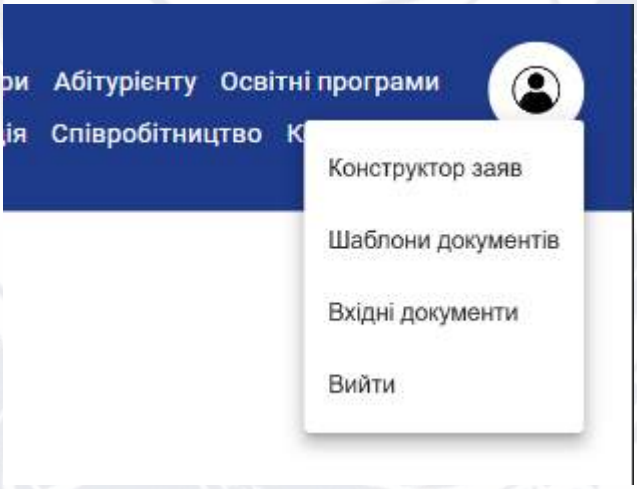


Рисунок 3.17. Піктограма авторизованого користувача із роллю «Адміністратор» з опціями

При натисканні, на опцію «Шаблони документів», відкриється сторінка із списком шаблонів, які завантажив адміністратор(див. рис. 3.18.).

Рисунок 3.18. Сторінка із списком шаблонів документів

Для кожного шаблону адміністратор може призначати роль користувача, якому буде доступний шаблон для роботи, а також видаляти шаблон. Для того щоб виконати одну з зазначених дій, потрібно натиснути на піктограму із зображенням вертикальних трьох крапок та обрати опцію(див. рис. 3.19).

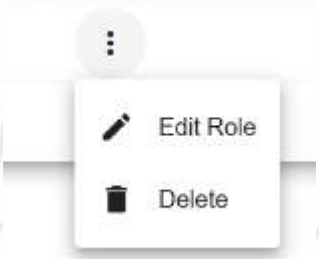


Рисунок 3.19. Піктограма із вибором дії для шаблону

При виборі дії зміни ролі доступу для шаблону, буде з’являтись вікно із вибором нової ролі(див. рис 3.20).

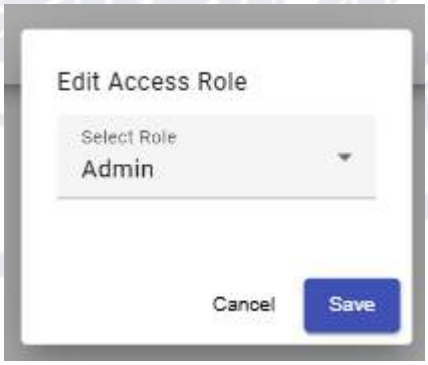


Рисунок 3.20. Вікно із вибором нової ролі для шаблону

При натисканні, на опцію «Конструктор заяв», відкриється сторінка із конструктором заяв, які завантажив адміністратор(див. рис. 3.21.).

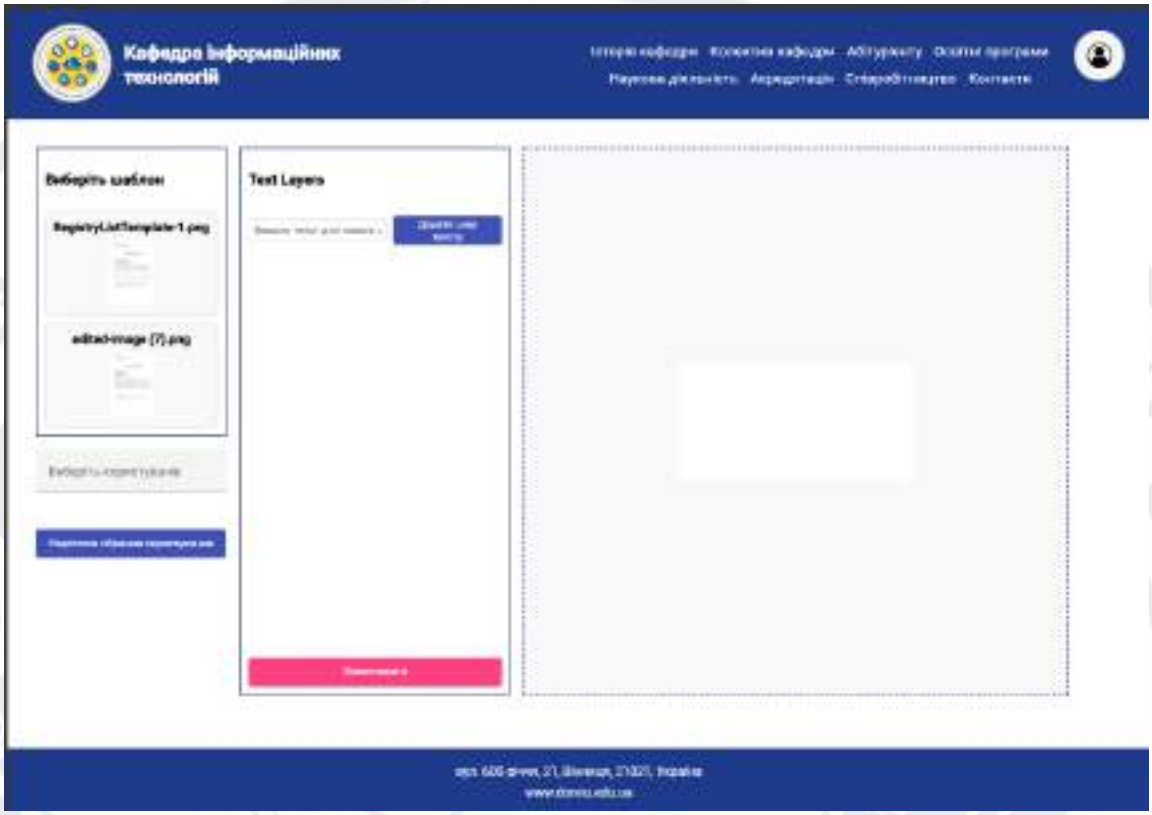


Рисунок 3.21. Конструктор заяв

Для кожної ролі у конструкторі заяв доступні лише ті шаблони, до яких у них є доступ. Для того що почати працювати із шаблоном, потрібно натиснути на нього, він з’явиться у робочій області. Далі потрібно створити шар тексту, який потрібно буде перемістити на шаблон у потрібне місце(див. рис. 3.22).

Рисунок 3.22 Конструктор заяв із створеними текстовими шарами

Текстовий шар можна видалити, а також змінити розмір шрифту. Сам стиль шрифту для текстового шару є Times New Roman. Коли потрібні маніпуляції із шаблоном виконані, його із змінами можна одразу надіслати іншим користувачам системи. Для цього потрібно обрати їх у полі вибору та натиснути кнопку «Надіслати обраним користувачам»(див. рис 3.23).

Рисунок 3.23. Поле вибору адресатів для готового шаблону

Після надсилання, користувачі можуть побачити файли перейшовши на сторінку «Вхідні документи». На цій сторінці також можна одразу працювати із

вхідними файлами за допомогою конструктору, а також надсилати іншим користувачам.

3.4 Перспективи розширення інформаційної системи

Інформаційна система є клієнт-серверним рішенням із 3-рівневою архітектурою. Ця архітектура дозволяє легко розширювати функціонал додатку. Збереження файлів користувачів у хмарному сховищі також сприяє цьому. Тому є можливість додати наступні покращення:

- Додаток може бути інтегрований з іншими системами, такими як система управління навчанням, електронний журнал або календар. Це дозволить автоматично синхронізувати дані між різними системами та забезпечити ще більшу автоматизацію процесу оформлення графіку.
- Додаток може бути розширений для підтримки різних платформ та пристроїв, таких як мобільні пристрої, планшети або інші. Це дозволить користувачам отримувати доступ до свого графіку та взаємодіяти з ним у зручний для них спосіб.
- Розширення може включати поліпшення інтерфейсу користувача, забезпечуючи зручну навігацію, візуалізацію даних та спрощення процесу взаємодії з додатком.
- Додаток може бути розширений для надання різних звітів та аналітичних даних щодо прогресу здобувачів, використання ресурсів або інших параметрів. Це дозволить здобувачам та викладачам отримувати цінну інформацію та приймати обґрунтовані рішення.

Ці перспективи розширення дозволять інформаційній системі стати ще більш корисним та ефективним інструментом для здобувачів та викладачів, сприяючи полегшенню та оптимізації процесів планування та моніторингу освітнього процесу.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи були вивчені сучасні підходи до обробки даних та розробки інформаційних систем, орієнтованих на потреби освітніх установ. Проведений аналіз дозволив визначити ключові принципи побудови ефективної клієнт-серверної архітектури, яка забезпечує зручність користувачів та відповідність сучасним вимогам до функціональності.

Розроблено архітектуру клієнт-серверної системи, що базується на технологіях .NET Core, Angular, PostgreSQL, і відповідає сучасним стандартам розробки інформаційних систем. Реалізовані програмні модулі дозволяють автоматизувати управління освітнім контентом, створення документів, моніторинг успішності студентів та забезпечують доступ до навчальних ресурсів.

Забезпечено захист інформації шляхом впровадження сучасних механізмів безпеки, включаючи авторизацію та аутентифікацію користувачів, шифрування даних, а також розмежування доступу відповідно до ролей.

Результати оцінки розробленої системи в реальних умовах функціонування закладу вищої освіти засвідчили її ефективність у зниженні адміністративного навантаження, покращенні організації навчального процесу та забезпеченні зручного доступу до інформації для всіх учасників освітнього процесу.

Таким чином, виконана робота підтверджує доцільність впровадження розробленої інформаційної системи для підвищення ефективності управління закладом вищої освіти та покращення якості освітніх послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Офіційна документація Moodle. URL: https://moodle.org/
2. Офіційна документація Blackboard Inc. URL: https://help.blackboard.com/
3. Офіційна документація Canvas. URL: https://www.instructure.com/canvas
4. Waterfall Model in SDLC. URL: https://www.simplilearn.com/waterfall-model-article
5. Sommerville, Ian. Software Engineering. URL: https://www.amazon.com/Software-Engineering-10th-Ian-Sommerville/dp/0133943038
6. Official Agile Manifesto. URL: https://agilemanifesto.org/
7. Swartout, Mike. Accelerate: The Science of Lean Software and DevOps. IT Revolution Press, 2018. 288 с.
8. Kendall, Kenneth E., and Kendall, Julie E. Systems Analysis and Design. Pearson, 2019. 552 с.
9. Pressman, Roger S. Software Engineering: A Practitioner's Approach. McGraw-Hill Education, 2015. 976 с.
10. Sutherland, Jeff. Scrum: The Art of Doing Twice the Work in Half the Time. Crown Business, 2014. 256 с.
11. Beck, Kent. Extreme Programming Explained: Embrace Change (2nd Edition). Addison-Wesley, 2015. 224 с.
12. Stair, Ralph M., and Reynolds, George W. Fundamentals of Information Systems. Cengage Learning, 2018. 464 с.
13. Watson, William R., and Watson, Sunnie Lee. Learning Management System Technologies and Software Solutions for Online Teaching: Tools and Applications. IGI Global, 2016. 412 с.
14. Офіційна документація .NET Web API. URL: https://learn.microsoft.com/en-us/aspnet/core/web-api/
15. Офіційна документація PostgreSQL. URL: https://www.postgresql.org/docs/
16. Angular. Angular Documentation. URL: https://angular.io/docs
17. Masse, Mark. REST API Design Rulebook. O'Reilly Media, 2015. 128 с.

18. Gourley, David, and Totty, Brian. HTTP: The Definitive Guide. O'Reilly Media, 2018. 656 с.
19. Long, Rick. Multitier Enterprise Applications for Microsoft .NET. Microsoft Press, 2019. 450 с.
20. Microsoft Corporation. ASP.NET Core Documentation. URL: https://learn.microsoft.com/en-us/aspnet/core/
21. JWT.IO. JSON Web Tokens Introduction. URL: https://jwt.io/introduction/
22. MacLane, Lindsay. JSON at Work: Practical Data Integration for the Web. O'Reilly Media, 2017. 300 с.
23. Microsoft Corporation. Entity Framework Core Documentation. URL: https://learn.microsoft.com/en-us/ef/core/
24. Stonebraker, Michael, and Hellerstein, Joseph M. Readings in Database Systems. MIT Press, 2015. 612 с.
25. Codd, Edgar F. The Relational Model for Database Management: Version 2. Addison-Wesley, 2014. 568 с.
26. Kroenke, David M., and Auer, David J. Database Concepts. Pearson, 2018. 600 с.
27. Coronel, Carlos, Morris, Steven, and Rob, Peter. Database Systems: Design, Implementation, & Management. Cengage Learning, 2020. 768 с.
28. Kifer, Michael, Bernstein, Arthur, and Lewis, Philip M. Database Systems: An Application-Oriented Approach. Addison-Wesley, 2018. 800 с.
29. Elmasri, Ramez, and Navathe, Shamkant B. Fundamentals of Database Systems. Pearson, 2020. 1272 с.
30. Özsu, M. Tamer, and Valduriez, Patrick. Principles of Distributed Database Systems. Springer, 2020. 674 с.
31. Gray, Jim, and Reuter, Andreas. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 2016. 792 с.
32. Haerder, Theo, and Reuter, Andreas. Principles of Transaction-Oriented Database Recovery. Springer, 2015. 300 с.

33. Garrett, Jesse James. The Elements of User Experience: User-Centered Design for the Web and Beyond. New Riders, 2015. 240 с.
34. Clark, Joe. Building Accessible Websites. New Riders, 2015. 432 с.
35. Frain, Ben. Responsive Web Design with HTML5 and CSS: Develop Future-Proof Responsive Websites Using the Latest HTML5 and CSS Techniques. Packt Publishing, 2020. 384 с.
36. Lidwell, William, Holden, Kritina, and Butler, Jill. Universal Principles of Design. Rockport Publishers, 2015. 272 с.
37. Cooper, Alan, Reimann, Robert, Cronin, David, and Noessel, Christopher. About Face: The Essentials of Interaction Design. Wiley, 2016. 720 с.
38. Johnson, Jeff. Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines. Morgan Kaufmann, 2020. 250 с.
39. Behrens, Roy R. Art, Design, and Gestalt Theory. Journal of Aesthetic Education, 2015. 45 с.
40. Frain, Ben. Responsive Web Design with HTML5 and CSS: Develop Future-Proof Responsive Websites Using the Latest HTML5 and CSS Techniques. Packt Publishing, 2020. 384 с.
41. Hogan, Greg. CSS Master: Advanced Web Standards Solutions. Apress, 2021. 350 с.
42. Coyier, Chris. Practical Flexbox: Flexible Box Layout Explained. CSS-Tricks, URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
43. Nielsen, Jakob. Usability Engineering. Morgan Kaufmann, 2014. 362 с.
44. Clifton, Brian. Advanced Web Metrics with Google Analytics. Wiley, 2015. 600 с.
45. Slatin, John M., and Rush, Sharron. Maximum Accessibility: Making Your Web Site More Usable for Everyone. Addison-Wesley, 2014. 432 с.
46. Figma. Official Figma Documentation. URL: https://help.figma.com/
47. Hunt, Troy. API Security in Action: A Guide to Modern Authentication. Manning Publications, 2020. 375 с.

48. DiFraia, Damien. Mastering OAuth 2.0: Implement Modern Authorization for Secure Apps. Packt Publishing, 2021. 350 с.
49. Ferraiolo, David F., Kuhn, Richard D., and Chandramouli, Ramaswamy. Role-Based Access Control, Second Edition. Artech House, 2016. 456 с.
50. Clarke, Justin. SQL Injection Attacks and Defense. Syngress, 2012. 576 с.
51. Heiderich, Mario. Web Application Obfuscation: -wrangling Web Security. Syngress, 2012. 248 с.

ДОДАТКИ

ДОДАТОК А

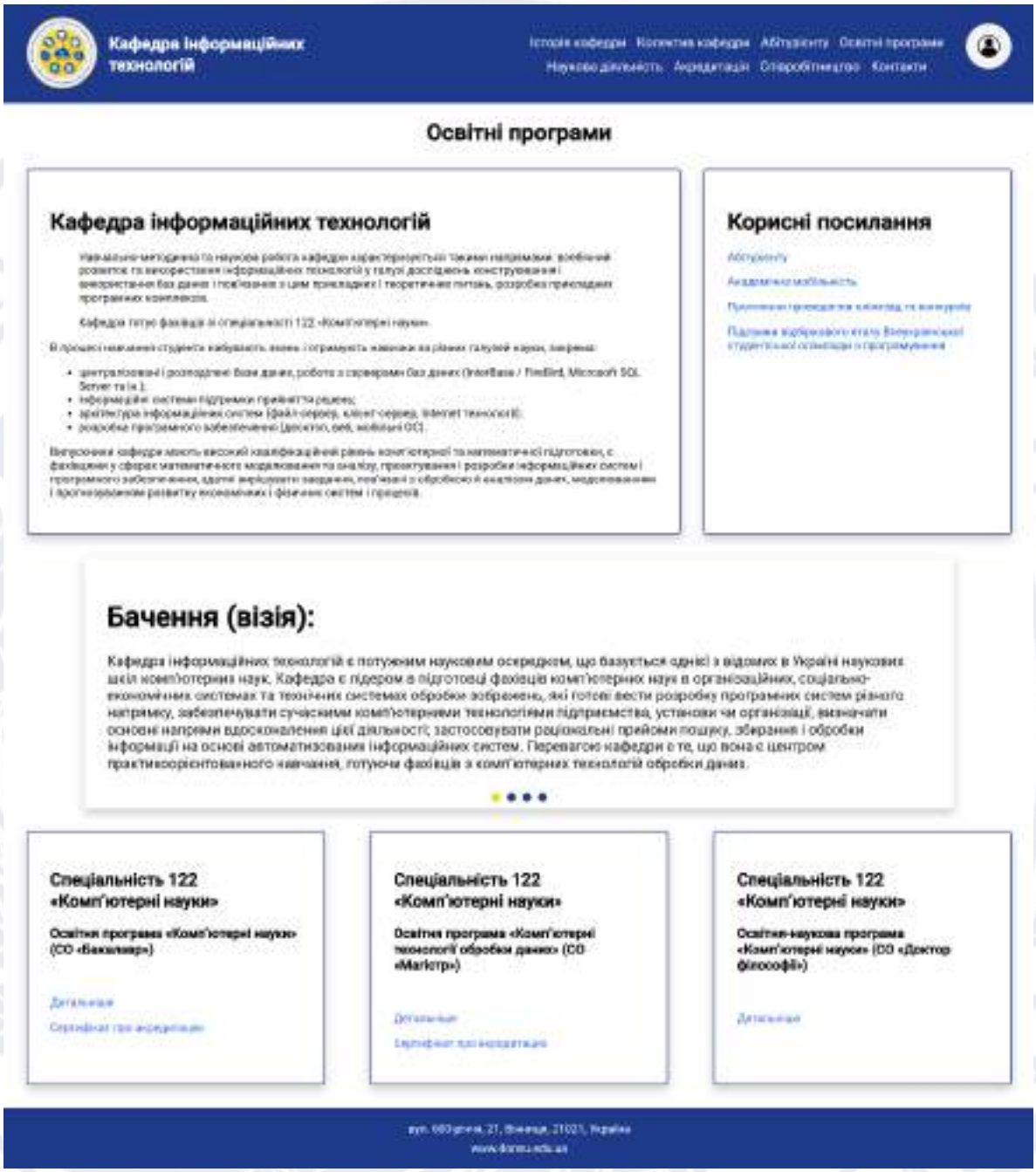

Історія кафедри
Кафедра інформаційних технологій
Кафедра інформаційних технологій

ДОДАТОК Б

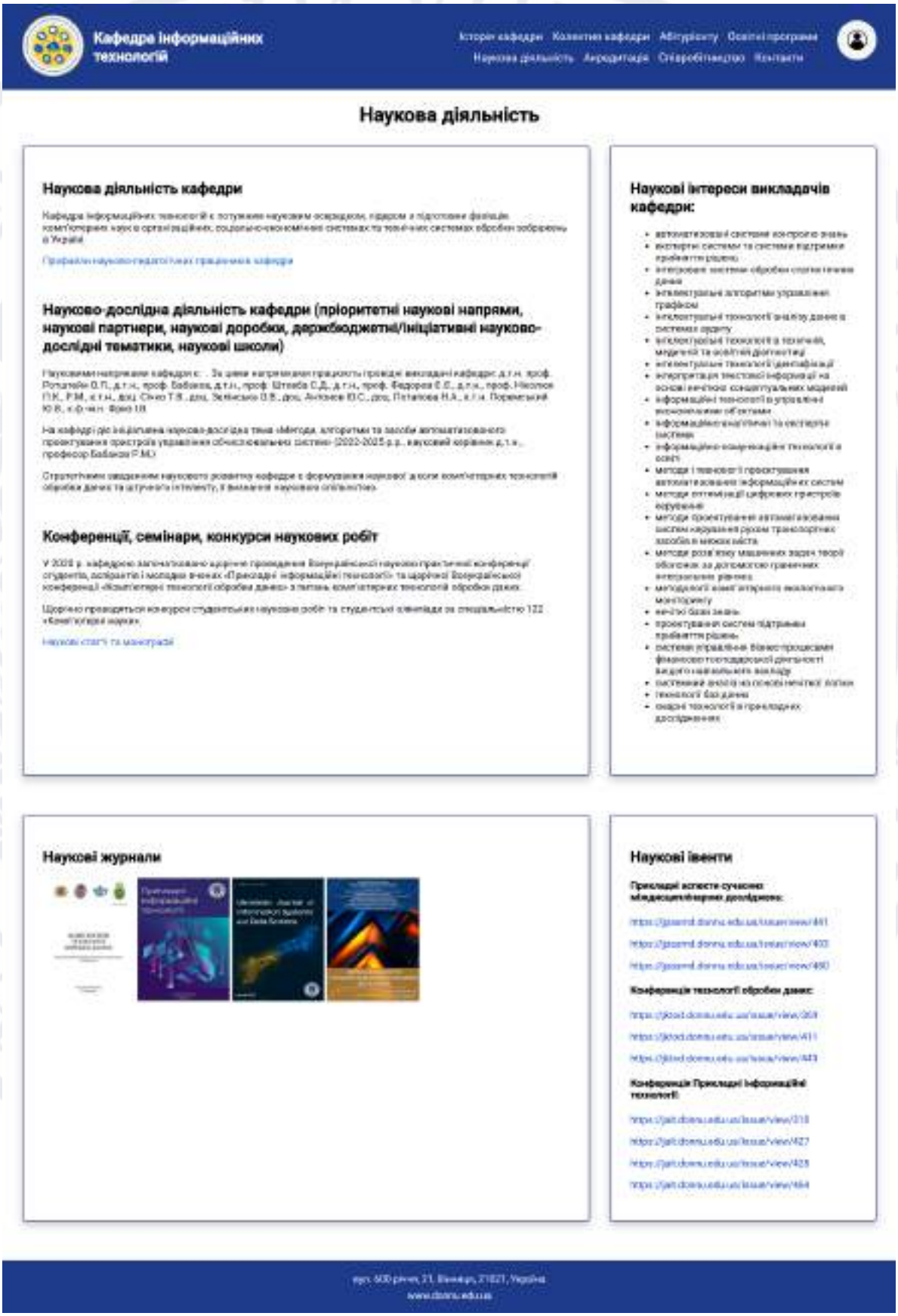
ДОДАТОК В


Кафедра інформаційних технологій
Абітурієнту
Шановний друже!
МІСЦЯ МАЙБУТНЬОГО ПРАЦЕВЛАШТУВАННЯ:

ДОДАТОК Г


Кафедра інформаційних технологій
Історія кафедри Колектив кафедри Абітурієнту Освітні програми
Наукова діяльність Акредитація Співробітництво Контакти
Освітні програми
Кафедра інформаційних технологій
Навчально-методична та наукова робота кафедри характеризується такими напрямами: особливий розвиток та використання інформаційних технологій у галузі досліджень конструювання і використання баз даних і пов'язаних з цим прикладних і теоретичних питань, розробка прикладних програмних комплексів.
Кафедра готує фахівців зі спеціальності 122 «Комп'ютерні науки».
В процесі навчання студенти набувають знань і отримують навички на рівні галузей наук, зокрема:
• централізовані і розподілені бази даних, робота з серверами баз даних (InterBase / FireBird, Microsoft SQL Server та ін.);
• інформаційні системи підтримки прийняття рішень;
• архітектура інформаційних систем (файл-сервер, клієнт-сервер, Internet технології);
• розробка програмного забезпечення (Delphi/10, Web, мобільні ОС).
Випускники кафедри мають високий кваліфікаційний рівень комп'ютерної та математичної підготовки, є фахівцями у сферах математичного моделювання та аналізу, проектування і розробки інформаційних систем і програмного забезпечення, здатні вирішувати завдання, пов'язані з обробкою й аналізом даних, моделюванням і прогнозуванням розвитку економічних і фізичних систем і процесів.
Корисні посилання
Абітурієнту
Академічна мобільність
Підготовка відділкового етапу Всеукраїнської студентської олімпіади з програмування
Бачення (візія):
Кафедра інформаційних технологій є потужним науковим осередком, що базується однієї з відомих в Україні наукових шкіл комп'ютерних наук. Кафедра є лідером в підготовці фахівців комп'ютерних наук в організаційних, соціально-економічних системах та технічних системах обробки зображень, які готові вести розробку програмних систем різного напрямку, забезпечувати сучасними комп'ютерними технологіями підприємства, установи чи організації, визначати основні напрями вдосконалення цієї діяльності; застосовувати раціональні прийоми пошуку, зберання і обробки інформації на основі автоматизованих інформаційних систем. Перевагою кафедри є те, що вона є центром практикоорієнтованого навчання, готуючи фахівців з комп'ютерних технологій обробки даних.
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки» (ОС «Бакалавр»)
Детальніше
Сертифікат про акредитацію
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні технології обробки даних» (ОС «Магістр»)
Детальніше
Сертифікат про акредитацію
Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Комп'ютерні науки» (ОС «Доктор філософії»)
Детальніше
вул. 600-річчя, 21, Вінниця, 21021, Україна
www.donnu.edu.ua

ДОДАТОК Д



Кафедра інформаційних технологій

Історія кафедри Колектив кафедри Абітурієнту Освітні програми
Наукова діяльність Акредитація Співробітництво Контакти

Наукова діяльність

Наукова діяльність кафедри

Кафедра інформаційних технологій є потужним науковим осередком, підвідом в підготовки фахівців комп'ютерних наук в організаційних, соціально-економічних системах та технічних системах обробки зображень в Україні.

Профайли науково-педагогічних працівників кафедри

Науково-дослідна діяльність кафедри (пріоритетні наукові напрями, наукові партнери, наукові доробки, держбюджетні/ініціативні науково-дослідні тематики, наукові школи)

Науковими напрямами кафедри є: ... За цими напрямами працюють викладачі кафедри: д.т.н. проф. Рогатинський О.П., д.т.н., проф. Бабаєва, д.т.н., проф. Штефа О.Д., д.т.н., проф. Федорова О.С., д.т.н., проф. Ніколов П.К., Р.М., к.т.н., доц. Сінчо Т.В., доц. Зелінська О.В., доц. Антонов Ю.С., доц. Петелова Н.А., к.т.н. Поремський Ю.В., к.ф.-м.н. Фрей І.В.

На кафедрі діє ініціативна науково-дослідна тема «Методи, алгоритми та засоби автоматизованого проектування пристроїв управління обчислювальних систем» (2022-2025 р.р., науковий керівник д.т.н., професор Бабаєва Р.М.)

Стратегічним завданням наукового розвитку кафедри є формування наукової школи комп'ютерних технологій обробки даних та штучного інтелекту, її визнання науковою спільнотою.

Конференції, семінари, конкурси наукових робіт

У 2008 р. кафедрою започатковано щорічне проведення Всеукраїнської науково-практичної конференції студентів, аспірантів і молодих вчених «Прикладні інформаційні технології» та щорічної Всеукраїнської конференції «Комп'ютерні технології обробки даних» з питань комп'ютерних технологій обробки даних.

Щорічно проводяться конкурси студентських наукових робіт та студентської олімпіади за спеціальністю 122 «Комп'ютерні науки».

Наукові статті та монографії

Наукові інтереси викладачів кафедри:

- автоматизований синтез конкретного знань
- експертні системи та системи підтримки прийняття рішень
- інтегровані системи обробки статистичних даних
- інтелектуальні алгоритми управління графіками
- інтелектуальні технології аналізу даних в системах охорони
- інтелектуальні технології в технічній, медичній та освітній діагностиці
- інтелектуальні технології ідентифікації
- інтерпретація текстової інформації на основі нечітких концептуальних моделей
- інформаційні технології в управлінні виробничими об'єктами
- інформаційно-аналітичні та системні системи
- інформаційно-комунікаційні технології в освіті
- методи і технології проектування автоматизованих інформаційних систем
- методи оптимізації цифрових пристроїв керування
- методи проектування автоматизованих систем керування рухом транспортних засобів в міських міста
- методи розв'язку машинних задач теорії оболонок за допомогою граничних інтегральних рівнянь
- методологія комп'ютерного екологічного моніторингу
- нечіткі бази знань
- проектування систем підтримки прийняття рішень
- системи управління якістю процесами фінансово-господарської діяльності вищого навчального закладу
- системний аналіз на основі нечіткої логіки
- технології баз даних
- сучасні технології в прикладних дослідженнях

Наукові журнали

Наукові івенти

Прикладні аспекти сучасних міждисциплінарних досліджень:

https://jpasmd.donnu.edu.ua/issue/view/461

https://jpasmd.donnu.edu.ua/issue/view/400

https://jpasmd.donnu.edu.ua/issue/view/460

Конференція технології обробки даних:

https://jtdod.donnu.edu.ua/issue/view/369

https://jtdod.donnu.edu.ua/issue/view/411

https://jtdod.donnu.edu.ua/issue/view/443

Конференція Прикладні інформаційні технології:

https://jait.donnu.edu.ua/issue/view/318

https://jait.donnu.edu.ua/issue/view/427

https://jait.donnu.edu.ua/issue/view/428

https://jait.donnu.edu.ua/issue/view/464

вул. 600-річчя, 21, Вінниця, 21027, Україна
www.donnu.edu.ua

ДОДАТОК Е

Співробітництво кафедри Інформаційних технологій

Департамент ІТ Вінницької міської ради

12 листопада 2021 року кафедрою інформаційних технологій факультету інформаційних та прикладних технологій Донецького національного університету імені Василя Стуса в особі завідувачки кафедри Носкородової Тетяни Василівни та Департаментом ІТ Вінницької міської ради в особі директора Романюка Володимира Борисовича було підписано договір про співробітництво на 2021-2022 навчальний рік.

Предметом Договору є співробітництво між Сторонами щодо спільних навчальних, наукових та прикладних проектів, проходження практичної підготовки та стажування здобувачів вищої освіти та підвищення кваліфікації викладачів, сприяння працевлаштуванню здобувачів вищої освіти і випускників ДонНУ імені Василя Стуса.

У рамках підписаного договору про співробітництво між Донецьким національним університетом імені Василя Стуса та Департаментом ІТ Вінницької міської ради складено план заходів на 2021-2022 навчальний рік: це проведення бінарних лекцій, інтерактивного тренінгу, засідання круглого столу та залучення співробітників департаменту для проведення практичних занять в групах.

Завідувач кафедри Тетяна Носкородова зазначила, що підписання договору про співпрацю з Департаментом ІТ Вінницької міської ради – є важливим і перспективним, оскільки співпраця з роботодавцями – необхідна умова забезпечення повноцінного освітнього процесу у підготовці фахівців спеціальності 122 «Комп'ютерні науки» та 125 «Кібербезпека».

Департамент кіберполіції Національної поліції України

17.11.2021 року на 6 парі відбулася онлайн-зустріч студентів 2, 3 курсу спеціальності 125 «Кібербезпека» кафедри інформаційних технологій факультету інформаційних та прикладних технологій Донецького національного університету імені Василя Стуса з представниками відділу протидії кіберзлочинам у Вінницькій області Департаменту кіберполіції Національної поліції України. Ця зустріч відбувалася з метою професійної орієнтації студентів на підставі ознайомлення з особливостями роботи в кіберполіції, з основними етапи управління інцидентами інформаційної та кібербезпеки.

Сама назва департаменту говорить про те, що це – відділ поліції, який попереджує та розкриває злочини, скоєні за допомогою ІТ, – розповідає старший інспектор кіберполіції Максим Сергійович Александров. – Які це злочини? Dos-атаки, злами систем і мереж, шахрайства з використанням комп'ютерних мереж та мережі Інтернет, крадіжки грошей в платіжних системах та в банківській галузі, незаконне використання цифрового контенту (аудіовізуальних творів). Робота в підрозділах кіберполіції є гібридними діяльності в галузі ІТ і правовому полі, що суттєво відрізняє її від роботи в IT компанії або в відділі інформаційної безпеки на підприємстві.

«Галузь IT завжди привабливе молодь, але, по-перше, не все, що знаходиться в мережі, є законним, по-друге, сподіваюсь, жоден з вас зайдуть служба в підрозділах кіберполіції». Максим Сергійович розповів студентам цікаві історії з власного досвіду роботи в кіберполіції та відповів на питання студентів та викладачів кафедри.

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

_____

Повністю вказується ПІБ та статус (освітня (освітньо-наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)