МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА

Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

Форма навчання: заочна

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:
«Розробка CRM-системи для закладів вищої освіти»
здобувача **Ляпкало Юлії Анатоліївни**

Науковий керівник: к.е.н., доцент Лозовик Ю.М.

_____
(підпис)

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти (ЕК)

Завідувач кафедри: к.е.н., доцент Тішков Б. О.

_____
(підпис)

Київ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»

галузь знань 12 Інформаційні технології
спеціальність 122 Комп'ютерні науки

ПОГОДЖЕНО:
Керівник проєктної групи (гарант) освітньо-професійної програми
_____Устенко С.В.
«23» січня 2024 р.

ЗАТВЕРДЖУЮ:
Завідувач кафедри інформаційних систем в економіці
_____ Тішков Б.О.
«23» січня 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
здобувача вищої освіти**_____Ляпкало Юлії Анатоліївни_____**
(прізвище, ім'я, по батькові)
_____заочної_____ форми навчання
очної (денної), заочної, дистанційної

на підготовку кваліфікаційної магістерської роботи
на тему: «**Розробка CRM-системи для закладів вищої освіти**»

Тему затверджено наказом ректора Університету від «23» січня 2024р. №127-ст

Кваліфікаційна магістерська робота виконується на матеріалах
_____

План кваліфікаційної магістерської роботи

Розділ І. Дослідження та аналіз підходів до створення інформаційних управляючих систем предметної області

Розділ ІІ. Характеристика інформаційних управляючих систем та методи і моделі

Розділ ІІІ. Розроблення проєктних рішень та їх реалізація

Об'єкт дослідження: розробка системи управління взаємовідносини з клієнтами (CRM) для закладів вищої освіти.

Предмет дослідження: охоплює всебічне дослідження теоретичних основ, методологічних підходів і практичних міркувань, пов'язаних із розробкою CRM-рішення, специфічного для потреб і динаміки вищих навчальних закладів

Мета кваліфікаційної магістерської роботи: систематизація та узагальнення передового досвіду з розробки інноваційної CRM-систем університету, яка забезпечує взаємодію студентів, викладачів, вступників, методистів, батьків та інших зацікавлених осіб вищих навчальних закладів, а також проектування і розробку власної CRM-системи для університету.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I провести аналіз сучасного стану та тенденцій розвитку інформаційних систем для вищих навчальних закладів, оцінити ключові характеристики, переваги та недоліки наявних систем. Дослідити теоретичні засади та сучасні підходи до створення CRM-систем університету для управління взаємодією між основними учасниками, зокрема, в частині забезпечення доступу до матеріалів з урахуванням визначених ролей у системі.

У розділі II обрати методологію і технології, які найкраще підходять для розробки системи CRM, адаптованої до унікальних вимог академічного середовища; визначити архітектуру та функціональні можливості системи CRM для вищих навчальних закладів.

У розділі III створити надійну схему бази даних, адаптовану до багатогранних вимог академічних установ у рамках CRM; визначити програмну інфраструктуру, необхідну для безперебійної інтеграції та розгортання системи CRM в університеті; розробити структуру технічної підтримки для забезпечення безперебійної роботи та постійного вдосконалення системи CRM.

Завдання підготував
науковий керівник _____ Ю.М. Лозовик

(підпис)

«23» січня 2023 р.

Завдання одержав
здобувач _____ Ю.А. Ляпкало

(підпис)

«23» січня 2023 р

Реферат

Кваліфікаційна магістерська робота містить 70 сторінки, 3 таблиці, 15 рисунків, список використаних джерел з 56 найменувань.

«Розробка CRM системи для закладів вищої освіти»

Об'єкт дослідження: Основною метою цієї кваліфікаційної магістерської роботи є процес розробки системи управління взаємовідносинами з клієнтами (CRM) для закладів вищої освіти.

Предмет дослідження: Предмет охоплює всебічне дослідження теоретичних основ, методологічних підходів і практичних міркувань, пов'язаних із розробкою CRM-рішення, специфічного для потреб і динаміки вищих навчальних закладів.

Мета та завдання дослідження: Основною метою цієї кваліфікаційної магістерської роботи є розробка інноваційної системи CRM, яка оптимізує взаємодію студентів та викладачів у вищих навчальних закладах. Для досягнення цієї головної мети були визначені такі завдання:

- проведено поглиблений аналіз існуючих систем CRM, що використовуються у вищих навчальних закладах.
- обрано методології і технологій, які найкраще підходять для розробки системи CRM, адаптованої до унікальних вимог академічного середовища.
- визначено архітектуру та функціональні можливості системи CRM для вищих навчальних закладів.
- створено надійну схему бази даних, адаптовану до багатогранних вимог даних академічних установ у рамках CRM.
- окреслено програмну інфраструктуру, необхідну для безперебійної інтеграції та розгортання системи CRM у різних відділах установи.
- розроблено структуру технічної підтримки для забезпечення безперебійної роботи та постійного вдосконалення системи CRM після впровадження.

Теоретичне, методологічне та практичне значення отриманих результатів. Протягом цього дослідження було проведено широке вивчення практик CRM у секторі вищої освіти. Це дослідження охоплює аналіз стратегій залучення студентів, адміністративних робочих процесів і технологічної інфраструктури, поширеної в академічних установах. Розробляючи спеціалізовану систему CRM, адаптовану до унікальних потреб вищої освіти, це дослідження намагається покращити досвід студентів, оптимізувати адміністративні операції та сприяти значущим відносинам між навчальними закладами та їхніми зацікавленими сторонами.

Рік виконання кваліфікаційної магістерської роботи – 2024.

Рік захисту роботи – 2024.

Ключові слова: CRM, вища освіта, залучення студентів, адміністративна ефективність.

В і д г у к

про кваліфікаційну магістерську роботу
здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»
освітньо-професійної програми «Інформаційні управляючі системи і технології»
Ляпкало Юлії Анатоліївни
на тему « **Розробка CRM-системи для закладів вищої освіти** »

1. Актуальність теми:

Актуальність дипломної роботи полягає у тому, що сучасні заклади вищої освіти зацікавлені у розробці власної інформаційної системи, яка б забезпечувала їх незалежність від сторонніх розробників та надавала основним користувачам (викладачам, студентам, аспірантам, вступникам та іншим учасникам) системи якісну і своєчасну інформацію. Також надзвичайно важливо для закладів вищої освіти розширювати свої сервіси, покращувати функціонал систем, доповнювати систему пошуку, пропонувати якісну систему підтримки тощо.

2. Позитивні риси кваліфікаційної магістерської роботи:

Автором проведено теоретико-практичне дослідження та аналіз підходів для створення CRM-системи університету, проаналізовані аналоги програмних рішень та здійснено проектування власної системи, яка мала б інтегруватися в інформаційну систему університету. В той же час, автор мав би більше уваги приділити створенню робочого прототипу. Автор роботи обмежився розробкою окремих веб-сторінок системи.

3. Наявність самостійних розробок автора.

Представлена кваліфікаційна магістерська робота є самостійною розробкою автора, зокрема, в роботі описано та реалізовано окремі функціональні рішення для основних користувачів CRM-системи університету – викладачів і студентів. Також реалізовано веб-додаток інформаційної системи з використанням сучасних технологій та фреймворків, таких як Hibernate, SpringBoot, SpringSecurity та інших, розроблено різноманітні модулі системи, які мають полегшити взаємодію між основними учасниками системи.

4. Цінність теоретичних висновків та практичних рекомендацій:

У ході виконання кваліфікаційної магістерської роботи Ляпкало Ю.А. проаналізувала наявні теоретичні підходи до створення інформаційних систем для університету, на основі чого розробила проєкт CRM-системи університету, який поєднує функціонал для різноманітних учасників та є вигідним для навчальних закладів з погляду оптимізації витрат на розробку та впровадження подібних рішень.

5. Наявність недоліків:

Представлене проєктне рішення (прототип) потребує подальшого вдосконалення й розвитку з урахуванням функціональних вимог основних учасників CRM-системи університету. У роботі представлені та реалізовані лише окремі рішення для менеджера, студента та викладача, тобто наразі система охоплює рішення лише для окремих учасників.

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК:

Загалом, представлена кваліфікаційна магістерська робота відповідає встановленим вимогам методичних вказівок щодо структури, обсягу і змісту та **рекомендується до захисту** з позитивною оцінкою.

Науковий керівник: доцент кафедри інформаційних систем у економіці, доцент, к.е.н. Лозовик Ю.М.

_____

(підпис)

«15» травня 2024 р.

Рецензія
на кваліфікаційну магістерську роботу
здобувача вищої освіти
Ляпкало Юлії Анатоліївни

Тема «Розробка CRM-системи для закладів вищої освіти»

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення. Тема розробки CRM-системи для вищих навчальних закладів є актуальною в сучасному конкурентному академічному середовищі. Оскільки університети прагнуть покращити досвід студентів, покращити показники утримання та сприяти зміцненню стосунків з випускниками, адаптована система CRM стає незамінною. У роботі вдало визнається ця потреба та підкреслюється важливість впровадження стратегічних технологій у вищій освіті.
Якість проведеного дослідження. Я можу відзначити суворий підхід до проведення дослідження для цієї дисертації. Ретельний огляд літератури, доповнений порівняльним аналізом, відображає повне розуміння як теоретичних основ, так і практичних міркувань. Об'єднання емпіричних даних і галузевих ідей ще більше посилює довіру та актуальність результатів дослідження.
Позитивні риси кваліфікаційної магістерської роботи. Глибина огляду проблем галузі демонструють міцне розуміння систем CRM, управління вищою освітою та технологічних тенденцій. Розглянуті та запропоновані рішення для болей зацікавлених сторін, включаючи адміністраторів, викладачів, студентів і випускників. Розуміння їхніх унікальних потреб і очікувань має вирішальне значення для розробки системи CRM, яка справді відповідає вимогам сектору вищої освіти. Цей аналіз слугує трампліном для пропозиції адаптованої системи CRM, яка відповідає конкретним викликам і можливостям у вищій освіті.
Зауваження. Хоча робота представляє міцну теоретичну основу та переконливі аргументи для розробки системи CRM для вищої освіти, важливо визнати необхідність постійної перевірки та вдосконалення. У міру того, як реалізуються ці ідеї, безперервний зворотний зв'язок і ітераційне тестування будуть важливими для забезпечення ефективності та актуальності рішень.
Практична значимість висновків і рекомендацій Висновки, зроблені в цій роботі, пропонують цінне уявлення про потенційний вплив систем CRM на операційну ефективність та залучення зацікавлених сторін закладів вищої освіти. Втіливши ці висновки в дієві рекомендації, є можливість не лише розширити функціонал таких продуктів, але й зробити вагомий внесок у ширший освітній сектор. Практичне значення цього дослідження полягає в його здатності сприяти позитивним змінам і надавати університетам можливість процвітати в умовах дедалі більшої конкуренції.

Рецензент:
"PlayTech", software developer Федосенко Е. Д.

ЗМІСТ

ВСТУП.....4

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ПРЕДМЕТНОЇ ОБЛАСТІ 8

1.1 АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ РОЗРОБЛЕНИХ ДЛЯ ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДІВ 8

1.1.1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 8

1.1.2 ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ 12

1.2 ОБҐРУНТУВАННЯ ВИБОРУ ПІДХОДІВ І ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ УПРАВЛЯЮЧОЇ СИСТЕМИ.....19

РОЗДІЛ 2. ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ.....24

2.1 СТРУКТУРА І ХАРАКТЕРИСТИКА СИСТЕМИ.....24

2.2 МЕТОДИ ТА МОДЕЛІ В ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМАХ І ТЕХНОЛОГІЯХ.....33

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ.....36

3.2. ПРОЄКТУВАННЯ БАЗИ ЗНАНЬ ІНФОРМАЦІЙНОЇ УПРАВЛЯЮЧОЇ СИСТЕМИ ТА/АБО ЗАСОБИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ 39

3.3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 45

3.3.1 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....45

3.3.3 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ДЛЯ СТВОРЕННЯ ПРИКЛАДНОГО 55

ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ 55

3.3.4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 56

3.3.5 КЕРІВНИЦТВО (ІНСТРУКЦІЯ) КОРИСТУВАЧА 58

3.4. ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 61

3.4.1 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ 61

3.4.2 РЕСУРСНІ ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ . 62

3.5. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ УПРАВЛЯЮЧОЇ СИСТЕМИ 64

ВИСНОВКИ 70

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 72

ДОДАТКИ 2

Вступ

Сучасна освітня сфера стоїть перед безпрецедентними викликами, вимагаючи постійного вдосконалення управлінських практик для забезпечення високої якості навчання та задоволення потреб усіх учасників освітнього процесу. У цьому контексті використання систем управління відносинами з клієнтами (CRM) стає надзвичайно важливим для ефективної організації комунікації, взаємодії та адміністративних процесів в університетах. Вищі навчальні заклади працюють у все більш динамічному середовищі, яке характеризується демографічними змінами студентів, загостренням конкуренції та зростанням попиту на персоналізовані послуги. У відповідь на ці виклики впровадження CRM систем стає ключовою стратегією. Цей вступ має на меті висвітлити нагальну актуальність і значення розробки CRM систем, розроблених спеціально для закладів вищої освіти.

В останні роки у вищій освіті відбулася зміна парадигми в бік підходів, орієнтованих на студента, з наголосом на персоналізованому залученні та стратегіях утримання. Визнаючи цю зміну, Крістіан Педрон і Танасіс Дарадуміс в своїй статі для «International Journal of Services Technology and Management», підкреслили необхідність впровадження систем CRM для закладів як засобу сприяння значущим зв’язкам зі студентами, покращення надання послуг та оптимізації адміністративних процесів.[7] Однак, незважаючи на зростаючий інтерес до CRM в академічних колах, залишається дефіцит всебічних досліджень щодо її індивідуального застосування у вищих навчальних закладах. Розробка та впровадження CRM системи для вищих навчальних закладів має величезний потенціал у покращенні студентського досвіду, оптимізації роботи викладацького складу та підвищенні ефективності адміністративних процесів. Але при цьому існують виклики та невирішені питання, пов'язані

з адаптацією загальних CRM підходів до конкретних потреб освітнього сектору та унікальності його процесів.

Це дослідження має на меті подолати цей розрив, окресливши унікальні виклики та можливості, властиві розробці систем CRM для закладів вищої освіти. Шляхом ретельного вивчення існуючої літератури та емпіричних досліджень це дослідження намагається з’ясувати відмінні аспекти CRM у контексті вищої освіти, спираючись на фундаментальний внесок Інма Родрігес-Ардура і Хав'єр Фаулін з університету Оберта де Каталонія. Визначаючи раніше невивчені аспекти всеосяжного дискурсу CRM, це дослідження має на меті зайняти нішу в науковому середовищі, вносячи нові ідеї та розв’язуючи складні проблеми, з якими стикаються вищі навчальні заклади. Аналіз останніх досліджень та публікацій свідчить про значний інтерес до застосування CRM в освітній сфері. Однак, багато робіт обмежуються загальними концепціями, не звертаючи належної уваги на специфіку освітнього процесу та потреби його учасників. Тому, існує потреба в дослідженні, спрямованому на розробку та впровадження CRM системи, яка враховуватиме усі особливості та вимоги сучасної вищої освіти. За своєю суттю, це дослідження бореться з головною науковою проблемою оптимізації залучення студентів, утримання та задоволеності студентів шляхом стратегічного розгортання систем CRM у вищій освіті. З’ясовуючи суть цієї проблеми, це дослідження прагне надати дієві рекомендації та ідеї, які можуть стати основою для розробки та впровадження ефективних стратегій CRM, адаптованих до унікальних потреб і динаміки закладів вищої освіти.

Метою даної магістерської дипломної роботи є створення адаптованої до потреб університетського середовища CRM системи з використанням мови програмування Java, зокрема Spring Framework. Для досягнення цієї мети передбачено вирішення наступних завдань:

- Детальний аналіз існуючих практик управління вищими навчальними закладами та огляд доступних CRM рішень.
- Розробка та узгодження архітектури CRM системи з урахуванням унікальних особливостей освітнього процесу.
- Реалізація ключових функціональних модулів системи з використанням сучасних технологій програмування.
- Проведення тестування та пілотного запуску розробленої CRM системи для перевірки її ефективності та придатності.
- Оцінка ефективності та впливу систем CRM на залучення, утримання та задоволеність студентів.

Об'єктом цього дослідження є процес залучення, утримання та задоволеності студентів у закладах вищої освіти, а *предмет* охоплює методологічні принципи та економіко-математичні методи, що стосуються проектування систем CRM. Для досягнення поставлених цілей та вирішення завдань будуть використані методи аналізу літературних джерел, емпіричні дослідження, проектування програмного забезпечення та тестування.

Актуальність цієї теми підкреслюється необхідністю для вищих навчальних закладів розвивати змістовні стосунки зі студентами, випускниками, викладачами та іншими зацікавленими сторонами. В епоху, яка характеризується цифровою трансформацією та демографічними змінами, здатність ефективно залучати й утримувати учасників є найважливішою для інституційного успіху. Таким чином, вивчення інноваційних підходів до управління інформацією, зокрема через призму систем CRM, має значні перспективи у вирішенні цих проблем і максимізації можливостей для успіху студентів та інституційного просування.

В основі цих дослідницьких зусиль лежить пошук нових, прогресивних рішень, які можна перетворити на дієві стратегії для реалізації в реальному світі. Використовуючи знання як теоретичних досліджень, так і практичних застосувань, це дослідження прагне розробити нові методології та рамки, які є не лише науковими, але й безпосередньо застосовними до операційних реалій вищих навчальних закладів. Завдяки синтезу передових результатів досліджень і практичного досвіду, це дослідження має на меті накреслити курс до трансформаційних змін у тому, як інституції залучають і підтримують своїх виборців.

На практиці результати цього дослідження мають далекосяжні наслідки для вищих навчальних закладів, пропонуючи практичні рішення та рекомендації. Крім того, сприяючи органічній інтеграції науки та практики, це дослідження підкреслює важливість подолання розриву між академічними колами та промисловістю для вирішення нагальних викликів і стимулювання суттєвих змін.

Підсумовуючи, розробка систем CRM для закладів вищої освіти являє собою конвергенцію теоретичних досліджень, методологічних інновацій і практичного застосування. Вирішуючи нагальні виклики, розвиваючи наукові знання та пропонуючи дієві рекомендації, ці дослідницькі зусилля спрямовані на каталізацію трансформаційних змін у середовищі вищої освіти. Отже, дана дипломна робота відповідає вимогам сучасного університетського управління та відображає актуальні потреби освітньої сфери. Результати дослідження можуть бути корисними для впровадження покращень у вищі навчальні заклади та підвищення їхньої ефективності.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ПРЕДМЕТНОЇ ОБЛАСТІ

1. 1 Аналіз існуючих інформаційних управляючих систем розроблених для вищих навчальних закладів

1.1.1 Дослідження предметної області

Розробка систем управління взаємовідносинами з клієнтами (CRM), призначених для закладів вищої освіти, є критичним перетином між інформаційними технологіями (IT) і академічним середовищем. У контексті IT-індустрії в Україні, яка зазнала значного зростання експорту послуг за останнє десятиліття разом з дефіцитом людських ресурсів, дослідження систем CRM у секторі вищої освіти є особливо актуальним.

Важливо встановити чітке розуміння ключових термінів, таких як CRM, вища освіта та системи управління інформацією. CRM відноситься до стратегій, технологій і практик, спрямованих на управління та підтримання відносин з клієнтами або складовими. Вища освіта охоплює вищі навчальні заклади, які пропонують наукові ступені та проводять дослідження, тоді як системи управління інформацією позначають інструменти та технології, які використовуються для збору, зберігання та аналізу даних для прийняття рішень.

Основним об’єктом дослідження є впровадження та оптимізація систем CRM у вищих навчальних закладах. Це передбачає розуміння різних процесів, пов’язаних із залученням студентів, наймом, утриманням і підтримкою. Процеси можуть включати, серед іншого, управління вступом, обслуговування студентів, зв’язки з випускниками, збір коштів та академічне консультування. Розуміння цих процесів має важливе значення

для розробки систем CRM, які відповідають конкретним потребам і цілям закладів вищої освіти.

У предметній області існують численні зв’язки та взаємодії між різними зацікавленими сторонами, системами та джерелами даних. Наприклад, системи CRM у вищих навчальних закладах повинні взаємодіяти з інформаційними системами студентів, системами управління навчанням і базами даних випускників, щоб забезпечити комплексне уявлення про взаємодію та переваги учасників. Крім того, системи CRM полегшують взаємодію між студентами, викладачами, співробітниками, випускниками, донорами та іншими зовнішніми партнерами, формуючи загальний досвід студентів та інституційні результати.

Дослідження в цій галузі мають також розглядати унікальні виклики та можливості, властиві розробці систем CRM для вищої освіти. Проблеми можуть включати складність інтеграції даних, проблеми конфіденційності, стійкість до змін і обмежені ресурси для впровадження та обслуговування системи. Навпаки, можливості можуть виникнути завдяки використанню нових технологій, таких як штучний інтелект, машинне навчання та прогнозна аналітика, для персоналізації взаємодії студентів, оптимізації зусиль щодо найму та підвищення інституційної ефективності.

Нарешті, важливо враховувати ширші контекстуальні фактори, що впливають на розробку та впровадження систем CRM у секторі вищої освіти. Це включає нормативні рамки, що регулюють конфіденційність і безпеку даних, культурні норми в академічних установах, інституційні місії та цілі, а також зміни очікувань студентів та інших учасників. Розуміння цих контекстуальних факторів має вирішальне значення для розробки систем CRM, які є не лише технічно надійними, але й узгодженими культурно та інституційно.

Економічний зміст проблем у предметній області розробки системи CRM для закладів вищої освіти обертається навколо оптимізації розподілу

ресурсів, підвищення операційної ефективності та максимального задоволення зацікавлених сторін. Рішення в цій сфері часто передбачають збалансування конкуруючих пріоритетів, таких як набір студентів, утримання та академічна якість, одночасно керуючи обмеженими фінансовими ресурсами. Об’єкти дослідження під управлінням систем CRM включають майбутніх студентів, поточних студентів, випускників, викладачів, персонал, донорів та інституційних партнерів. Роль і актуальність прийняття рішень у цьому контексті полягають у виконанні функцій управління відносинами, ресурсами та інституційною репутацією, що в кінцевому підсумку сприяє успіху студента та інституційному просуванню.

Суть проблемних ситуацій у розробці системи CRM для вищих навчальних закладів часто полягає в покращенні комунікації та взаємодії із зацікавленими сторонами, оптимізації адміністративних процесів та використанні інформації, що керується даними, для прийняття рішень. Прийняття рішень відбувається на різних організаційних рівнях, від стратегічного планування на інституційному рівні до тактичного втручання на відомчому чи програмному рівні. Альтернативні варіанти рішень можуть включати вибір постачальників програмного забезпечення CRM, розробку комунікаційних стратегій, розподіл бюджетних ресурсів і впровадження політики управління даними. Періодичність і тривалість прийняття рішень залежать від характеру проблеми, при цьому деякі рішення є звичайними, а інші більш стратегічними або випадковими. Обмеження у прийнятті рішень можуть випливати з бюджетних обмежень, технологічних бар’єрів, нормативних вимог та організаційної культури, серед інших факторів. Критерії прийняття рішень часто включають міркування щодо економічної ефективності, масштабованості, впливу на зацікавлених сторін і відповідності інституційним цілям і цінностям.

Середовище прийняття рішень у розробці системи CRM для закладів вищої освіти залежить від багатьох факторів, включаючи технологічний прогрес, динаміку ринку, нормативні зміни, демографічні зміни та інституційні пріоритети. Характер і ступінь впливу різняться, причому такі фактори, як наявність фінансування та демографічний склад студентів, справляють значний тиск на тих, хто приймає рішення. Крім того, внутрішні фактори, такі як організаційна структура, динаміка керівництва та очікування зацікавлених сторін, формують контекст прийняття рішень.

Оптимальне прийняття рішень у розробці системи CRM для вищих навчальних закладів потребує поєднання технічного досвіду, знань галузі та стратегічної кмітливості. Особи, які приймають рішення, повинні мати глибоке розуміння принципів і технологій CRM, а також уявлення про унікальну динаміку сектору вищої освіти. Знайомство з аналітикою даних, методологіями управління проектами, принципами управління змінами та стратегіями залучення зацікавлених сторін також є важливим для прийняття обґрунтованих рішень.

Ситуації прийняття рішень у розробці CRM-системи для вищих навчальних закладів характеризуються складністю, невизначеністю та взаємопов’язаністю. Взаємодія економічних, технологічних та організаційних факторів вимагає тонкого підходу до прийняття рішень, балансу між короткостроковими імперативами та довгостроковими стратегічними цілями. Успішне прийняття рішень залежить від здатності синтезувати різноманітні джерела інформації, передбачати майбутні тенденції та адаптуватися до обставин, що змінюються.

Розробка інформаційних систем підтримки прийняття рішень для розвитку системи CRM у вищій освіті є обов’язковою для вирішення багатогранних викликів і можливостей, притаманних цій галузі. Сценарії підтримки прийняття рішень, такі як моделі прогнозної аналітики, інструменти візуалізації інформаційної панелі та системи автоматизації

робочого процесу, можуть підвищити ефективність і ефективність процесів прийняття рішень. Використовуючи технологічну підтримку прийняття рішень, установи можуть оптимізувати розподіл ресурсів, пом’якшити ризики та отримати вигоду від нових можливостей, зрештою сприяючи сталому зростанню та конкурентній перевагі.

1.1.2 Дослідження інформаційних управляючих систем

Ефективність систем CRM у вищій освіті залежить від їхньої здатності надавати студентам персоналізований і контекстуально відповідний досвід. Однак багато існуючих систем не мають переваги у цьому відношенні, покладаючись на статичну сегментацію та загальні комунікаційні стратегії. Щоб подолати це обмеження, нові тенденції, такі як прогнозна аналітика, машинне навчання та штучний інтелект, пропонують багатообіцяючі шляхи для підвищення деталізації та витонченості стратегій залучення студентів у системах CRM.

Крім того, етичні аспекти управління даними та конфіденційності постають як першочергові питання в контексті впровадження CRM у вищій освіті. Оскільки навчальні заклади збирають і аналізують величезні обсяги студентських даних для інформування про прийняття рішень і персоналізованого втручання, захист конфіденційності та забезпечення безпеки даних стає обов’язковим. Таким чином, будь-яка система CRM, розроблена для вищої освіти, повинна відповідати суворим правилам захисту даних і втілювати принципи прозорості, згоди та підзвітності.

Незважаючи на ці проблеми, потенційні переваги систем CRM у вищій освіті є різноманітними. Централізувавши інформацію про взаємодію, CRM-системи дозволяють навчальним закладам отримати цілісне розуміння поведінки, уподобань і потреб студентів. Це, у свою чергу, сприяє цілеспрямованим втручанням, персоналізованим службам

підтримки та проактивним ініціативам утримання, зрештою сприяючи культурі успіху та задоволеності студентів та менеджменту.

Це був виклик, з яким зіткнувся Університет Джорджа Вашингтона, який має понад 26 000 студентів і 14 шкіл і коледжів.

Університет Джорджа Вашингтона (GW), престижний приватний дослідницький університет, розташований у Вашингтоні, округ Колумбія, використовує комплексну систему управління взаємовідносинами з клієнтами (CRM) для управління своїми відносинами з різними зацікавленими сторонами, включаючи майбутніх студентів, поточних студентів, випускників, викладачів, співробітників, донорів. та інституційні партнери.

CRM-система GW пропонує широкий спектр функцій, адаптованих до сектору вищої освіти. Ці функції можуть включати керування потенційними клієнтами, обробку заявок, консультування студентів, управління подіями, залучення випускників, збір коштів та аналітичні можливості для відстеження та оптимізації взаємодії зі складовими протягом усього життєвого циклу.

«Коли ми думаємо про життєвий цикл студента в GW — від зацікавлених студентів до їх вступу, навчання та випуску, а потім життя випускників — існує так багато даних про кожен із цих етапів», — сказав Джаред Джонсон, асоційований віце-президент. за академічні технології та клієнтський досвід у GW. «Різні компоненти університету будують нові відносини з цією складовою, навіть якщо дані та знання про цю складову вже існують в інших частинах університету».

За словами Джонсона, це завдання ускладнюється тим, що тисячі учасників «взаємодіють з університетом через різних персонажів». Деякі люди одночасно є і викладачами, і ненауковцями. Деякі є студентами і працівниками одночасно. Деякі співробітники також є випускниками та

батьками нинішніх студентів і взаємодіють з GW у всіх трьох сферах діяльності, і кожен потік спілкування записувався в окремому відділенні.[1]

Концептуальна схема CRM-системи GW включає централізовану базу даних, що містить комплексні профілі складових, інтегровану з різними модулями та робочими процесами, призначеними для оптимізації зв'язку, автоматизації процесів і надання корисної інформації для прийняття рішень.

CRM-система GW орієнтована на підтримку процесів прийняття рішень, пов'язаних із набором студентів, вступом, управлінням зарахуванням, академічним консультуванням, ініціативами успіху студентів, кампаніями збору коштів, зв'язками з випускниками та інституційним плануванням. Завдання прийняття рішень можуть включати виявлення студентів групи ризику, орієнтацію на роботу, персоналізацію комунікацій та оптимізацію розподілу ресурсів.

CRM-система GW, ймовірно, використовує надійну архітектуру бази даних, здатну обробляти великі обсяги структурованих і неструктурованих даних. Вони можуть використовувати хмарні рішення для зберігання масштабованості та доступності. Крім того, система може включати базу знань, що включає інституційну політику, найкращі практики та історичні дані для інформування щодо прийняття рішень. Програмні компоненти можуть включати платформи CRM, такі як Salesforce або Microsoft Dynamics, доповнені інтеграцією з іншими корпоративними системами, такими як системи інформації для студентів і системи управління навчанням. Технічна підтримка необхідна для обслуговування системи, навчання користувачів і усунення несправностей.

GW може використовувати інноваційні інформаційні технології, такі як штучний інтелект (AI), машинне навчання (ML), аналітика великих даних, Інтернет речей (IoT), доповнена реальність (AR) і віртуальна реальність (VR), щоб посилити залучення учасників, прогнозне

моделювання та персоналізований досвід. Інформаційно-організаційні аспекти передбачають узгодження стратегій CRM з інституційними цілями, сприяння культурі прийняття рішень на основі даних і забезпечення дотримання правил конфіденційності.

Їх CRM-система може інтегрувати спеціалізовані системи підтримки інформації та прийняття рішень, засновані на таких платформах, як Arduino, ESP8266, Orange Pi або Raspberry Pi, для конкретних випадків використання, таких як управління об’єктами кампусу з підтримкою Інтернету речей або персоналізовані послуги для студентів.

Архітектурні рішення в системі CRM GW можуть включати розробку інтелектуальних компонентів, таких як чат-боти, системи рекомендацій, моделі прогнозної аналітики та інструменти аналізу настроїв для автоматизації завдань, персоналізації взаємодії та оптимізації результатів.

Користувачі CRM-системи GW, включаючи адміністраторів, викладачів, співробітників, студентів, випускників і донорів, можуть засвідчити кілька переваг, отриманих від її використання. Ці переваги можуть включати покращення зв’язку та співпраці, підвищення ефективності роботи, прийняття рішень на основі даних, персоналізований досвід і зміцнення стосунків із учасниками.

Підсумовуючи, аналіз CRM-системи GW ілюструє її ключову роль в управлінні відносинами та діяльністю в складній екосистемі вищої освіти. Використовуючи передові системи управління інформацією та інноваційні технології, GW демонструє свою відданість наданню виняткового досвіду та розвитку інституційної досконалості.

Ось ще кілька прикладів систем CRM, які зазвичай використовуються в навчальних закладах:

- Salesforce for Education:

Ця платформа пропонує низку рішень CRM для навчальних закладів через свою освітню хмару. Він надає інструменти для управління взаємодією зі студентами, набору, зарахування, збору коштів і залучення випускників. Salesforce також пропонує рішення для шкіл, коледжів і університетів.[2]

- Ellucian CRM:

  CRM-рішення, спеціально розроблені для вищих навчальних закладів. Платформа CRM допомагає залучати студентів, керувати зарахуванням, утримувати їх і залучати випускників. Він інтегрується з іншими продуктами Ellucian, такими як студентські інформаційні системи, для безперешкодного керування даними.[3]

- TargetX:

  платформа яка тепер є частиною Salesforce.org, надає рішення CRM, спеціально розроблені для вищих навчальних закладів. Вона зосереджена на наборі студентів, управлінні реєстрацією та спілкуванні з майбутніми студентами. TargetX CRM інтегрується з Salesforce for Education, щоб забезпечити комплексне рішення.[4]

- Slate від Technolutions:

  це CRM та програмне забезпечення для вступу, розроблене для вищих навчальних закладів. Він спрощує процес вступу, полегшує спілкування з майбутніми студентами та пропонує інструменти для керування заявками та зарахування.[5]

Це лише кілька прикладів систем CRM, які зазвичай використовуються в навчальних закладах. Вибір CRM залежить від конкретних потреб, розміру та бюджету установи, а також вимог інтеграції з іншими системами та платформами.

Для прикладу пропоную провести порівняльну характеристику двох популярних рішень : Salesforce for Education та Ellucian CRM.

Таблиця 1.1 – Порівняльна характеристика популярних CRM систем для закладів освіти

	Salesforce for Education	Ellucian CRM
Цільова аудиторія	Обслуговує широкий спектр навчальних закладів, включаючи школи, коледжі та університети, пропонуючи гнучку платформу CRM, яку можна налаштувати відповідно до різноманітних потреб.	Насамперед зосереджується на закладах вищої освіти, надаючи індивідуальні рішення для коледжів і університетів із спеціальними функціями для набору студентів, управління зарахуванням і залучення випускників.
Налаштування та гнучкість	Salesforce відомий своїм високим рівнем налаштування та гнучкості, що дозволяє установам адаптувати платформу CRM до своїх конкретних робочих процесів, процесів і вимог. Він пропонує широкі можливості налаштування через свою екосистему AppExchange.	хоча Ellucian CRM пропонує параметри налаштування, він може бути не таким гнучким і легким в налаштуванні, як Salesforce. Однак він надає готові модулі та функції, спеціально розроблені для вищих навчальних закладів, що спрощує впровадження для коледжів та університетів.
Інтеграція та екосистема	Salesforce має величезну екосистему з численними інтеграціями сторонніх розробників, доступними через Salesforce AppExchange. Він забезпечує повну інтеграцію з іншими продуктами Salesforce і широким спектром зовнішніх систем, що дозволяє комплексно керувати даними та аналізувати їх.	Ellucian CRM добре інтегрується з іншими продуктами Ellucian, такими як студентські інформаційні системи (SIS), фінанси та системи управління персоналом, які зазвичай використовуються у вищих навчальних закладах. Хоча він може пропонувати менше інтеграцій порівняно з Salesforce, він забезпечує глибоку інтеграцію з основними освітніми системами.

Продовження таблиці 1.1		
Інтерфейс користувача та взаємодія з користувачем	Salesforce пропонує сучасний та інтуїтивно зрозумілий інтерфейс користувача з настроюваними інформаційними панелями та звітами. Його користувацький досвід (UX) загалом добре оцінений, хоча рівень складності може змінюватися залежно від ступеня налаштування.	Ellucian CRM надає зручний інтерфейс, адаптований до потреб фахівців вищої освіти. Його користувальницький інтерфейс розроблено для оптимізації завдань, пов'язаних із набором студентів, керуванням зарахуванням і залученням випускників, пропонуючи більш спеціалізований досвід для освітніх користувачів.
Підтримка та навчання	Salesforce надає розширену документацію, навчальні ресурси та надійну спільноту для користувачів, щоб навчатися та вирішувати проблеми. Крім того, Salesforce пропонує різні варіанти підтримки, зокрема онлайн-підтримку, навчальні курси та консультаційні послуги.	Ellucian пропонує комплексну підтримку та навчання, адаптоване до потреб закладів вищої освіти. Він пропонує навчальні програми, вебінари, конференції для користувачів і спеціальну команду підтримки, яка спеціалізується на освітніх рішеннях CRM.

Джерело: сформовано автором на основі виконаного дослідження

Окремо хочу виділити сильна та слабкі сторони кожної із систем:

Таблиця 1.2 – Порівняння сильних та слабких сторін CRM систем для закладів освіти

	Salesforce for Education	Ellucian CRM
Сильні строни	- Високий рівень налаштування та гнучкості - Розгалужена екосистема з інтеграцією сторонніх розробників. - Сучасний інтерфейс .	- Розроблено спеціально для вищих навчальних закладів. Повна інтеграція з іншими продуктами Ellucian. - Зручний інтерфейс, призначений для освітян.

Продовження таблиці 1.2		
Слабкі сторони	- Складність може вимагати більших ресурсів для налаштування та впровадження. - Може вимагати додаткових інвестицій у навчання та підтримку для комплексного використання.	- Обмежена гнучкість у порівнянні з Salesforce для дуже індивідуальних потреб. - Довіра до екосистеми Ellucian може обмежити інтеграцію зі сторонніми системами.

Джерело: сформовано автором на основі виконаного дослідження

Зрештою, вибір між Salesforce for Education і Ellucian CRM залежить від таких факторів, як конкретні потреби та пріоритети навчального закладу, рівень необхідного налаштування, вимоги до інтеграції та доступні ресурси для впровадження та підтримки.[6]

Аналіз існуючих систем управління інформацією, зокрема систем CRM, розроблених для вищих навчальних закладів, показує ландшафт, що характеризується як можливостями, так і проблемами. Оскільки навчальні заклади все більше віддають пріоритет підходам, орієнтованим на студента, і прагнуть до укріплення стосунків, впровадження систем CRM набуло популярності. Однак критичний аналіз існуючих систем підкреслює кілька ключових міркувань.

1. 2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи

Приступаючи до розробки CRM-системи для закладів вищої освіти, було запропоновано низку стратегічних концептуальних рішень. Ці

рішення охоплюють методологічні, технологічні та організаційні виміри, усі спрямовані на забезпечення ефективності, масштабованості та узгодження системи з унікальними вимогами сектору вищої освіти.

Застосовуючи орієнтований на користувача підхід до проектування, запропонована система CRM віддаватиме пріоритет потребам і перевагам ключових зацікавлених сторін, включаючи студентів, викладачів та співробітників. Цей підхід передбачає проведення широкого дослідження користувачів, використання тестування зручності використання та повторення прототипів дизайну, щоб забезпечити зручність використання, доступність системи та загальне задоволення користувачів. Крім того, процес розробки CRM дотримуватиметься принципів гнучкості, сприяючи гнучкості та швидкому реагуванню на зміни вимог і відгуків зацікавлених сторін протягом життєвого циклу проекту.

Технологічно запропонована система CRM використовуватиме потужність інноваційних інформаційних технологій для підвищення її функціональності та продуктивності. Зокрема, система буде розроблена з використанням Java, універсальної та широко поширеної мови програмування, відомої своєю надійністю та масштабованістю. Використовуючи екосистему Java, система використовуватиме Spring Framework — комплексну платформу для створення корпоративних додатків Java — для оптимізації розробки, стимулювання повторного використання коду та полегшення інтеграції з іншими системами та службами. Крім того, керування даними системи здійснюватиметься за допомогою PostgreSQL, потужної системи керування реляційними базами даних із відкритим кодом, відомої своєю надійністю, розширюваністю та розширеними функціями, такими як відповідність ACID, повнотекстовий пошук і підтримка JSON. Завдяки використанню цих технологій система CRM буде добре обладнана для виконання складних вимог до обробки та управління даними, властивих сфері вищої освіти.

Організаційно запропонована система CRM буде ретельно розроблена, щоб узгодити її з інституційною структурою, процесами та цілями закладів вищої освіти. Це передбачає планування робочих процесів, потоків даних і каналів зв’язку для оптимізації адміністративних процесів, сприяння співпраці між відділами та забезпечення цілісності та безпеки даних. Крім того, розробка та впровадження системи здійснюватиметься з дотриманням найкращих галузевих практик і стандартів відповідності, включаючи правила захисту даних, щоб захистити конфіденційну інформацію та зменшити ризики.

Архітектура бази даних системи CRM буде продумано розроблена для задоволення різноманітних потреб вищих навчальних закладів у даних. PostgreSQL слугуватиме основою системи керування даними, забезпечуючи надійну та масштабовану основу для зберігання та запитів до величезних обсягів структурованих і неструктурованих даних. Крім того, система включатиме централізовану базу знань — сховище інституційної політики, передового досвіду та історичних даних — щоб надати користувачам цінні відомості та ресурси для прийняття обґрунтованих рішень. Завдяки ретельній розробці та управлінню базами даних система CRM забезпечить ефективний пошук даних, аналіз і звітність, дозволяючи установам отримувати корисну інформацію та стимулювати стратегічні ініціативи.

Користувальницький інтерфейс системи CRM надаватиме пріоритет простоті, інтуїтивності та доступності для користувачів із різним рівнем технічної підготовки. Чуйний веб-інтерфейс буде розроблено з використанням сучасних інтерфейсних технологій, таких як HTML5, CSS3 що забезпечує сумісність між пристроями та браузерами. Інтерфейс користувача системи міститиме інтуїтивно зрозумілі навігаційні меню, інтерактивні інформаційні панелі та персоналізовані перегляди, адаптовані до ролей і обов’язків різних груп користувачів. Крім того, система

запропонує широкі можливості налаштування, дозволяючи користувачам налаштовувати свій робочий простір, визначати параметри та ефективно отримувати доступ до відповідної інформації. Розставляючи пріоритети для взаємодії з користувачем та дизайну інтерфейсу, система CRM підвищить залучення користувачів, продуктивність і задоволення, сприяючи прийняттю та максимізації корисності системи в установі.

Розробка системи CRM відбуватиметься за систематичним і ітеративним підходом, який охоплюватиме такі ключові етапи, як збір вимог, аналіз, проектування, розробка, тестування, розгортання та обслуговування. Гнучкі методології, такі як Scrum або Kanban, будуть використовуватися для сприяння співпраці, гнучкості та постійного вдосконалення протягом життєвого циклу проекту. Крім того, процес розробки підтримуватиметься надійними методами управління проектами, системами контролю версій та інструментами співпраці для сприяння комунікації, прозорості та підзвітності між членами команди. Дотримуючись усталених методологій розробки та використовуючи сучасні інструменти та методи, система CRM буде розроблятися ефективно, ітеративно та з акцентом на надання цінності зацікавленим сторонам.

При обґрунтуванні запропонованих концептуальних рішень оцінювання проводитиметься на основі заздалегідь визначених критеріїв, таких як зручність використання, масштабованість, продуктивність, безпека, відповідність та узгодженість з інституційними цілями та вимогами. Вибір Java-розробки за допомогою Spring і PostgreSQL як базових технологій підтверджується їх перевіреним досвідом, широкою підтримкою спільноти та придатністю для програм корпоративного рівня. Розрахунки та моделювання можуть бути виконані для оцінки очікуваного впливу запропонованих рішень на ключові показники продуктивності, такі як задоволеність користувачів, надійність системи та інституційна ефективність. Загалом, запропоновані концептуальні рішення мають на меті

закласти міцну основу для розробки системи CRM, яка відповідає мінливим потребам і викликам закладів вищої освіти, надаючи їм можливість розвивати змістовні стосунки, рухати стратегічні ініціативи та досягати своєї місії та цілей.

РОЗДІЛ 2. ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ

2.1 Структура і характеристика системи

Відповідно до проведеного аналізу, концептуальні рішення для створення CRM системи, призначеної для закладів вищої освіти (ВНЗ), зосереджені на ефективному управлінні взаємодією студентів, викладачів, випускників та адміністрації. Функціональна структура цієї CRM охоплює різні завдання, включаючи зарахування, управління курсами, зв'язки з випускниками, та менеджмент розкладу зайнять. Ця структура представлена за допомогою нотації Уніфікованої мови моделювання (UML), зосереджуючись на діаграмах класів, діаграмах послідовності, діаграмах діяльності та діаграмах варіантів використання [8]

У розробці CRM-системи для вищих навчальних закладів на основі Java Spring і PostgreSQL концептуальні рішення відіграють вирішальну роль у формуванні функціональної структури ІУС.

Викладені вище концептуальні рішення створюють міцну основу для розробки CRM-системи вищих навчальних закладів. Завдяки використанню нотації UML і різних методів моделювання функціональна структура ІУС є чітко визначеною, що дозволяє ефективно обробляти інформацію та керувати нею відповідно до конкретних потреб освітнього контексту.

CRM-система для ВНЗ складатиметься з кількох взаємопов'язаних модулів, кожен з яких обслуговуватиме певні функції: Модуль управління студентами забезпечує зарахування студентів, реєстрацію, відстеження академічного прогресу та розкладу. Модуль управління курсами керує пропозиціями курсів, розкладом, призначеннями викладачів, оновленнями навчального плану та записом оцінок. Модуль академічної підтримки викладачів надає послуги академічної підтримки та професійної орієнтації.

Результати *моделювання поведінки* на рівні підсистеми можна подати, як:

- Діаграму прецеденту, яка також відома як діаграма варіантів

використання або діаграма пріоритету, — це візуальне представлення, яке ілюструє залежності або зв’язки між різними завданнями чи діями в проекті чи системі.[9] Діаграма варіантів використання ілюструє взаємодію між системою та її акторами, фіксуючи функціональні вимоги у формі типових взаємодій. [12]

У цьому випадку ми створимо діаграму прецедентів на основі акторів і прецедентів:

Актори:

- Адміністратори
- Студенти
- Викладачі
- Кури (припустимо, що це нетрадиційний актор для ілюстрації)

Прецеденти:

- Управління студентами
- Управління викладачами
- Відображення викладачів і курсів
- Відображення студентів і курсів

На цій діаграмі прямокутні рамки представляють завдання або дії (прецеденти). Стрілки вказують на залежності або зв’язки між цими завданнями. Актори (адміністратори, студенти, вчителі, курси) розташовуються навколо завдань на основі їх участі або залежності в кожному завданні. Наприклад: «Керування учнями», і «Керування вчителями» — це завдання, які безпосередньо виконуються адміністраторами. Участь студентів у «Відображенні вчителів і курсів» є нетрадиційним доповненням для ілюстрації, що свідчить про те, що це завдання може включати деякі несподівані елементи чи залежності.

Представимо ці елементи на діаграмі прецеденту:

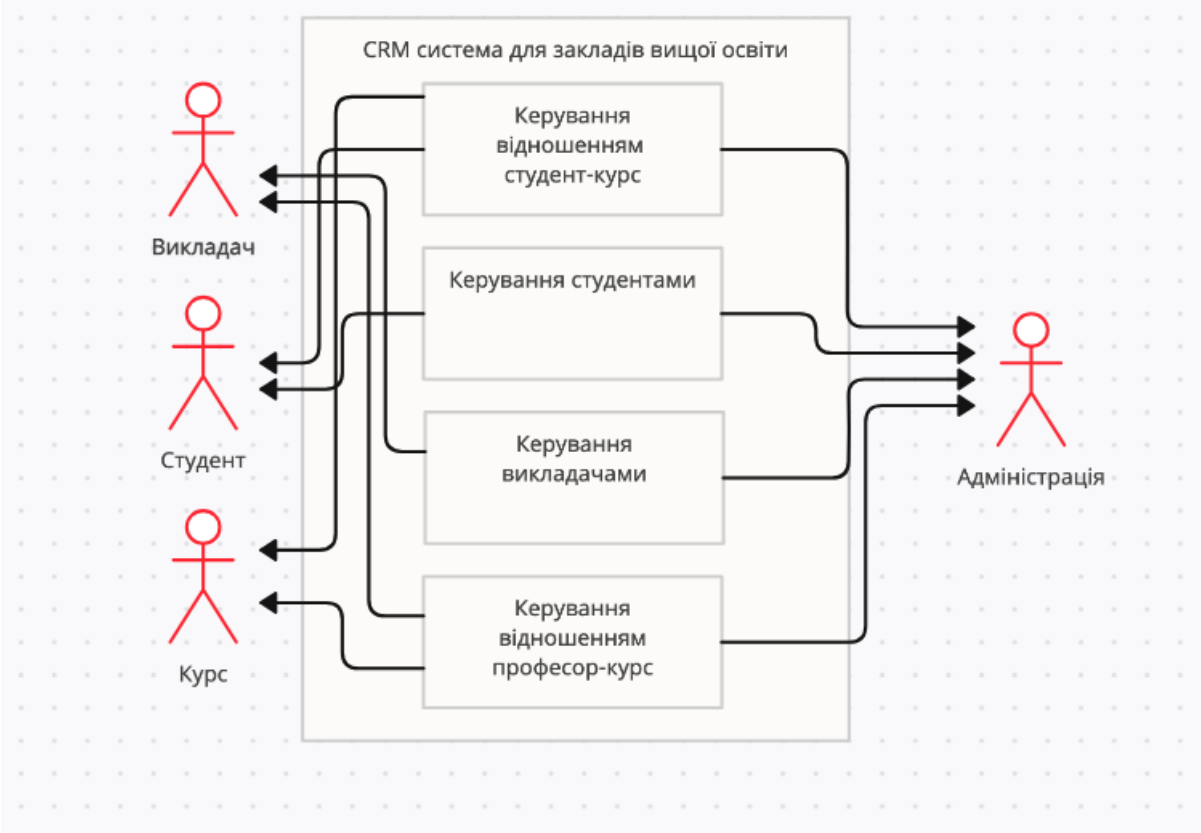


Рисунок 2.1 – Діаграма прецеденту

Джерело: розроблено автором

- Діаграма послідовності для вказаних сценаріїв використання [11] за участю адміністраторів, студентів, викладачів і курсів із відповідними прецедентами: керування студентами, керування викладачами, зіставлення викладачів і курсів, зіставлення студентів і курсів. Діаграми послідовності зображують взаємодію між об’єктами (компонентами) системи для кожного конкретного сценарію використання. Вони надають детальне уявлення про те, як об’єкти співпрацюють для досягнення певної функціональності. Діаграми послідовності охоплюють

різні сценарії, такі як зарахування студентів, реєстрація на курси, пожертвування випускників і співпраця викладачів.[13]

Взаємодія адміністратора:

Адміністратор запускає варіант використання «Керування студентами» > Система відображає інтерфейс управління студентами > Адміністратор вибирає параметри додавання, редагування або видалення студентів >

Система перевіряє введені дані та відповідно оновлює базу даних студентів.

Взаємодія студентів:

Студент взаємодіє з системою, щоб переглянути доступні курси > Система отримує інформацію про курс і відображає її студенту > Студент обирає потрібні курси та надсилає заявку на реєстрацію > Система перевіряє реєстрацію та оновлює записи про зарахування студентів.

Взаємодія вчителя:

Адміністратор ініціює сценарій використання «Керування вчителями» > Система представляє інтерфейс керування викладачем > Адміністратор виконує такі дії, як додавання, редагування або видалення викладачів > Система перевіряє введені дані та відповідно оновлює базу даних викладачів.

Відображення викладачів і курсів:

Адміністратор вибирає опцію відображення викладачів і курсів > Система отримує доступні курси та інформацію про викладачів > Адміністратор призначає викладачів за окремими курсами > Система перевіряє завдання та оновлює записи курсу.

Відображення студентів і курсів:

Адміністратор ініціює варіант використання для зіставлення студентів і курсів > Система отримує дані про студентів і курси >

Адміністратор розподіляє студентів на відповідні курси > Система перевіряє завдання та оновлює записи про зарахування студентів.

Представлення діаграми послідовності:

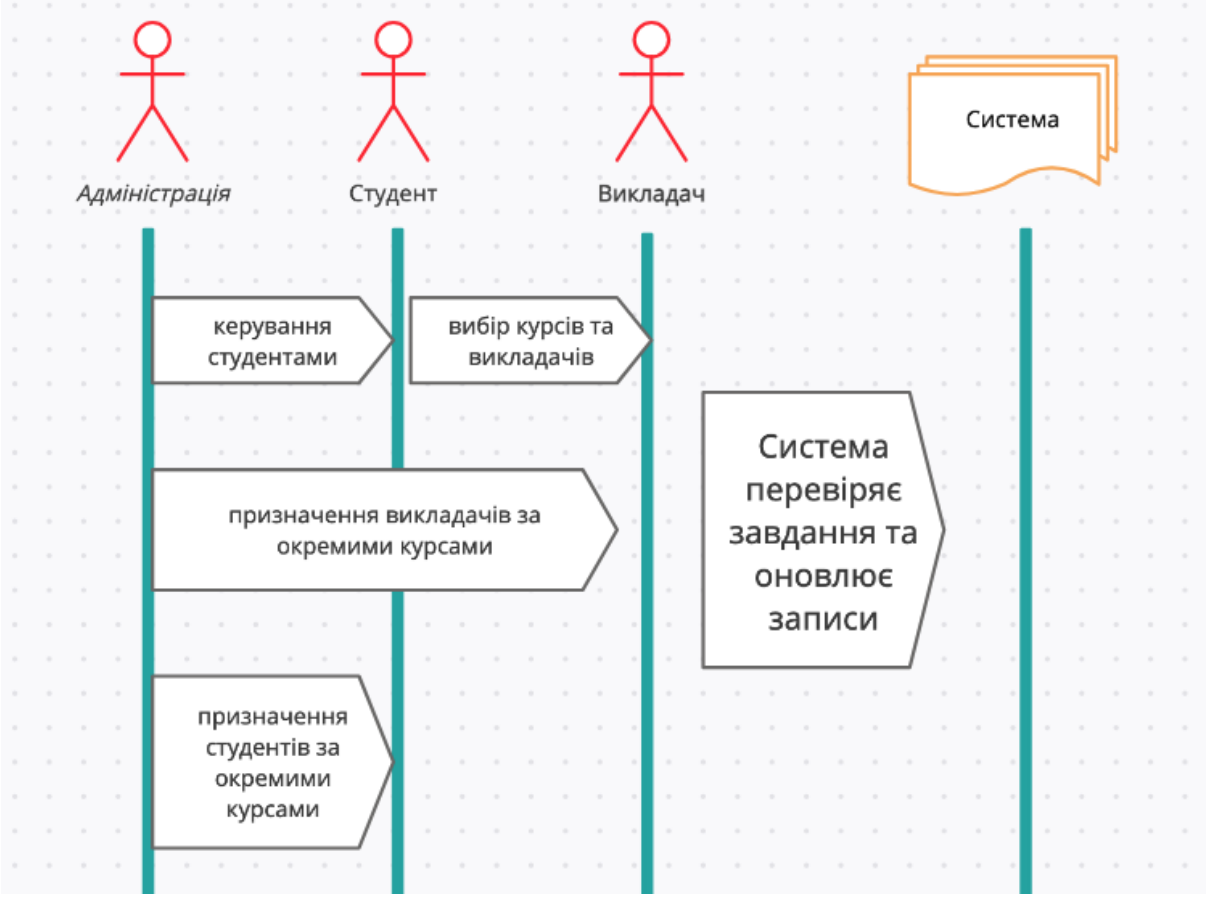


Рисунок 2.2 – Діаграма прецеденту

Джерело: розроблено автором

- Діаграми класів UML, що представляє статичну структуру підсистеми для системи CRM у вищих навчальних закладах. Діаграма містить пакети класів, зв’язки між ними, підмножину класів із зв’язками, а також атрибути та операції класу.[10] Також включає класи сутностей, граничні класи та класи керування згідно з наданими вказівками.

Пакети класів:

Entity: містить класи, що представляють сутності з довгим життєвим циклом.

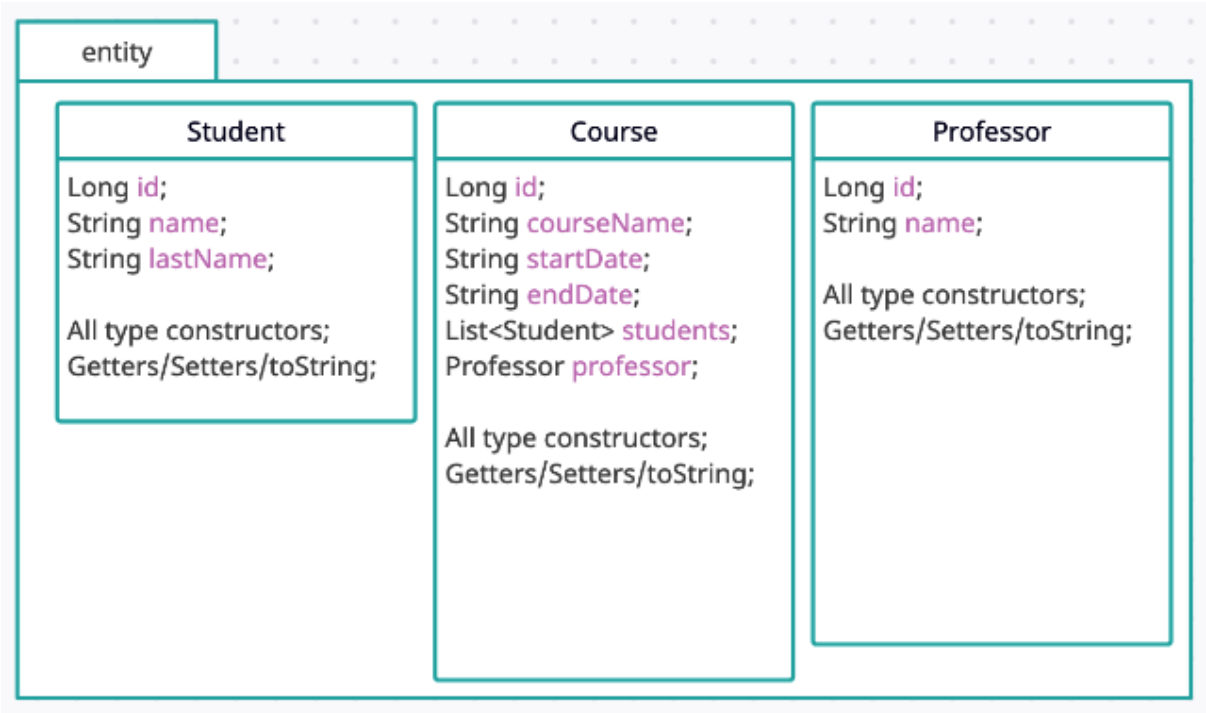


Рисунок 2.3 – Пакет entity

Джерело: розроблено автором

Repository: містить класи, що обробляють доступ до даних і зберігання в базах даних.

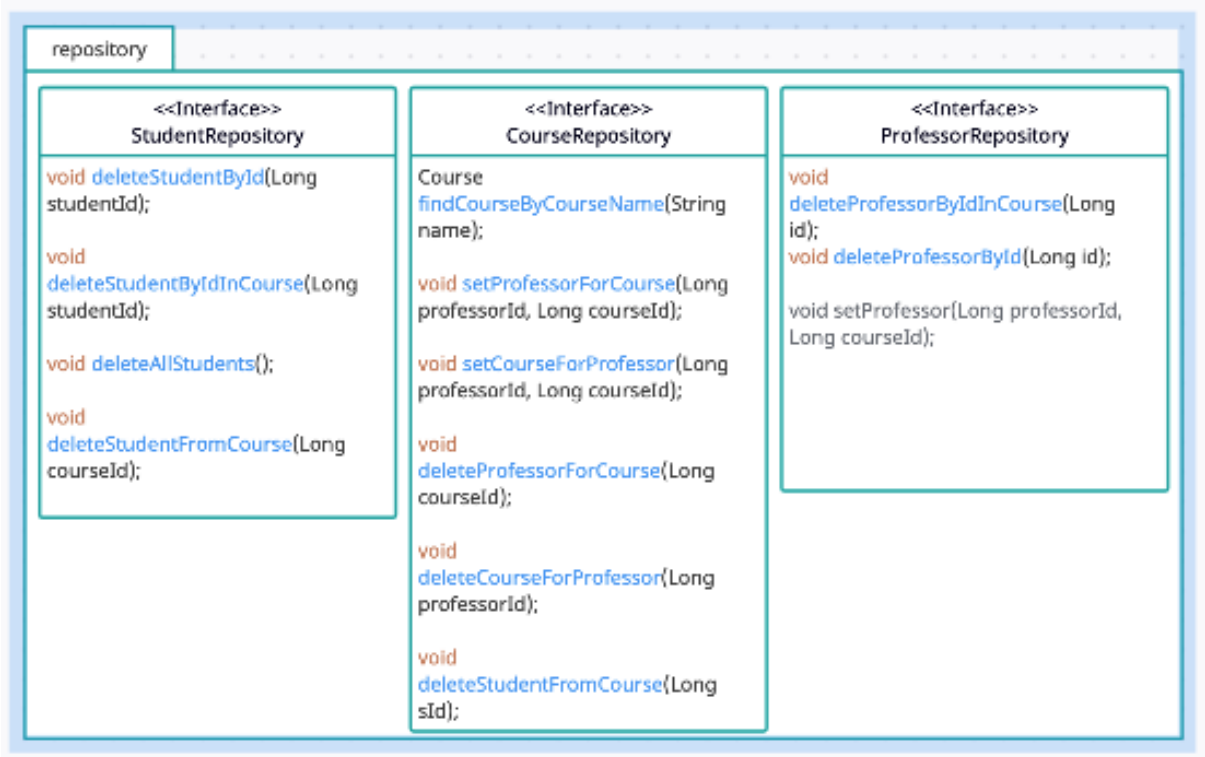


Рисунок 2.4 – Пакет reposirory

Джерело: розроблено автором

Controller: містить класи, відповідальні за обробку вхідних запитів і керування потоком.

Рисунок 2.5 – Пакет controller

Джерело: розроблено автором

Service: містить класи, що реалізують бізнес-логіку та координують операції.

Рисунок 2.6 – Пакет service

Джерело: розроблено автором

Config: містить класи для конфігурації та налаштування системи.

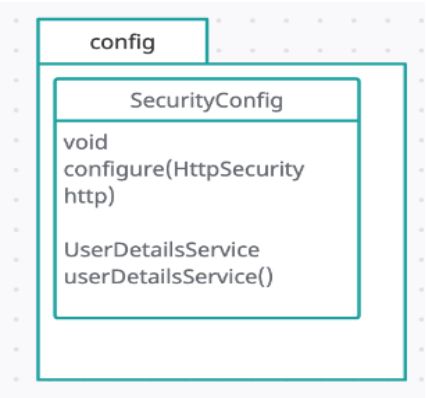


Рисунок 2.7 – Пакет config

Джерело: розроблено автором

Ці діаграми класів забезпечує структуроване представлення підсистем у системі CRM для вищих навчальних закладів. Вони включає пакети класів, зв’язки між ними, підмножину класів із зв’язками, а також атрибути та операції класу. Діаграма організована відповідно до вказаних пакетів класів, що забезпечує прозорість і зрозумілість моделі.

Викладені концептуальні рішення є схемою розробки CRM-системи, адаптованої до потреб вищих навчальних закладів. Завдяки використанню нотації UML і різних методів моделювання функціональна структура системи управління інформацією є чітко визначеною, що сприяє ефективній обробці інформації та управлінню різними завданнями та зацікавленими сторонами в освітньому контексті.

2.2 Методи та моделі в інформаційних управляючих системах і технологіях

Еволюція технічної, фінансової, кредитної та соціально-економічної діяльності тягне за собою все більш складні завдання, які часто характеризуються багатокритеріальними та багатоцільовими процесами

прийняття рішень.[14] Ці завдання вимагають одночасного розгляду різних цілей, часто суперечливих, і оцінки багатьох критеріїв при оцінці альтернатив. Сценарії прийняття рішень у сучасному контексті можуть варіюватися від слабко структурованих до зовсім неструктурованих, часто пов’язаних із недостатньою чи величезною кількістю даних, що ускладнює розпізнавання відповідної інформації. При вирішенні цих проблем використання відповідної математичної основи в рамках ІУС стає обов’язковим, зокрема за допомогою економіко-математичних методів і моделей, вбудованих у відповідні модулі, включаючи компоненти систем штучного інтелекту.[15]

Математичний апарат сучасної ІУС для структурованих задач включає такі кількісні методи, як лінійне, нелінійне та динамічне програмування, а також теорія ігор. Ці методи підходять для завдань, у яких зв’язки між елементами можна визначити кількісно. Для слабко структурованих проблем використовуються математичні моделі, що мають нечіткі набори та нечітку логіку, нейронні мережі та еволюційні алгоритми. Ці моделі враховують якісні та кількісні залежності всередині економічних і соціальних систем та їхнього зовнішнього середовища. Такі методи, як евристичні методи та підходи колективного штучного інтелекту, включаючи алгоритми бджіл, алгоритми мурашок, методи роїв частинок і штучні імунні системи, вони є цінними для навігації в невизначених і динамічних середовищах. Для неструктурованих задач інструментальними є методи та моделі штучного інтелекту, включаючи експертні системи та нейронечіткі моделі. Ці підходи, що керуються інтуїцією, логікою та експертними знаннями, добре підходять для обробки сценаріїв, де кількісні залежності невідомі або погано визначені. Крім того, процедури діалогу та системний аналіз відіграють ключову роль у розумінні та вирішенні складних, неструктурованих проблем.[16]

У контексті розробки системи CRM для закладів вищої освіти ці математичні методи та моделі відіграють вирішальну роль у покращенні процесів прийняття рішень та оптимізації продуктивності системи. Наприклад, кількісні методи допомагають оптимізувати розподіл ресурсів і планування курсу. Нечітка логіка та нейронні мережі можуть сприяти персоналізованій підтримці студентів та прогнозній аналітиці для успіху студентів. Технології штучного інтелекту дозволяють використовувати ефективні стратегії залучення випускників і кампанії зі збору коштів. Експертні системи допомагають у вирішенні складних адміністративних питань та оптимізації служб академічної підтримки.

Використовуючи ці математичні підходи, системи CRM для закладів вищої освіти можуть ефективно вирішувати різноманітні та постійно зростаючі проблеми в управлінні взаємодією студентів, викладачів, випускників та адміністраторів, зрештою підвищуючи загальну інституційну ефективність та ефективність.

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

3.1. Проєктування бази даних та/або сховища даних для інформаційної управляючої системи

Якщо говорити про розробку інфологічної (логічної) моделі бази даних, то процес проектування інформації для системи CRM передбачає ідентифікацію та визначення об’єктів та їхніх зв’язків у межах предметної області. Ці об’єкти представляють основні елементи даних, які мають зберігатися в базі даних, і структуровані відповідно до вимог користувача та програми. Дизайн повинен відповідати принципам нормалізації, що забезпечити цілісність даних і мінімізувати надмірність.

Сутності, визначені для системи CRM:

1. Student з атрибутами: studentID (первинний ключ), firstName, lastName.
2. Course з атрибутами: courseID (первинний ключ), courseName, start_date, end_date, professor_id (зовнішній ключ).
3. Professor з атрибутами: professorID (первинний ключ), name.
4. Course_students з атрибутами: courseID (зовнішній ключ), studentID (зовнішній ключ)

Студенти можуть записатися на кілька курсів, і кожен курс може мати кілька студентів (відношення «Many-to-Many»), що сприяє об’єкту реєстрації. Курсами керують викладачі, встановлюючи зв’язок «One-to-Many» між предметом і суб’єктами курсу.

Кожна реєстрація пов’язана з конкретним студентом і курсом, ідентифікованими відповідними ідентифікаторами.

Інфологічна модель:

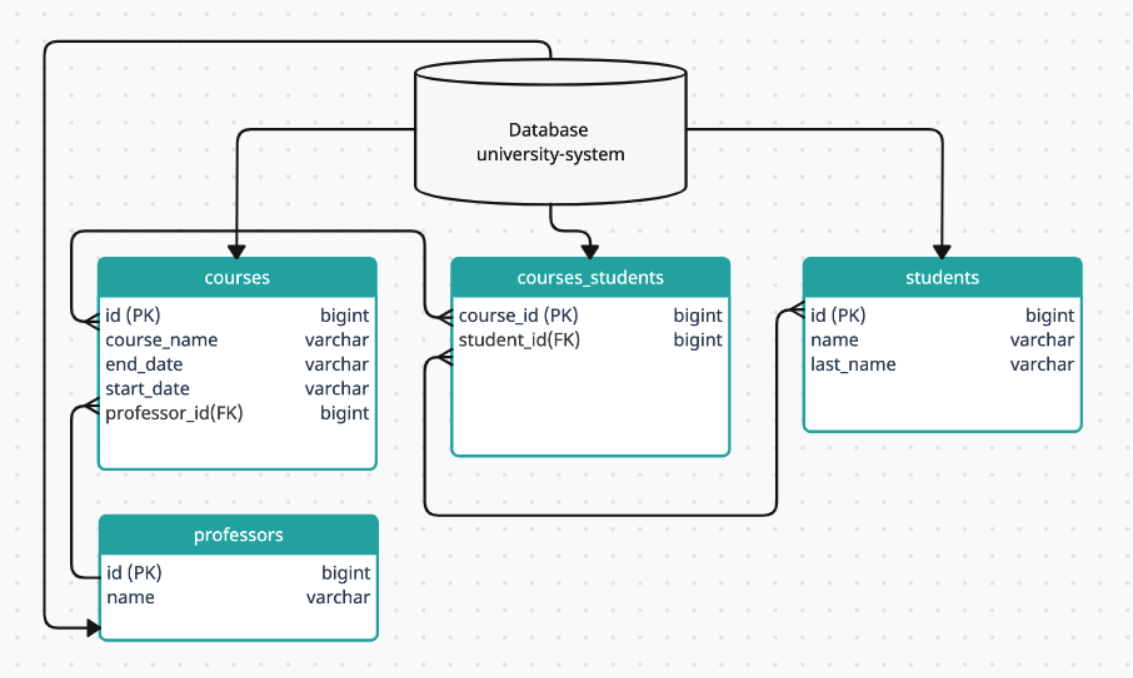


Рисунок 3.1 - Інфологічна модель БД

Джерело: розроблено автором

Інфологічна модель ілюструє сутності «Студент», «Курс», «Професор» а також їхні відповідні атрибути. Зображуються зв’язки між об’єктами, включаючи зв’язки «багато-до-багатьох» між студентом і курсом, а також зв’язки «один-до-багатьох» між професором і курсом. Для системи CRM PostgreSQL обрано як СУБД через її надійність і придатність для обробки складних структур даних. Підтримка PostgreSQL розширених функцій, таких як тип даних JSON і механізми керування паралелізмом, добре відповідають вимогам системи. Дизайн включає створення таблиць із відповідними типами даних, первинними та зовнішніми ключами та індексами для оптимізації продуктивності.

Контрольний приклад набору даних завантажується в базу даних, що представляє набір зразків студентів, курсів, професорів і записів про зарахування.

За допомогою ERwin або подібних інструментів прогнозований обсяг бази даних обчислюється на основі таких характеристик, як довжина запису, початковий розмір таблиці, розмір індексу, максимальна кількість записів і щомісячне зростання запису.[17] Прогнозування виконується за певний часовий горизонт, щоб оцінити розмір бази даних у часі. Розрахунок прогнозованого обсягу бази даних є важливим для планування потужностей, розподілу ресурсів і забезпечення оптимальної продуктивності системи. Прогнозуючи зростання бази даних з часом, адміністратори можуть передбачати майбутні потреби в ресурсах і планувати масштабованість.

Характеристики, які розглядаються для прогнозування:

- середній розмір окремого запису в кожній таблиці, включаючи розмір атрибутів і будь-які додаткові витрати.
- початковий розмір кожної таблиці в базі даних, включаючи дані та сховище індексів.
- розмір індексів, створених у таблицях для полегшення ефективного пошуку даних.
- максимальна кількість записів, які очікується зберігати в кожній таблиці на основі системних вимог і моделей використання.
- швидкість, з якою очікується збільшення кількості записів у кожній таблиці з часом. [18]

На основі розрахованих темпів зростання та прогнозованих тенденцій визначаються прогнозні обсяги таблиць бази даних для кожного місяця на заданому часовому горизонті.

Ці прогнозовані обсяги дають цінну інформацію про очікувані потреби в ресурсах і міркування про продуктивність системи CRM.

Проектування та прогнозування бази даних для системи CRM є критичними компонентами процесу розробки системи. Ретельно

розробляючи інфологічну (логічну) і фізичну моделі бази даних і прогнозуючи її майбутні обсяги, зацікавлені сторони можуть приймати обґрунтовані рішення щодо розподілу ресурсів, масштабованості та оптимізації продуктивності. Контрольний приклад і розраховані прогнозні обсяги є цінними інструментами для забезпечення ефективної роботи та довгострокової життєздатності CRM-системи вищих навчальних закладів.

3.2. Проєктування бази знань інформаційної управляючої системи та/або засоби інтелектуального аналізу даних

У контексті розробки CRM-системи для вищих навчальних закладів впровадження інтелектуального аналізу великих даних відіграє вирішальну роль у наповненні бази знань. Це передбачає використання різних технологій, методів і алгоритмів для отримання цінної інформації з великих обсягів даних. Для ефективного використання під час обробки великих даних у сховищі керування знаннями рекомендовано використовувати певні методи й алгоритми.

Процеси «Видобуток, перетворення, завантаження» (ETL)[19] і «Вилучення, завантаження, перетворення» (ELT)[20] використовуються для ефективного вилучення даних із багатьох джерел, перетворення їх у узгоджений формат і завантаження в база знань.

Такі методи, як аналіз головних компонентів (PCA)[21] або декомпозиція сингулярного значення (SVD)[22], використовуються для зменшення розмірності даних великої розмірності, зберігаючи важливу інформацію.

Методи регулярізації даних гарантують, що дані масштабуються належним чином і відповідають стандартизованому формату,

забезпечуючи кращу продуктивність алгоритмів машинного навчання. Методи візуалізації використовуються для представлення складних шаблонів даних у форматі, який можна візуально інтерпретувати, допомагаючи розуміти та приймати рішення. Ієрархічні та неієрархічні методи кластерного аналізу використовуються для ідентифікації природних груп або кластерів у даних.[23] Дерева рішень створюються для розуміння базової бізнес-логіки, тоді як мережі довіри будуються для встановлення довірчих відносин між об'єктами в даних.[25] Такі алгоритми, як Apriori, використовуються для виявлення цікавих асоціацій або шаблонів у даних, які можуть бути цінними для прийняття рішень і систем рекомендацій.[26] Наївний класифікатор Байєса застосовується для напівструктурованих даних, тоді як структуровані дані класифікуються за допомогою послідовних підходів, таких як k-найближчі сусіди і опорні векторні машини (SVM).[26] Неглибокі ШНМ навчаються на структурованих даних для завдань прогнозного моделювання, фіксуючи складні зв'язки між вхідними характеристиками та цільовими змінними.[27]

Для реалізації цих методів і алгоритмів рекомендуються різні програмні засоби. Інструменти з відкритим кодом і пробні версії, такі як RapidMiner, Orange DataMiner, Weka, KNIME, Rattle і WizWhy, забезпечують зручний інтерфейс і підтримують широкий спектр завдань обробки й аналізу даних. Додатково можна використовувати стохастичне математичне програмування з використанням генетичних алгоритмів (наприклад, Evolver) для оптимізації конкретних проблем.[29]

Пристрої нечіткої логіки (наприклад, CubiCalc) можна використовувати для моделювання та симуляції невизначеності знань, підвищуючи здатність системи справлятися з невизначеністю.[28]

Реалізація завдань інтелектуального аналізу великих даних відповідає принципам управління проектами та дотримується міжгалузевого стандарту процесу інтелектуального аналізу даних (CRISP-DM). Цей стандартизований підхід забезпечує систематичне виконання та документування процесу аналізу даних.

Проектування та впровадження бази знань у CRM-системі передбачає декілька послідовних етапів, що забезпечують ефективне використання інформаційних ресурсів.

Так от, завдання полягає в тому, щоб розробити систему бази даних для управління набором студентів, інформацією про курси, відомостями про професора та їхніми асоціаціями у вищому навчальному закладі. Основною метою є створення надійної схеми бази даних, яка ефективно зберігає та отримує відповідні дані для полегшення адміністративних завдань, пов’язаних із керуванням курсом студентів.

База даних слугуватиме централізованим сховищем для зберігання інформації про студентів, курси, викладачів та їхні асоціації. Вона підтримуватиме такі операції, як реєстрація, керування курсами та призначення професорів. Була визначена структура таблиць для зберігання інформації про студента, курс, професора та запис. Встановлені зв’язки між таблицями для відображення зв’язків між студентами, курсами та викладачами. Задані обмеження, такі як первинні ключі, зовнішні ключі та посилальна цілісність, щоб підтримувати цілісність даних. Створені індекси для оптимізації продуктивності запитів для даних, до яких часто звертаються. Нормалізована схема бази даних, щоб мінімізувати надмірність і покращити узгодженість даних.

Вхідні дані включають відомості про студента, інформацію про курс і дані професора. Вихідні дані включають записи про реєстрацію, розклад курсів і призначення викладачів. Прикладом вирішення проблем

є впровадження методів нормалізації для усунення надмірності даних, створення відповідних індексів для покращення продуктивності запитів, прийняття обмежень посилальної цілісності для підтримки узгодженості даних. Критеріями оцінки якості рішення є цілісність даних, продуктивність запитів, масштабованість, простота обслуговування, дотримання принципів нормалізації та сумісність із галузевими стандартами.

Цей етап ідентифікації бази даних забезпечує комплексний план розробки та впровадження системи бази даних для керування набором студентів, інформацією про курси, відомостями про викладачів та їх асоціаціями у вищому навчальному закладі. Він закладає основу для наступних етапів розробки бази даних і забезпечує узгодження з цілями проекту та обмеженнями ресурсів.

Етап концептуалізації бази даних передбачає ретельний аналіз проблемної області, зосереджуючись на розумінні вимог і відносин у сфері управління студентом і курсом у вищих навчальних закладах.

БД має певну ієрархію, на курси можуть бути записані кілька студентів (багато-до-багатьох), а професори призначаються для викладання окремих курсів (один-до-багатьох). Зміни в наборі студентів впливають на можливості курсу та призначення викладачів. Студенти беруть участь у процесі реєстрації, який є складовою управління курсом.

Рівень деталізації включає розуміння атрибутів і характеристик кожної сутності (наприклад, studentID, courseName) і їхніх зв’язків у схемі бази даних. Вхідні дані включають дані про зарахування студентів, розклад курсів і призначення викладачів. Вихідні дані складаються з записів про реєстрацію, розкладу курсів і призначень професора.[30]

До використаніх стратегій та гіпотез належать нормалізвання схеми бази даних, щоб усунути надмірність і забезпечити узгодженість

даних. Обмеження посилальної цілісності для підтримки цілісності даних. Оптимізація продуктивність запитів за допомогою індексування та налаштування запитів. До типів обмежень, що накладаються на процеси можна віднести дотримання принципів нормалізації для забезпечення узгодженості даних і мінімізації дублювання даних. Відповідність обмеженням посилальної цілісності для підтримки цілісності даних.

Цей етап концептуалізації закладає основу для розробки схеми бази даних, яка ефективно фіксує зв’язки та вимоги системи управління студентом і курсом у вищих навчальних закладах. Завдяки комплексному аналізу концепцій, взаємозв’язків і методів вирішення проблем зацікавлені сторони можуть з упевненістю переходити до наступних етапів проектування та впровадження бази даних.

На етапі формалізації бази даних, знання, отримані на попередніх етапах, формалізуються та структуруються в дизайн бази даних. Це передбачає визначення методів представлення та інтерпретації знань, забезпечення ясності та узгодженості у представленні понять і зв'язків. Відбувається формалізація основних понять і зв'язків за допомогою формальної мови, наприклад SQL для визначення схеми бази даних. Визначаються структури бази даних, включаючи таблиці, стовпці та зв'язки між сутностями. Вибір відповідних інструментів розробки програмного забезпечення для реалізації бази даних, таких як PostgreSQL для управління реляційною базою даних є не менш важливим. Обов’язковою є оцінка адекватності дизайну бази даних для досягнення цілей системи управління інформацією та забезпечення повноти представлення понять і зв'язків та моделювання роботи системи бази даних, включаючи визначення механізмів логічного висновку для пошуку та маніпулювання даними.

На стадії впровадження бази даних реалізується формалізований дизайн бази даних на основі знань, отриманих в результаті інтелектуального аналізу даних. Основні кроки включають створення таблиць бази даних, індексів і обмежень відповідно до формалізованої схеми; наповнення бази даних даними, отриманими на етапі аналізу; перевірка організації та структури бази даних для забезпечення ефективного функціонування інформаційної системи управління; розробка одного або кількох прототипів системи бази даних для перевірки її дизайну та функціональності.[31]

Під час тестування компетентність і функціональність системи бази даних оцінюються за допомогою серії типових завдань. Це передбачає інтерактивне тестування системи баз даних з використанням різних сценаріїв і тестів; перевірка здатності системи ефективно обробляти типові та крайні випадки; продовження ітерацій тестування, доки система не досягне необхідного рівня компетентності, визначеного експертами та зацікавленими сторонами.

Стадія дослідної експлуатації включає оцінку придатність системи бази даних для кінцевих користувачів оцінюється в реальному середовищі.

Протягом цих етапів процес розробки та впровадження бази даних забезпечує ефективну розробку та розгортання системи управління інформацією для вищих навчальних закладів, як зазначено в наданій схемі бази даних.[32]

3.3. Програмне забезпечення

3.3.1 Проєктування програмного забезпечення

Обрана методологія має бути гнучкою, щоб відповідати мінливим вимогам і ітераційній розробці. Гнучкі методології, такі як Scrum або Kanban, можуть підійти для цього проекту завдяки своїй гнучкості та адаптивності.

Java обрано як основну мову програмування через її універсальність, незалежність від платформи та великі бібліотеки.

Середовище розробки включатиме такі інструменти, як IntelliJ IDEA, що надаватиме функції для ефективного кодування, налагодження та тестування.

Обрано саме використання Spring для створення програми, оскільки він пропонує надійні функції для швидкої розробки, впровадження залежностей і легкої інтеграції з іншими технологіями.

PostgreSQL вибрано як систему керування реляційною базою даних для зберігання та керування даними програми через її надійність, продуктивність і підтримку складних запитів.

Використання Hibernate ORM для взаємодії з базою даних, щоб абстрагуватися від низькорівневих операцій SQL і забезпечити можливості об’єктно-реляційного відображення (ORM).

В проекті також використовується Thymeleaf як механізм створення шаблонів для відтворення HTML-сторінок на стороні сервера, що забезпечує повну інтеграцію з Spring MVC. Maven використовується як інструменту автоматизації збірки для керування залежностями та спрощення процесу збирання.

Прийняття шаблону MVC для структурування архітектури програми полегшить розділення проблем, коли модель обробляє дані, представлення керує рівнем презентації, а контролер керує взаємодією користувачів і бізнес-логікою.[35]

Розгляд шаблону MVVM для розробки веб-інтерфейсів є цікавим рішенням, особливо якщо є потреба в реактивних і динамічних елементах інтерфейсу користувача. Цей шаблон можна реалізувати за допомогою сучасних фреймворків JavaScript, таких як Angular, React або Vue.js, покращуючи розділення проблем між даними, інтерфейсом користувача та логікою.[34]

Основні компоненти системи CRM включають такі сутності, як студент, курс, професор, а також їхні відповідні контролери та служби. Крім того, будуть представлені інтерфейси користувача та сховища для взаємодії з базою даних.

Програмний продукт можна розгортати на хмарних платформах, таких як AWS, Azure або Google Cloud, або на локальних серверах залежно від вимог і наявності інфраструктури закладу.[33]

Дотримуючись цих принципів розробки програмного забезпечення та використовуючи відповідні технології та шаблони, систему CRM для вищих навчальних закладів можна розробити ефективно, забезпечуючи масштабованість, зручність обслуговування та надійність програмного продукту.

3.3.2 Вибір архітектури програмного забезпечення

Підходячи до вибору архітектури програмного забезпечення для системи CRM для вищих навчальних закладів, потрібно визначити, найбільше підходить монолітна чи розподілена архітектура. Це рішення є

критичним і має бути узгоджене з ідентифікацією користувачів системи та впровадженням заходів корпоративної безпеки.

Монолітна архітектура передбачатиме побудову всієї системи CRM як єдиного інтегрованого блоку. Це може бути доцільним, якщо функціональні можливості системи є відносно простими, і немає найближчих планів щодо широкої масштабованості або незалежного розгортання компонентів.[36] Враховуючи всі фактори керування даними студента, викладача та курсу, а також потребу в масштабованості та гнучкості в середовищі вищої освіти, монолітна архітектура може бути більш підходящою.

З іншого боку, розподілена архітектура передбачає розбиття системи на менші взаємопов’язані служби. Це забезпечує модульну розробку, незалежне масштабування компонентів і кращу ізоляцію несправностей.[37]

Оскільки JWT використовується для автентифікації користувачів у проекті, дуже важливо узгодити архітектуру з цим вибором. У монолітній архітектурі, де сервіс самостійно обробляє автентифікацію, токени JWT використовуються частіше через їх характер без збереження стану та легкість масштабованості. Це контрастує з програмами, де автентифікація на основі сеансу за допомогою файлів cookie може бути більш поширеною. JWT також можна використовувати для авторизації та профілювання. Кодуючи ролі користувачів і дозволи в маркерах JWT, система може ефективно застосовувати політики контролю доступу в розподілених службах.

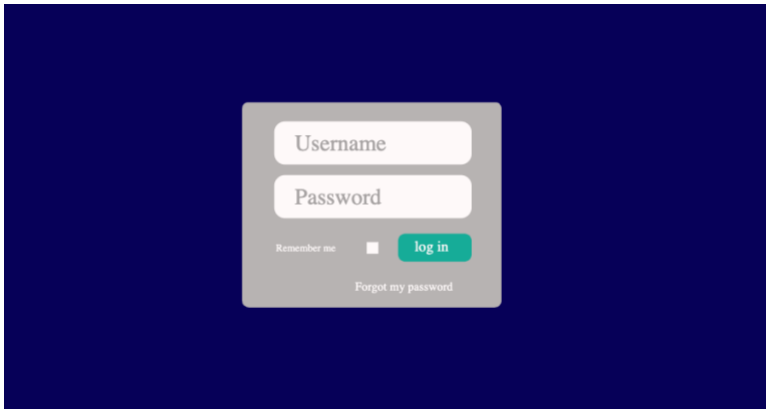


Рисунок 3.2 – Прототип форми для авторизації користувачів

Джерело: розроблено автором

При реалізації авторизації з трьома ролями (студент, професор і адміністратор) принцип роботи передбачає наступні кроки:

Визначення ролі (студент, професор, адміністратор) у конфігурації безпеки програми; Налаштування правила керування доступом на основі ролей, щоб обмежити доступ до певних кінцевих точок або ресурсів на основі ролей, призначених користувачам. Коли користувач входить або запитує доступ до захищеного ресурсу, його облікові дані (наприклад, ім’я користувача та пароль) перевіряються. Після успішної автентифікації створюється JWT, що містить інформацію та ролі користувача. JWT підписується за допомогою секретного ключа, відомого лише серверу, що забезпечує його цілісність і запобігає втручанню. Після успішної автентифікації сервер видає маркер JWT, що містить ідентифікаційні дані користувача (наприклад, ім’я користувача) і ролі (наприклад, студент, професор, адміністратор). Маркер JWT повертається клієнту як частина відповіді автентифікації. Для наступних запитів до захищених ресурсів клієнт включає маркер JWT у заголовки запиту.

Сервер перевіряє підпис маркера JWT і декодує його вміст, щоб отримати ролі користувача.

Додаток В

```java
@Monitor
@RestController
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
@RequestMapping("/api/v1/auth")
@CrossOrigin(origins = "http://localhost:3000")
public class AuthController {

    AuthenticationService authenticationService;

    @PostMapping("/authenticate")
    public ResponseEntity<AuthenticationResponse>
authenticate(@RequestBody AuthenticationRequest request) {
        return
ResponseEntity.ok(authenticationService.authenticate(request));
    }

    @PostMapping("/logout")
    public ResponseEntity<?> logout(@RequestHeader("Authorization")
String token) {
        authenticationService.logout(token);
        return ResponseEntity.ok().build();
    }
}
```

```java
@Service
@Slf4j
@RequiredArgsConstructor
public class AuthenticationService {

    private final AccountService service;
    private final TokenService tokenService;
    private final JwtService jwtService;
    private final AuthenticationManager authenticationManager;
```

Рисунок 3.3 – Фрагмент додатку В. Реалізація JWT токена

Джерело: розроблено автором

Незалежно від вибраної архітектури впровадження TLS (раніше SSL) для захисту зв’язку між клієнтами та серверами має важливе значення. TLS гарантує, що дані, що передаються через мережу, зашифровані, зменшуючи ризик прослуховування або підробки.

Крім того, шифрування конфіденційних даних, що зберігаються в базі даних, незалежно від архітектури, підвищує безпеку. Це гарантує, що

навіть якщо дані скомпрометовані, вони залишаться незрозумілими для неавторизованих сторін.

Використання JWT для автентифікації, ідентифікації та авторизації відповідає сучасним найкращим практикам безпеки. Токени JWT забезпечують ідентифікацію користувача та права доступу, сприяючи безпечній автентифікації та авторизації в розподілених службах. Хоча JWT обрано для цього проекту, інші методи автентифікації, такі як ключі SSH або протоколи SSL/TLS, також можуть бути актуальними, залежно від конкретних вимог безпеки та обмежень.[38]

В додатку передбачено, що кожна роль користувача – викладачі, студенти та менеджери – може ефективно виконувати свої завдання та обов'язки в академічному середовищі установи. Функціональні можливості, призначені для викладачів у додатку системи CRM:

- викладачі можуть створювати нові курси в системі CRM, вказуючи такі деталі, як назва курсу, опис, попередні умови та кредитні години.
- вони можуть завантажувати та керувати навчальними планами курсу, окреслюючи цілі курсу, розклад, обов'язкову літературу, завдання та критерії оцінювання.
- професори можуть налаштовувати розклад занять, включаючи час лекцій, місця проведення та будь-які повторювані події, такі як лабораторні роботи чи дискусії.
- вони можуть створювати та розповсюджувати завдання в електронному вигляді, вказуючи дати виконання, інструкції та правила подання.
- професори вводять і оновлюють оцінки студентів за завдання, іспити, тести та інші види оцінювання в системі CRM.

- система CRM може мати функції для автоматичного розрахунку загальних оцінок за курс на основі зважених категорій або спеціальних формул.
- професори можуть публікувати оголошення, щоб інформувати студентів про важливі оновлення курсу, нагадування або зміни в розкладі.

Ці функції дають змогу викладачам ефективно керувати своїми курсами, взаємодіяти зі студентами, оцінювати прогрес студентів і брати участь у науково-дослідній діяльності в системі CRM навчального закладу.

Функціональні можливості, розроблені для студентів у системі CRM:

- студенти можуть шукати доступні курси за такими критеріями, як назва курсу, відділ, викладач або розклад.
- щойно вони знайдуть потрібні курси, студенти можуть зареєструватися на них безпосередньо через систему CRM, якщо немає обмежень щодо реєстрації.
- якщо курс заповнений, студенти можуть приєднатися до списку очікування та отримувати сповіщення, якщо звільниться місце.
- студенти можуть подавати завдання в електронному вигляді через спеціальний портал подання завдань системи CRM.
- вони можуть завантажувати файли, документи або мультимедійні матеріали відповідно до інструкцій із завдання.
- після успішного подання студенти отримують підтвердження про подання разом із міткою часу.
- студенти можуть переглядати свої оцінки за окремі завдання, іспити, тести та загальні оцінки за курс за допомогою функції журналу оцінок системи CRM.

- вони можуть аналізувати свою успішність, визначати сфери, які потрібно покращити, і відстежувати свій прогрес протягом семестру.
- студенти можуть отримати доступ до матеріалів курсу, таких як слайди лекцій, матеріали для читання, відео та додаткові ресурси, завантажені викладачами в систему CRM.
- інтеграція з бібліотечною системою закладу може дозволити студентам отримувати доступ до онлайн-журналів, баз даних та інших академічних ресурсів безпосередньо з платформи CRM.

Ці функції дають змогу студентам ефективно керувати своєю академічною подорожжю, від реєстрації на курс до подання завдань, відстеження оцінок, спілкування з викладачами та доступу до ресурсів курсу, і все це в системі CRM навчального закладу.

Та функціональні можливості, для менеджерів або адміністраторів у програмі системи CRM:

- менеджери мають повноваження створювати, змінювати та деактивувати облікові записи викладачів, студентів та інших адміністраторів у системі CRM.
- вони призначають конкретні ролі та дозволи для кожного облікового запису користувача на основі їхніх обов’язків і потреб у доступі.
- менеджери можуть контролювати рівні доступу до конфіденційних даних і системних функцій, забезпечуючи відповідність політикам і нормам безпеки.
- менеджери контролюють точність і цілісність даних, що зберігаються в системі CRM, з можливістю вводити, редагувати або видаляти записи за потреби.

- менеджери вживають заходів безпеки даних, таких як шифрування, контроль доступу та регулярне резервне копіювання, щоб захистити конфіденційну інформацію, що зберігається в системі CRM.
- вони аналізують дані звітів, щоб виявити закономірності, тенденції та ідеї, які дають змогу інформувати ініціативи щодо прийняття рішень та стратегічного планування.
- вони контролюють інтеграцію системи CRM з іншими інституційними системами, такими як інформаційні системи для студентів, системи управління навчанням або фінансові системи, забезпечуючи безперебійний потік даних і взаємодію.

Ці функції дають менеджерам можливість контролювати ефективну роботу, підтримку та оптимізацію системи CRM, підтримуючи інституційні цілі та завдання в академічних, адміністративних і стратегічних областях.

Таблиця 3.1 – Порівняльна характеристика можливостей користувачів, в залежності від ролі.

Функціональність	Студенти	Викладачі	Менеджери/Адміністратори
Управління курсом	-Запис на курс - Здача завдань	- Створення курсу -Керування навчальним планом -Керування розкладом - Створення завдань	- Створення курсу - Управління програмою - Управління розкладом

Продовження таблиці 3.1			
Керування оцінками	- Перегляд оцінок -Сповіщення про оцінки	- Введення оцінок - Розрахунок оцінок - Розподіл оцінок	- Перевірка даних - Моніторинг оцінок - Звітування
Комунікацій	-Обмін повідомленнями -Дискусійні форуми -Перегляд оголошень	- Розміщення оголошень - Повідомлення - Спрощення обговорення	- Підтримка користувачів - Збір відгуків - Застосування політики
Керування користувачами			-Керування обліковими записами користувачів - Контроль доступу
Керування даними			- Введення та редагування даних - Перевірка даних

Джерело: сформовано автором на основі виконаного дослідження

Обґрунтовуючи вибір архітектури, методів автентифікації та заходів безпеки, CRM-система для вищих навчальних закладів розроблена для ефективного задоволення як функціональних потреб, так і потреб безпеки.

При розробці системи CRM для закладів вищої освіти вкрай важливо вибрати архітектуру, методи автентифікації та заходи безпеки, які ефективно відповідають як функціональним вимогам, так і проблемам безпеки. Беручи до уваги порівняльні характеристики можливостей користувачів для різних ролей, наприклад студентів, викладачів і менеджерів/адміністраторів, вибрана архітектура системи має сприяти безперебійному доступу до функціональних можливостей, одночасно забезпечуючи надійні заходи безпеки для захисту конфіденційних даних.

3.3.3 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем

Для розробки серверної частини CRM-системи для ВНЗ та враховуючи вимоги проекту щодо масштабованості, гнучкості та швидкості, було обрано такий стек технологій:

Java — це універсальна та широко використовувана мова програмування, відома своєю надійністю, масштабованістю та великою екосистемою бібліотек і фреймворків.[39] Spring, популярний фреймворк Java, пропонує такі функції, як впровадження залежностей, аспектно-орієнтоване програмування та легку інтеграцію з базами даних, що робить його придатним для розробки програм корпоративного рівня, [40] таких як система CRM.

Щоб взаємодіяти з базою даних, керувати з'єднаннями та перетворювати запити в об'єкти, що використовуються в бізнес-логіці, використовується

ORM. Spring бездоганно інтегрується з фреймворками ORM, такими як Hibernate, що автоматизує відображення об'єктів Java у таблиці реляційної бази даних, спрощуючи взаємодію з базою даних і зменшуючи шаблонний код. [41]

Для ефективного зберігання та доступу до реляційних даних можна розглянути PostgreSQL, будучи надійною РБД з відкритим кодом, добре підходить для керування структурованими даними в системі CRM. Його підтримка складних запитів, цілісності даних і транзакцій ACID робить його надійним вибором.[42]

Для розгортання додатку на локальному сервері використовується Apache Tomcat — це веб-сервер із відкритим вихідним кодом і контейнер сервлетів, який можна використовувати для розгортання веб-додатків на

основі Java, у тому числі створених за допомогою Spring. Він забезпечує легке та ефективне середовище виконання для розміщення веб-служб і сервлетів.[43]

Для керування залежностями проекту та створення програми використовується Maven — це потужний інструмент автоматизації збірки для проектів Java, який спрощує процес збирання, керує залежностями проекту та створює проектну документацію. Це спрощує робочий процес розробки та забезпечує послідовність проекту.[44]

Щоб полегшити передачу даних між компонентами клієнта та сервера, можна використовувати такі генератори шаблонів, як Thymeleaf — це сучасний механізм шаблонів Java на стороні сервера, який бездоганно інтегрується з Spring, дозволяючи генерувати HTML та інші формати у веб-додатках.[45]

Для оптимізації продуктивності сервера за допомогою кешування даних в програму інтегровано Ehcache — це широко використовувана бібліотека кешування на основі Java, яка надає рішення для кешування в пам’яті для покращення продуктивності програми шляхом зменшення навантаження на базу даних.[46]

Використовуючи ці інструменти та технології, розроблена високопродуктивна, масштабована та ефективна система CRM для вищих навчальних закладів, яка забезпечує плавну взаємодію з користувачами.

3.3.4 Тестування програмного забезпечення

Метою цього розділу є звіт про тестування та оцінка функціональності, ефективності та якості програмного забезпечення CRM, розробленого для закладів вищої освіти. Процес тестування мав на меті виявити будь-які помилки, дефекти або розбіжності в програмному

забезпеченні та оцінити його загальну продуктивність за заздалегідь визначеними критеріями.[47]

Цілю тестування було перевірте функціональність системи CRM, включаючи взаємодію з інтерфейсом користувача, перевірку даних і бізнес-логіку, оцінка продуктивності та масштабованості системи в умовах нормального та пікового навантаження. Також було взято до уваги перевірка функції безпеки, засоби контролю доступу та механізми захисту даних, реалізовані в програмному забезпеченні.

Для тестування використовувалися комбінації ручних і автоматизованих методів тестування. Розроблено комплексні тестові приклади, що охоплюють різні аспекти програмного забезпечення CRM. Проведено тестування пристрою, інтеграції та системи для підтвердження різних рівнів функціональності. Пріоритезація заходів тестування на основі критичності та впливу на кінцевих користувачів.[48]

Виконані тестові випадки в кількох середовищах тестування, що моделюють виробничі умови. Проведено ручне тестування для перевірки взаємодії користувача, цілісності даних і поведінки системи та автоматизоване тестування з використанням інфраструктур тестування для автоматизації повторюваних сценаріїв тестування. Проведене модульне тестування для перевірки функціональності окремих компонентів та інтеграційне тестування для перевірки взаємодії між модулями.

За результатами тестування було виявлено та задокументовано загальну кількість 4 дефектів і невідповідностей під час тестування. Класифіковані дефекти на основі рівнів серйозності, починаючи від критичних проблем, що впливають на основні функції, і закінчуючи незначними невідповідностями в елементах інтерфейсу користувача. Вирішено 100% повідомлених дефектів шляхом подальшого тестування

та виправлення помилок. Перевірено функції безпеки системи CRM, включаючи автентифікацію користувачів, контроль доступу та механізми шифрування даних. Тестування продуктивності виявило 18% покращення чутливості системи та пропускної здатності після заходів з оптимізації продуктивності.

Отже, завдяки різним механізмам тестування було усунуто критичні дефекти та аномалії, виявлені під час тестування, щоб забезпечити надійність і зручність використання програмного забезпечення CRM. Впроваджені додаткові заходи безпеки для посилення захисту даних і запобігання несанкціонованому доступу та відстежено продуктивність і масштабованість системи, щоб відповідати майбутньому зростанню та попиту користувачів.

Процес тестування дав цінну інформацію про функціональність, ефективність і якість програмного забезпечення CRM, розробленого для закладів вищої освіти. Усуваючи виявлені дефекти та впроваджуючи рекомендовані вдосконалення, система CRM може ефективно задовольняти потреби користувачів і сприяти покращенню відносин із клієнтами та ефективності організації.

3.3.5 Керівництво (інструкція) користувача

Інструкція користувача є вичерпним посібником для різних груп користувачів, які беруть участь у використанні та управлінні системою CRM, розробленою для вищих навчальних закладів. Метою цієї інструкції є надання детальних рекомендацій, призначених для програмістів, системних адміністраторів, кінцевих користувачів та інших зацікавлених сторін, залучених до експлуатації та обслуговування програмного забезпечення.

Для програмістів:

- Встановлення Java Development Kit (JDK) версії 17 або новішої.
- Вибір інтегрованого середовище розробки (IDE), сумісне з Java і Maven, наприклад IntelliJ IDEA або Eclipse.
- Клонування репозиторію проекту з системи контролю версій (наприклад, Git).
- Налаштування залежності проекту за допомогою Maven, налаштувавши файл `pom.xml`.
- Інтеграція з системами контролю версій для спільної розробки, дотримуючись кращих практик розгалуження та злиття.
- Ознайомлення зі структурою проекту, включаючи каталоги, пакети та основні компоненти (наприклад, контролери, служби, сховища).
- Дотримання стандартів кодування та умов, визначених у проектній документації.
- Забезпення повної документацію коду за допомогою JavaDoc або еквівалентних стандартів.

Для системних адміністраторів:

- Завантаження Apache Tomcat або подібний контейнер сервлетів для розміщення програми CRM.
- Налаштування підключення до бази даних у файлі властивостей програми (наприклад, `application.properties`).
- Розгортання артефактів програми, створених Maven, на сервері Tomcat.
- Перевірка розгортання, щоб переконатися, що програма доступна за вказаною URL-адресою.
- Налаштування ролі користувачів і дозволи в програмі, призначаючи відповідні рівні доступу для різних груп користувачів.
- Налаштування параметрів сповіщень електронною поштою для системних сповіщень і сповіщень користувачів.

- Інтеграція із зовнішніми службами, такими як LDAP, для автентифікації та авторизації користувачів.

- Відстеження стану системи за допомогою інструментів моніторингу, таких як Nagios або Zabbix, відстежуючи такі показники, як використання ЦП, споживання пам'яті та продуктивність бази даних.
- Виконання регулярних завданнь з обслуговування, включаючи резервне копіювання бази даних, ротацію файлів журналу та оновлення програмного забезпечення.
- Оптимізація використання ресурсів, налаштувавши конфігурації сервера та налаштувавши параметри JVM за потреби.
- Регулярна перевірка плану аварійного відновлення, щоб переконатися в його ефективності в зниженні потенційних ризиків.
- Задокументування процедур відновлення та розподіл обов’язктв між членами команди, щоб сприяти своєчасному реагуванню в разі надзвичайних ситуацій.

Для кінцевих користувачів:

- Реєстрація нового облікові записи користувачів за допомогою наданої реєстраційної форми, надаючи необхідну особисту інформацію.
- Автентифікація користувачів за допомогою автентифікації імені користувача/паролю.
- Навігація інтерфейсом системи CRM за допомогою наданих меню та навігаційних посилань.
- Доступ до різних модулів системи, таких як реєстрація студентів, керування курсами та звітність.
- Вводити та оновлювати дані в системі через призначені форми введення, забезпечуючи точність і повноту даних.
- Перевірка введених даних відповідно до попередньо визначених правил перевірки, щоб запобігти помилкам і невідповідностям.

- Інтерпретація аналітичних результатів для визначення областей для вдосконалення та оптимізації організаційних стратегій.

Посібник користувача є комплексним ресурсом для користувачів системи CRM, пропонуючи докладні інструкції та керівництво для різних груп користувачів. Дотримуючись інструкцій, викладених у цьому розділі, користувачі можуть ефективно використовувати та керувати програмним забезпеченням CRM для підвищення ефективності роботи та досягнення цілей організації.

3.4. Технічне забезпечення

3.4.1 Обґрунтування вибору технічного забезпечення

Обґрунтування вибору технічного забезпечення розробленої системи CRM для вищих навчальних закладів передбачає оцінку наявних технічних засобів у відповідній предметній галузі та аналіз їх відповідності вимогам системи. У контексті CRM-системи необхідно враховувати кілька технічних аспектів. Серверна інфраструктурами оцінює існуючі серверні технології, такі як служби хмарного хостингу (наприклад, AWS, Azure, Google Cloud)[49] і постачальників виділених серверів. Важливо оцінити такі фактори, як масштабованість, надійність, функції безпеки та економічна ефективність. Враховуючи конфіденційність даних студентів, заходи безпеки та дотримання правил захисту даних (таких як GDPR) є ключовими міркуваннями. Вибір СУБД (наприклад, PostgreSQL) має бути обґрунтованим на основі таких факторів, як продуктивність, масштабованість, легкість інтеграції та підтримка реляційного керування даними. Враховуючи реляційну природу даних і необхідність відповідності ACID, PostgreSQL стає підходящим вибором завдяки своїй надійності, підтримці спільноти та сумісності з нашим стеком технологій.[50] Були

проаналізовані доступні фреймворки та бібліотеки, сумісні з Java Spring, PostgreSQL та іншими компонентами стеку технологій. Це включає оцінку зрілості, підтримки спільноти, документації та простоти інтеграції цих фреймворків. Наприклад, використання Spring Framework виправдовується завдяки його широким можливостям, включаючи впровадження залежностей, аспектно-орієнтоване програмування та архітектуру MVC[51], які добре відповідають вимогам системи CRM.

Були використані засоби розробки та IDE, які покращують продуктивність, співпрацю та якість коду.[52] Такі фактори, як системи контролю версій (наприклад, Git),[53] інструменти безперервної інтеграції та розгортання (наприклад, Jenkins) і системи відстеження проблем (наприклад, Jira), оцінюються на основі їх придатності для робочого процесу розробки та командної співпраці. Також проведена оцінка доступних протоколів безпеки та механізми автентифікації, щоб забезпечити захист конфіденційних даних і конфіденційності користувачів. Це передбачає розгляд таких варіантів, як OAuth для автентифікації та авторизації користувачів, HTTPS для безпечного зв’язку та методи шифрування для захисту даних у стані спокою та під час передачі.[54]

На основі порівняльного аналізу цих технічних засобів та їх узгодження з вимогами та цілями CRM-системи можна обґрунтувати вибір конкретних інструментів, фреймворків та компонентів інфраструктури. Обрана технічна підтримка має забезпечити ефективну розробку, розгортання та підтримку системи CRM, одночасно забезпечуючи масштабованість, безпеку та відповідність галузевим стандартам і правилам.

3.4.2 Ресурсні вимоги до технічного забезпечення

Характеристики обладнання, необхідного для забезпечення продуктивної роботи розробленої CRM-системи вищих навчальних

закладів мають вирішальне значення для підтримки продуктивності, масштабованості та надійності системи. Розгортання системи CRM відбувається на серверах, оснащених багатоядерними процесорами (наприклад, Intel Xeon, AMD EPYC), щоб ефективно обробляти одночасні запити користувачів. Для додатку виділено достатню кількість оперативної пам'яті для забезпечення робочого набору системи та вимог до кешування.[55] Ціллю було отримати мінімум 8 ГБ оперативної пам’яті з можливостями масштабування, які можна розширити в міру зростання бази користувачів. Використано накопичувач SSD (твердотільний накопичувач) для швидшого отримання даних і покращеної чутливості системи та виділено достатній простір для зберігання програми CRM, бази даних і будь-яких пов’язаних файлів або носіїв. Не менш важливо було забезпечити високошвидкісне підключення до мережі, щоб мінімізувати затримку та підтримувати безперебійну передачу даних між клієнтами та сервером.

Вибрано потужний ЦП, здатний ефективно обробляти складні запити бази даних і транзакції та виділено пам'ять для сервера бази даних, щоб оптимізувати продуктивність запитів і зменшити операції вводу/виводу диска. Для зберігання впровадженно високопродуктивні рішення для зберігання даних (наприклад, масиви SSD RAID), щоб забезпечити швидке читання/записування та мінімізувати затримку отримання даних. Запровадженно надійну систему резервного копіювання та відновлення, щоб запобігти втраті даних і забезпечити безперервність роботи в разі апаратних збоїв або аварій та розширені пристрої безпеки для захисту системи CRM від несанкціонованого доступу, зловмисного програмного забезпечення та інших кіберзагроз.[56] Програмне забезпечення для моніторингу (наприклад, Nagios, Zabbix) для відстеження показників продуктивності системи, виявлення вузьких місць і завчасного вирішення проблем та централізовану консоль керування для оптимізації завдань системного адміністрування, таких як керування конфігурацією,

оновлення програмного забезпечення та контроль доступу користувачів також було враховано в розробкі CRM системи для ВНЗ. Для маштабованості і резервування було реалізовано механізми балансування навантаження для розподілу вхідного трафіку між кількома серверами та запобігання перевантаженню окремих вузлів та налаштовано резервні апаратні компоненти (наприклад, блоки живлення, мережеві інтерфейси) і використано механізми відновлення після збоїв, щоб забезпечити високу доступність і відмовостійкість.

Дотримуючись цих вимог до ресурсів для технічної підтримки, система CRM може працювати продуктивно, відповідати очікуванням ефективності та ефективно задовольняти потреби закладів вищої освіти та їх зацікавлених сторін.

3.5. Реалізація інформаційної управляючої системи

Критерії та методології, які використовуються для розробки та впровадження ІУС у рамках завершеного проекту управління взаємовідносинами з клієнтами (CRM) для університетів, рекомендації та пропозиції ґрунтуються на базових теоретичних основах, методологічних підходах, алгоритмічних рішеннях і технологічних інструментах, що використовуються протягом життєвого циклу проекту. Основною метою є забезпечення оптимальної функціональності, цільової ефективності та практичної життєздатності ІУС.

В рамках проекту CRM були впроваджені нові методи збору, обробки й аналізу даних, що призвели до оригінальних рішень і результатів. Методи включали сегментацію клієнтів, аналітику залучення та персоналізовані комунікаційні стратегії, адаптовані до сфери вищої освіти. Крім того, були розроблені оригінальні моделі, які відображали складність взаємодії студентів та інституційну динаміку. Ці моделі сприяли створенню

систем прогнозної аналітики та систем підтримки прийняття рішень, спрямованих на посилення залучення та утримання студентів. Також, були розроблені технічні та програмні інновації, які дозволили бездоганно інтегрувати функції CRM в існуючу університетську інфраструктуру. Інструменти, такі як Apache Tomcat і Maven, спростили процеси розгортання та керування залежностями програмного забезпечення. Крім того, були запроваджені нові показники ефективності та критерії оцінки, які дозволили оцінити результати впровадження CRM. Ці показники охоплювали індекси задоволеності студентів, коефіцієнти конверсії зарахованих і показники залученості випускників, що надали корисну інформацію для прийняття стратегічних рішень.

Щодо практичної реалізації, система CRM була успішно апробована та протестована на різних об’ємах даних. Реальні результати можуть включати покращений набір студентів, покращені стосунки з випускниками та оптимізовані стратегії розподілу ресурсів. За допомогою комплексної економічної оцінки було виявлено що імовірною є значна віддача від інвестицій впровадження CRM.

Інтерфейс сайту для CRM системи ВНЗ буде включати :

Заголовок, який зазвичай містить логотип установи або системи CRM, що забезпечує ідентифікацію бренду та легку навігацію назад на домашню сторінку. Він містить основні елементи навігації, такі як посилання на різні модулі або розділи системи CRM, функції пошуку та параметри облікового запису користувача. Специфічні для користувача дії, такі як вхід/вихід, налаштування профілю та сповіщення, часто розміщуються в заголовку для легкого доступу.

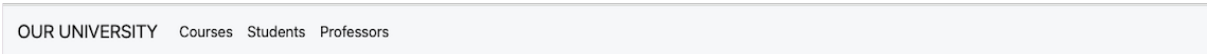


Рисунок 3.4 – Прототип header’у

Джерело: розроблено автором

Головне навігаційне меню розташоване під заголовком і складається з основних параметрів навігації, які дозволяють користувачам отримувати доступ до різних областей або функцій системи CRM.

Загальні пункти меню можуть включати «Інформаційна панель», «Курси», «Електронний журнал», «Повідомлення», «Календар», «Професійні наробки»(для викладачів) та «Налаштування».

Залежно від ролі та дозволів користувача певні пункти меню можуть бути видимими або прихованими.

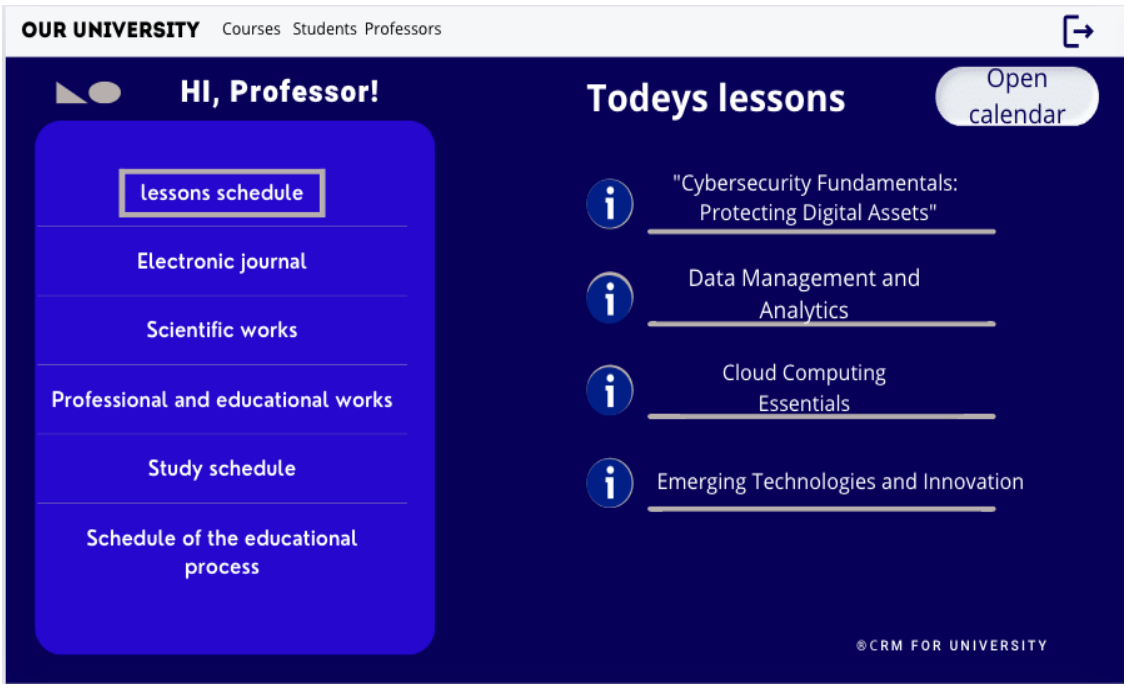


Рисунок 3.5 – Прототип меню для користувача з роллю Професор

Джерело: розроблено автором

Інформаційна панель служить центром, де користувачі можуть отримати огляд своєї відповідної інформації та діяльності.

Він може містити віджети або модулі, що відображають майбутні події, статус реєстрації на курс, останні оцінки, оголошення та завдання.

Параметри налаштування дозволяють користувачам персоналізувати макет інформаційної панелі та вибрати, які віджети відображати.

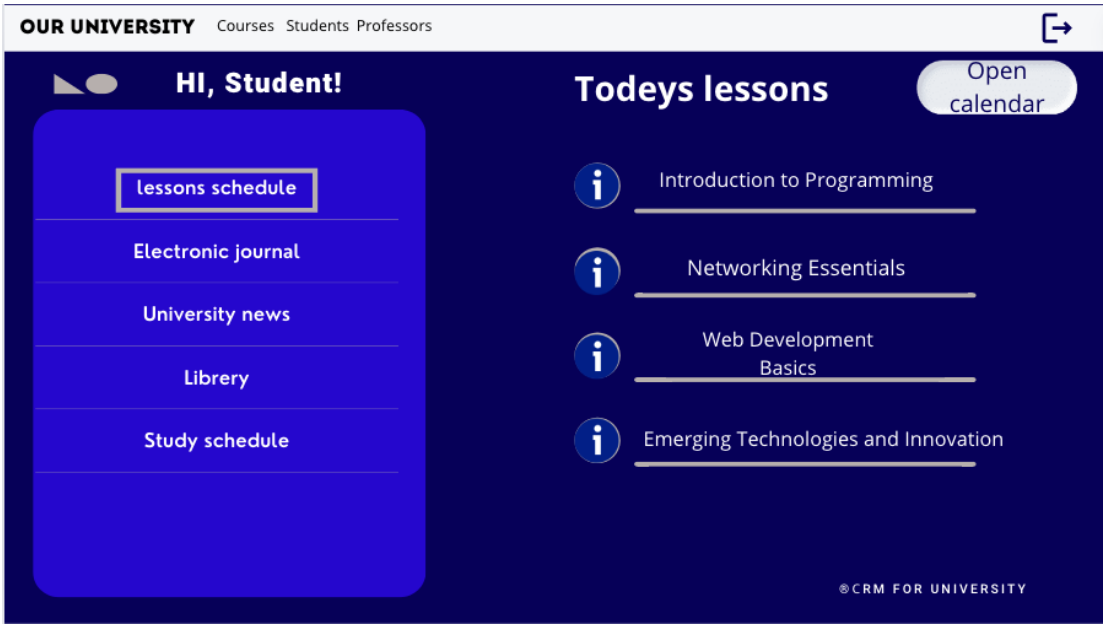


Рисунок 3.6 – Прототип відображення розкладу занять для користувача з роллю Студент

Джерело: розроблено автором

Нижній колонтитул зазвичай містить додаткові навігаційні посилання, контактну інформацію, юридичні повідомлення та посилання на відповідні ресурси чи правила.

Він може містити швидкі посилання на сторінки, до яких часто звертаються, піктограми соціальних мереж та інформацію про авторські права.

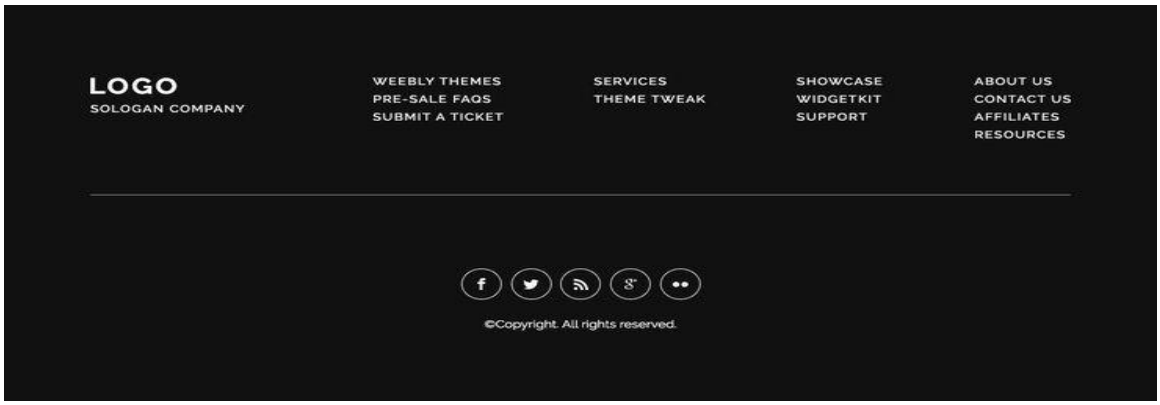


Рисунок 3.7 – Прототип flooter’a

Важливість мобільної версії системи управління для університетів неможливо переоцінити в сучасному цифровому середовищі. Завдяки

поширенню смартфонів і планшетів студенти, викладачі та співробітники очікують доступу до основних університетських послуг та інформації в будь-який час і в будь-якому місці. Мобільна версія системи гарантує, що користувачі можуть зручно взаємодіяти з платформою на своїх улюблених пристроях, незалежно від того, чи знаходяться вони в кампусі, вдома чи в дорозі. Доступність для мобільних пристроїв збільшує залучення користувачів, надаючи інтуїтивно зрозумілий інтерфейс для доступу до матеріалів курсу, розкладів, оцінок та іншої важливої інформації. Студенти більш схильні брати активну участь в академічних заходах і бути в курсі університетських подій і оновлень, коли вони можуть легко отримати до них доступ зі своїх мобільних пристроїв. Університети, які пропонують надійну та зручну мобільну систему, отримують конкурентну перевагу в залученні та утриманні студентів. Оцінюючи потенційні навчальні заклади, майбутні студенти враховують такі фактори,

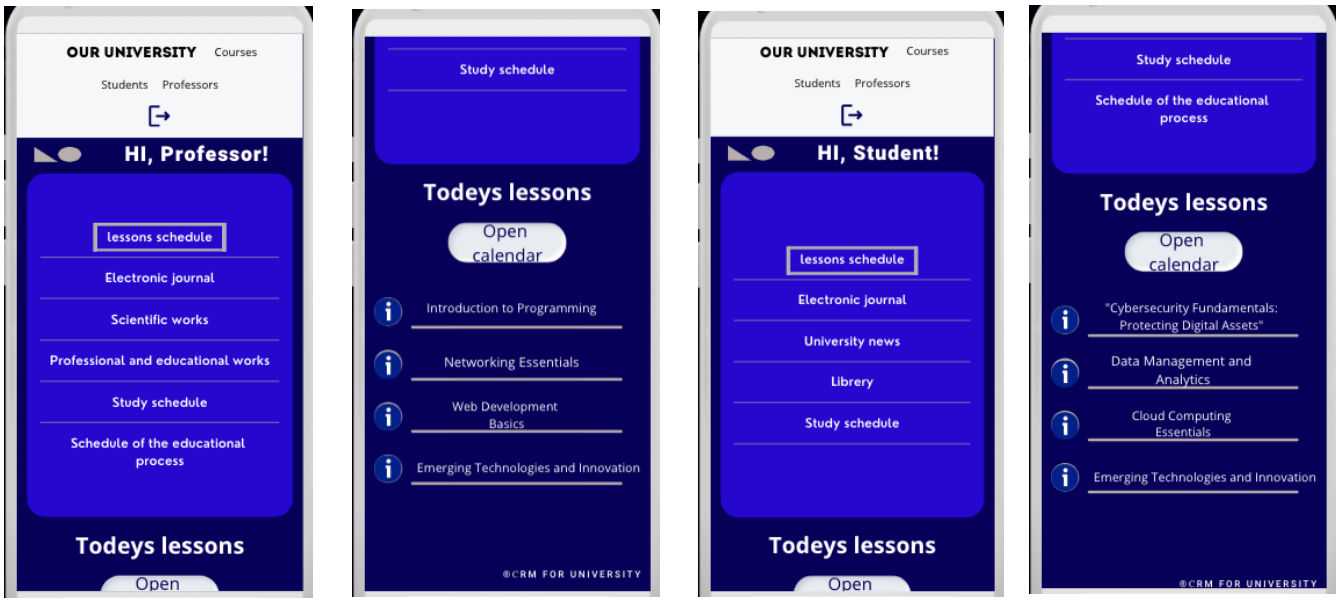


Рисунок 3.8 – Прототип відображення розкладу занять для користувачів з роллю Студент та Професcор

Джерело: розроблено автором

Таким чином, мобільна версія системи відіграє вирішальну роль у підвищенні доступності, залучення, персоналізації, ефективності та конкурентоспроможності університетів. Застосовуючи мобільні технології, навчальні заклади можуть надавати студентам, викладачам і співробітникам більш бездоганний і збагачувальний досвід, що зрештою сприяє їхньому успіху та задоволенню в академічній спільноті.

Проект CRM для ВНЗ відкриває нові можливості для подальших досліджень передових методів управління вищою освітою. Напрямки досліджень можуть включати аналіз настроїв, видобуток соціальних мереж і прогнозне моделювання. Також, сформульовано практичні рекомендації для керівництва університетами щодо ефективного розгортання та використання систем CRM. Основні міркування включають залучення зацікавлених сторін, дотримання конфіденційності даних і постійний моніторинг і оцінку системи.

ВИСНОВКИ

Підсумовуючи, розробка системи управління взаємовідносинами (CRM) дала значні результати, як якісні, так і кількісні, покращуючи процеси управління в секторі вищої освіти. Протягом усього проекту було прийнято комплексний підхід для вирішення різноманітних проблем, з якими стикаються університети щодо управління та залучення студентів. Використовуючи сучасні методології та інноваційні технології, проект здатний зробити значний внесок у сферу управління освітньою інформацією.

Одним із ключових досягнень проекту є впроваджені методичні досягнення. Були розроблені нові методи збору, обробки та аналізу даних, спеціально пристосовані до унікальних вимог сфери вищої освіти. Ці методи сприяли сегментації клієнтів, уможливлювали складну аналітику залучення студентів і підтримували формулювання персоналізованих комунікаційних стратегій. У результаті університети отримали цінну інформацію про поведінку та вподобання студентів, що дозволило вживати більш цілеспрямованих та ефективних заходів із поширення інформації.

Окрім методологічних інновацій, проект також призвів до розробки оригінальних моделей та програмних рішень. Ці моделі були розроблені, щоб відобразити складність взаємодії студентів та інституційну динаміку в університетських умовах. Використовуючи ці моделі, були створені системи прогнозної аналітики та інструменти підтримки прийняття рішень, щоб підвищити рівень залученості та утримання студентів. Крім того, розроблено спеціальну архітектуру програмного забезпечення для безпроблемної інтеграції функцій CRM в існуючу університетську інфраструктуру, забезпечуючи ефективне розгортання та керування системою.

Загалом, поставлені завдання були виконані на високому рівні, розроблена система CRM може бути успішно впроваджена на різних факультетах університетів. Реальні результати включали покращення кількості студентів, покращення стосунків з випускниками та оптимізовані стратегії розподілу ресурсів. Проект продемонстрував трансформаційний потенціал CRM-технологій в управлінні інформацією у вищих навчальних закладах.

Забігаючи наперед, проект також передбачає кілька напрямків для подальшої роботи над обраною темою. Майбутні дослідження можуть вивчати передові аналітичні методи, такі як аналіз настроїв, аналіз соціальних медіа та прогнозне моделювання для подальшого покращення стратегій залучення студентів та інституційної ефективності. Необхідно постійно вдосконалювати системи CRM, щоб адаптуватись до технологічних тенденцій, що розвиваються, і мінливих потреб користувачів. Крім того, розширення масштабів впровадження CRM для охоплення інших секторів у сфері освіти та сприяння співпраці та обміну знаннями між установами та галузевими партнерами буде важливим для максимізації впливу технологій CRM у секторі освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. J. O'Connell. A Roadmap for Improved Constituent Management at George Washington University. *Educause*. URL: https://er.educause.edu/articles/2023/2/a-roadmap-for-improved-constituent-management-at-george-washington-university.
2. Compare Ellucian CRM Advise and Salesforce Education Cloud. *g2.com*. URL: https://www.g2.com/compare/ellucian-crm-advise-vs-salesforce-education-cloud.
3. Ellucian CRM Advance | Ellucian. *Ellucian*. URL: https://www.ellucian.com/solutions/ellucian-crm-advance.
4. Home - TargetX |. *TargetX*. URL: https://www.targetx.com/.
5. Salesforce for education. *salesforce.org*. URL: https://www.salesforce.org/.
6. Technolutions. *Technolutions*. URL: https://technolutions.com/.
7. CRM and higher education: developing a monitoring system to improve relationships in e-learning environments. *researchgate.net*. URL: https://www.researchgate.net/publication/220398789_CRM_and_higher_education_developing_a_monitoring_system_to_improve_relationships_in_e-learning_environments.
8. What are the Structure and Functions of Management Information System?. *misnotesformba*. URL: https://misnotesformba.wordpress.com/2015/01/15/what-are-the-structure-and-functions-of-management-information-system/
9. What is Use Case Diagram?. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: https://www.visual-

paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

10. Contributors to Wikimedia projects. Class diagram - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Class_diagram.
11. Contributors to Wikimedia projects. Sequence diagram - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Sequence_diagram.
12. Use Case Diagrams | Unified Modeling Language (UML) - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/use-case-diagram/.
13. What is Sequence Diagram?. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.
14. Decision-making process. *UMass Dartmouth*. URL: https://www.umassd.edu/fycm/decision-making/process/.
15. How can you use decision-making scenarios to anticipate market changes?. *LinkedIn: Log In or Sign Up*. URL: https://www.linkedin.com/advice/0/how-can-you-use-decision-making-scenarios-anticipate#:~:text=Decision-making%20scenarios%20are%20stories,data,%20events,%20or%20trends.
16. Mathematical models of information systems developing. *researchgate.net*. URL: https://www.researchgate.net/publication/216462793_Mathematical_models_of_information_systems_developing.
17. Contributors to Wikimedia projects. ERwin Data Modeler – Википедия. *Википедия – свободная энциклопедия*. URL: https://ru.wikipedia.org/wiki/ERwin_Data_Modeler.

18. erwin Data Modeler - DBMS Tools. *Data and Database Tools Catalog - DBMS Tools*. URL: https://dbmstools.com/tools/erwin-data-modeler.
19. What is ETL (Extract, Transform, Load)? | IBM. *IBM in Deutschland, Österreich und der Schweiz*. URL: https://www.ibm.com/topics/etl.
20. ELTs & Homebuilt Aircraft | EAA. *EAA | Experimental Aircraft Association | Oshkosh, Wisconsin*. URL: https://www.eaa.org/eaa/aircraft-building/intro-to-aircraft-building/frequently-asked-questions/elts-and-homebuilt-aircraft.
21. Sartorius. What Is Principal Component Analysis (PCA) and How It Is Used?. *Sartorius*. URL: https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186#:~:text=Principal%20component%20analysis,%20or%20PCA,more%20easily%20visualized%20and%20analyzed.
22. *CMU School of Computer Science*. URL: https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf.
23. Clustering. *University of Regina - Department of Computer Science*. URL: https://www2.cs.uregina.ca/~dbd/cs831/notes/clustering/clustering.html#:~:text=Two%20types%20of%20clustering%20algorithms,is%20included%20in%20the%20hierarchy.
24. Contributors to Wikimedia projects. Decision tree - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Decision_tree#:~:text=A%20decision%20tree%20is%20a,only%20contains%20conditional%20control%20statements.
25. Contributors to Wikimedia projects. Apriori algorithm - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Apriori_algorithm.

26. Contributors to Wikimedia projects. Naive Bayes classifier - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier.
27. Agrawal R. Shallow Neural Networks. *Medium*. URL: https://towardsdatascience.com/shallow-neural-networks-23594aa97a5.
28. Амортизаційна політика як складова процесу відтворення - CORE Reader. *CORE – Aggregating the world's open access research papers*. URL: https://core.ac.uk/reader/32620420.
29. Orange vs RapidMiner. *capterra.com*. URL: https://www.capterra.com/data-mining-software/compare/164505-148220/Orange-vs-RapidMiner.
30. Malovich L. The Different Steps in Database Development Life Cycle. *Medium*. URL: https://laura-malovich.medium.com/the-different-steps-in-database-development-life-cycle-aa79c92a0b8a.
31. What are the 7 Phases of Database Design? - DatabaseTown. *DatabaseTown*. URL: https://databasetown.com/what-are-the-7-phases-of-database-design/.
32. Luna-Reyes L. F., Martin E. G., Ivonchyk M. 7.4 Database Systems Development Life Cycle. *Pressbooks Create – Your partner in open publishing*. URL: https://pressbooks.pub/decisions/chapter/565/.
33. Simplilearn. Top Cloud Platforms to Watch Out for & Reasons to Learn Them. *Simplilearn.com*. URL: https://www.simplilearn.com/top-cloud-platforms-article.
34. Getting to Grips with MVVM Architecture. *Digital Acceleration Company | Netguru*. URL: https://www.netguru.com/blog/mvvm-architecture#:~:text=20%20min%20read-,The%20MVVM%20architecture,%20short%20for%20Model-View-ViewModel,,from%20the%20underlying%20business%20logic.

35. Contributors to Wikimedia projects. Model–view–controller - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Model–view–controller.
36. Contributors to Wikimedia projects. Monolithic architecture - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Monolithic_architecture#:~:text=Monolithic%20architecture%20describes%20buildings%20which,the%20Pancha%20Rathas%20in%20India.
37. What Are Distributed Architectures: 4 Types & Key Components. *Real-time ETL | Estuary*. URL: https://estuary.dev/distributed-architecture/#:~:text=Distributed%20architecture%20refers%20to%20a,relying%20on%20a%20central%20server.
38. What is SSL, TLS and HTTPS? | DigiCert. *SSL Digital Certificate Authority | Encryption & Authentication | DigiCert.com*. URL: https://www.digicert.com/what-is-ssl-tls-and-https.
39. What is Java technology and why do I need it?. *java.com*. URL: https://www.java.com/en/download/help/whatis_java.html.
40. Contributors to Wikimedia projects. Spring Framework - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Spring_Framework.
41. Contributors to Wikimedia projects. Hibernate (framework) - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Hibernate_(framework).
42. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/.
43. Contributors to Wikimedia projects. Apache Tomcat - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Apache_Tomcat.
44. Gaba I. What is Maven: Here's What You Need to Know [Updated]. *Simplilearn.com*.

URL: https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven.

45. Introduction to Using Thymeleaf in Spring. *baeldung*. URL: https://www.baeldung.com/thymeleaf-in-spring-mvc.
46. Ehcache. *Ehcache*. URL: https://www.ehcache.org/.
47. How do you test and evaluate your CRM system before and after deployment?. *LinkedIn*. URL: https://www.linkedin.com/advice/3/how-do-you-test-evaluate-your-crm-system.
48. 5 Best Practices for CRM Testing. *Qualysoft Group - Competence-based IT services since 1999*. URL: https://qualysoft.com/en/blog/5-best-practices-crm-testing-ensure-efficiency-and-success.
49. What's the Difference Between AWS vs. Azure vs. Google Cloud?. *Coursera*. URL: https://www.coursera.org/articles/aws-vs-azure-vs-google-cloud?utm_medium=sem&utm_source=gg&utm_campaign=b2c_emea_coursera_ftcof_career-academy_arte_march_24_dr_geo-multi-set3_pmax_gads_lg-all&campaignid=21103949440&adgroupid=&device=c&keyword=&matchtype=&network=x&devicemodel=&adposition=&creativeid=&hide_mobile_promo&gad_source=1&gclid=Cj0KCQjwqpSwBhClARIsADlZ_TlV0LsiUDynn7GBtVFmtzcyRRjqTR94W79zqqdZi95vSd9V-5sMInsaAtQQEALw_wcB.
50. What is a database?. *Oracle | Cloud Applications and Cloud Platform*. URL: https://www.oracle.com/database/what-is-database/#:~:text=Is%20a%20Database?-,Database%20defined,database%20management%20system%20(DBMS).
51. Web MVC framework. *Spring | Home*. URL: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html.

52. Contributors to Wikimedia projects. IDE - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/IDE.
53. Contributors to Wikimedia projects. Git - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Git.
54. How HTTPS Works. *How HTTPS works - How HTTPS works*. URL: https://howhttps.works/ru/https-ssl-tls-differences/.
55. Contributors to Wikimedia projects. Multi-core processor - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Multi-core_processor.
56. Sliwa C. What is SSD RAID (solid-state drive RAID)? | Definition from TechTarget. *Storage*. URL: https://www.techtarget.com/searchstorage/definition/SSD-RAID-solid-state-drive-RAID#:~:text=The%20term%20SSD%20RAID%20is,and%20superior%20I/O%20performance.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного код

```
@SpringBootApplication(scanBasePackages = "com.university")

public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

```
@RequestMapping("/courses")
@Controller
public class CourseController {

    @Autowired
    private final CourseService courseService;
    @Autowired
    private ProfessorService professorService;
    @Autowired
    private StudentService studentService;
    @Autowired
    public CourseController(CourseService courseService) {
        this.courseService = courseService;
    }

    @GetMapping("/new")
    public String newCourse(Model model){
        model.addAttribute("course", new Course());
        return "course/new";
    }

    @PostMapping()
    public String create(@ModelAttribute("course") Course course){
        this.courseService.save(course);

professorService.delete(courseService.getProfessorOfCourse(course.getId
()).getId());
        return "redirect:/courses";
    }
    @GetMapping()
    String findAll(Model model){
        model.addAttribute("courses",
                courseService.findAll().stream()
                        .map(c -> new Course(c.getId(),
c.getCourseName(), c.getStartDate(), c.getEndDate(), c.getStudents(),
c.getProfessor()))
                        .collect(Collectors.toList()));
        return "course/findAll";
    }
        @GetMapping("/edit/{id}")
    public String update(Model model, @PathVariable("id") Long id){
        model.addAttribute("course", courseService.findById(id));
```

```
        return                                          "course/updateCourse";
    }
    @PatchMapping                                              ("/edit/{id}")
    public   String   edit(@ModelAttribute("course")   Course   course,
@PathVariable                          Long                            id){
        courseService.update(id,                 course.getCourseName(),
course.getStartDate(),                             course.getEndDate());
        return                                      "redirect:/courses";
    }

    @GetMapping                                               ("/delete/{id}")
    public       String       delete(@PathVariable       Long       id){
        courseService.delete(id);
        return                                          "redirect:/courses";
    }

    @GetMapping("/set-professor/{id}")
    public String setProfessorToCourse(Model model, @PathVariable("id")
Long                              courseId)                                 {
        model.addAttribute("course",  courseService.findById(courseId));
        model.addAttribute("professors",
professorService.findAllProfessors());
        return                                  "course/professorForCourse";
    }

    @PatchMapping("/set-professor/{id}/{pId}")
    public String setProfessor(@ModelAttribute("course") Course course,
                               @PathVariable("pId") Long professorId) {
        courseService.setProfessorToCourse(professorId,
course.getId());
        return                                          "redirect:/courses";
    }

    @GetMapping("/get-professor/{id}")
    public String getProfessorOfCourse(Model model, @PathVariable Long
id)                                                                       {
        model.addAttribute("professor",
courseService.getProfessorOfCourse(id));
        model.addAttribute("course",        courseService.findById(id));
        return                                   "course/professorOfCourse";
    }

    @GetMapping("/set-student/{id}")
    public String setStudentToCourse(Model model, @PathVariable("id")
Long                              courseId)                                 {
        model.addAttribute("courseForStudent",
courseService.findById(courseId));
        model.addAttribute("allStudents",
studentService.findAllStudent());
        model.addAttribute("students",
courseService.findById(courseId).getStudents());
        model.addAttribute("studentNotFromCourse",
courseService.getStudentsNotFromCourse(courseId));
        return                                    "course/studentForCourse";
    }
    @PatchMapping                                 ("/set-student/{id}/{sId}")
    public String setStudent(@ModelAttribute("courseForStudent") Course
course,
```

```
                            @PathVariable("sId") Long studentId) {
        courseService.setStudentForCourse(studentId, course.getId());
        return "redirect:/courses";
    }
    @GetMapping("/get-student/{id}")
    public String getStudentOfCourse(Model model, @PathVariable Long id)
{
        model.addAttribute("students",
courseService.getStudentOfCourse(id));
        model.addAttribute("course", courseService.findById(id));
        return "course/studentOfCourse";
    }

    @GetMapping("/delete-professor/{cId}/{pId}")
    public String deleteProfessorFromCourse(@PathVariable Long cId,
@PathVariable Long pId) {
        courseService.deleteProfessorFromCourse(cId, pId);
        return "redirect:/courses";
    }

    @GetMapping("/delete-student/{sId}")
    public String deleteStudentFromCourse(@PathVariable Long sId) {
        courseService.deleteStudentFromCourse(sId);
        return "redirect:/courses";
    }
}
```

```
      @Controller
@RequestMapping("/professors")
public class ProfessorController {

    @Autowired
    private final ProfessorService professorService;
    @Autowired
    public ProfessorController(ProfessorService professorService) {
        this.professorService = professorService;
    }

    @GetMapping("/new")
    public String createProfessor(Model model) {
        model.addAttribute("professor", new Professor());
        return "professor/new";
    }
    @PostMapping()
    public String newProfessor(@ModelAttribute("professor") Professor
professor) {
        professorService.create(professor);
        return "redirect:/professors";
    }
    @GetMapping()
    public String showAllProfessors(Model model) {
        model.addAttribute("professors",
professorService.findAllProfessors());
        return "professor/allProfessors";
    }
    @GetMapping("/delete/{id}")
    public String deleteProfessor(@PathVariable Long id) {
        professorService.delete(id);
```

```
        return                                  "redirect:/professors";
    }
    @GetMapping("/edit/{id}")
    public String editProfessor(Model model, @PathVariable("id") Long id)
{
        model.addAttribute("professor",  professorService.findById(id));
        return                                  "professor/updateProfessor";
    }
    @PatchMapping("/edit/{id}")
    public String updateProfessor(@ModelAttribute("professor") Professor
professor,         @PathVariable("id")         Long         id)         {
        professorService.updateProfessor(id,       professor.getName());
        return                                  "redirect:/professors";
    }
```

```
       @Controller
@RequestMapping("/students")
public              class              StudentController              {

    @Autowired
    private                     StudentService          studentService;

    @GetMapping
    public       String       findAllStudents(Model       model)       {
        model.addAttribute("students",
studentService.findAllStudent());
        return                                  "student/findAllStudents";
    }

    @GetMapping("/new")
    public         String         newStudent(Model         model)         {
        model.addAttribute("student",          new          Student());
        return                                   "student/new";
    }

    @PostMapping
    public   String   createStudent(@ModelAttribute("student")   Student
student)                                                                {
        this.studentService.saveStudent(student);
        return                                  "redirect:/students";
    }


    @GetMapping("/delete/{id}")
    public    String    deleteStudent(@PathVariable    Long    id)    {
        studentService.deleteStudent(id);
        return                                  "redirect:/students";
    }
       @GetMapping("/edit/{id}")
public String editStudent(Model model, @PathVariable("id") Long id) {
    model.addAttribute("student",    studentService.getStudentById(id));
    return                                  "student/updateStudent";
}

@PatchMapping("/edit/{id}")
public   String   update(@ModelAttribute("student")   Student   student,
@PathVariable("id")               Long               id)               {
    studentService.update(id, student.getName(), student.getLastName());
```

```
    return "redirect:/students";
}
```

```
@Entity
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String courseName;
    private String startDate;
    private String endDate;

    @OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    private List<Student> students = new ArrayList<>();

    @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    private Professor professor = new Professor();

    public Course(){

    }
    public Course(String courseName, String startDate, String endDate,
List<Student> students, Professor professor) {
        this.courseName = courseName;
        this.startDate = startDate;
        this.endDate = endDate;
        this.students = students;
        this.professor = professor;
    }
    public Course(Long id,String courseName, String startDate, String
endDate, List<Student> students, Professor professor) {
        this.id = id;
        this.courseName = courseName;
        this.startDate = startDate;
        this.endDate = endDate;
        this.students = students;
        this.professor = professor;
    }


    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getCourseName() {
        return courseName;
    }

    public void setCourseName(String courseName) {
        this.courseName = courseName;
    }

    public String getStartDate() {
        return startDate;
```

```
    }

    public void setStartDate(String startDate) {
        this.startDate = startDate;
    }

    public String getEndDate() {
        return endDate;
    }

    public void setEndDate(String endDate) {
        this.endDate = endDate;
    }

    public List<Student> getStudents() {
        return students;
    }

    public void setStudents(List<Student> students) {
        this.students = students;
    }

    public Professor getProfessor() {
        return professor;
    }

    public void setProfessor(Professor professor) {
        this.professor = professor;
    }

    @Override
    public String toString() {
        return "Course{" +
                ", courseName='" + courseName + '\'' +
                ", startDate='" + startDate + '\'' +
                ", endDate='" + endDate + '\'' +
                ", students=" + students.toString() +
                ", professor=" + professor.getName() +
                '}';
    }
}
```

```
@Entity
public class Professor {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;

    public Professor(){

    }

    public Professor(Long id, String name) {
        this.id = id;
        this.name = name;
    }
```

```
    public Professor(String name) {
        this.name = name;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    @Override
    public String toString() {
        return name;
    }
}
```

```
    @Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;
    private String lastName;

    public Student(){
    }
    public Long getId() {
        return id;
    }

    public Student(String name, String lastName) {
        this.name = name;
        this.lastName = lastName;
    }
    public Student(Long id, String name, String lastName) {
        this.id = id;
        this.name = name;
        this.lastName = lastName;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public void setId(Long id) {
        this.id = id;
    }


    @Override
    public String toString() {
        return "Student " + name + " " + lastName + " with ID - " + id;
    }
}
```

```
@Service
public class CourseService {
    @Autowired
    private final CourseRepository courseRepository;
    @Autowired
    private StudentRepository studentRepository;
    @Autowired
    public CourseService(CourseRepository courseRepository) {
        this.courseRepository = courseRepository;
    }

    @PersistenceContext
    private EntityManager entityManager;

    public Course findById(Long id){
        Optional<Course> foundCourse = courseRepository.findById(id);
        return foundCourse.orElse(null);
    }

    @Transactional
    public Long save(Course course){
        return courseRepository.save(course).getId();
    }
    public List<Course> findAll(){
        return courseRepository.findAll();
    }
    @Transactional
    public void update(Long id, String courseName, String startDate,
String endDate){
        Course updatedCourse = courseRepository.findById(id)
                .orElseThrow(() -> new IllegalStateException("course
with id: " + id + "does not exist"));
        updatedCourse.setCourseName(courseName);
        updatedCourse.setStartDate(startDate);
        updatedCourse.setEndDate(endDate);

    }
    @Transactional
```

```
    public void delete(Long id){
        courseRepository.deleteById(id);
    }

    @Transactional
    public void setProfessorToCourse(Long professorId, Long courseId) {
        courseRepository.setProfessorForCourse(professorId, courseId);
        courseRepository.setCourseForProfessor(professorId, courseId);
    }

    public Professor getProfessorOfCourse(Long id) {
        Course course = courseRepository.findById(id).get();
        return course.getProfessor();
    }

    public List<Student> getStudentOfCourse(Long id) {
        Course course = courseRepository.findById(id).get();
        return course.getStudents();
    }

    public List<Student> getStudentsNotFromCourse (Long courseId){
        List<Student> studentListOfCourse =
courseRepository.getOne(courseId).getStudents();
        List<Student> listAllStudent = studentRepository.findAll();
        listAllStudent.removeAll(studentListOfCourse);
        return listAllStudent;
    }
    @Transactional
    public void deleteProfessorFromCourse(Long cId, Long pId) {
        courseRepository.deleteCourseForProfessor(pId);
        courseRepository.deleteProfessorForCourse(cId);
    }

    @Transactional
    public void setStudentForCourse(Long cId, Long sId) {
        entityManager.createNativeQuery("INSERT INTO course_students
(course_id, students_id) VALUES (?,?)")
                .setParameter(1, sId)
                .setParameter(2, cId)
                .executeUpdate();
    }
    @Transactional
    public void deleteStudentFromCourse(Long sId) {
        courseRepository.deleteStudentFromCourse(sId);
    }
}
```

```
    <?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>university-sys</artifactId>
    <version>1.0-SNAPSHOT</version>
```

```
    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    </properties>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.2.RELEASE</version>
        <relativePath/>
    </parent>

    <dependencies>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
```

```
        <dependency>
            <groupId>org.hibernate.validator</groupId>
            <artifactId>hibernate-validator</artifactId>

        </dependency>


        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
        </dependency>

        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-dbcp2</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>


        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>


    </dependencies>

    <build>
        <finalName>university-sys</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>8</source>
                    <target>8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

```
      spring:
  thymeleaf:
    prefix:                              file:src/main/resources/templates/
    cache:                                                              false

  jpa:
    database:                                                       POSTGRESQL
    properties.hibernate.temp.use_jdbc_metadata_defaults:                false
    hibernate:
      ddl-auto:                                                       update

  datasource:
    platform:                                                         postgres
    url:                jdbc:postgresql://localhost:5432/universitysystem
    username:                                                         postgres
    password:                                                             2901
    driver-class-name:                                 org.postgresql.Driver

server:
  port: 9999
```

```
      <!DOCTYPE                                                          html>
<html                                xmlns:th="https://www.thymeleaf.org">
<head>
  <meta                                                    charset="UTF-8">
  <title>All                  university                  students</title>
  <meta name="viewport" content="width-device-width, initial-scale=1.0">
  <link                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css">
  <script                              src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.
js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.mi
n.js"></script>
  <script        src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper
.min.js"                                                integrity="sha384-
vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh"
crossorigin="anonymous"></script>
  <script          src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.2/js/bootstrap.min.js"                             integrity="sha384-
alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ"
crossorigin="anonymous"></script>
  <link                                                    rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.m
in.css"
        integrity="sha384-
```

```
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
        crossorigin="anonymous"                                          />
</head>
<body>
<nav    class="navbar    navbar-expand-lg    navbar-light    bg-light">
  <a       class="navbar-brand"       href="#">OUR       UNIVERSITY</a>
  <button  class="navbar-toggler"  type="button"  data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle                                          navigation">
    <span                            class="navbar-toggler-icon"></span>
  </button>
  <div       class="collapse       navbar-collapse"       id="navbarNav">
    <ul                                          class="navbar-nav">
      <li                     class="nav-item                    active">
        <a      class="nav-link"      th:href="@{/courses}">Courses<span
class="sr-only">(current)</span></a>
      </li>
      <li                     class="nav-item                    active">
        <a     class="nav-link"     th:href="@{/students}">Students<span
class="sr-only">(current)</span></a>
      </li>
      <li                     class="nav-item                    active">
        <a   class="nav-link"   th:href="@{/professors}">Professors<span
class="sr-only">(current)</span></a>
      </li>
    </ul>
  </div>
</nav>
<div                                                    class="row">
  <div                                              class="col-md-4">
  </div>
  <div align="center" ></div> <h2> ALL UNIVERSITY COURSES </h2>  </div>

<br>

<div                                              class="container">

  <h3>List                        Of                        Courses</h3>

  <hr/>
  <a   th:href="@{/courses/new}"   class="btn   btn-primary">Create   new
course</a>
  <br/>
  <br/>

  <table        class="table        table-bordered        table-striped">

    <thead>
    <tr>
      <th>ID</th>
      <th>Course                                              name</th>
      <th>Date                       of                      start</th>
      <th>Date                        of                       end</th>
      <th>Professor</th>
      <th>Students</th>
      <th>Actions</th>
```

```
    </tr>
    </thead>
    <tbody>
    <tr                      th:each="c:                      ${courses}">
      <td                        th:text="${c.id}"                        />
      <td                     th:text="${c.courseName}"                   />
      <td                     th:text="${c.startDate}"                    />
      <td                      th:text="${c.endDate}"                     />
      <td>
        <span           th:if="${c.professor          !=          null}">
          <a        th:href="@{/courses/get-professor/{id}(id=${c.id})}"
class="btn                                      btn-info">Professor</a>
        </span>
        <span           th:if="${c.professor          ==          null}">
          <a        th:href="@{/courses/set-professor/{id}(id=${c.id})}"
class="btn                                      btn-info">Professor</a>
        </span>
      </td>
      <td>
          <span          th:if="${c.students          !=          null}">
          <a          th:href="@{/courses/get-student/{id}(id=${c.id})}"
class="btn                                       btn-info">Students</a>
        </span>
        <span          th:if="${c.students          ==          null}">
          <a          th:href="@{/courses/set-student/{id}(id=${c.id})}"
class="btn                                       btn-info">Students</a>
        </span>
      </td>
      <td>
        <a  th:href="@{/courses/edit/{id}(id=${c.id})}"  class="btn  btn-
info">Edit</a>
        <a th:href="@{/courses/delete/{id}(id=${c.id})}" class="btn btn-
danger">Delete</a>
      </td>

    </tr>
    </tbody>
  </table>

</div>
</body>
</html>
```

```
       <!DOCTYPE                                                    html>
<html                              xmlns:th="https://www.thymeleaf.org">
<head>
    <meta                                                 charset="UTF-8">
    <title>All                  university                students</title>
    <meta    name="viewport"    content="width-device-width,    initial-
scale=1.0">
    <link                                                 rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css">
    <script                           src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.
js"></script>
```

```
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.mi
n.js"></script>
    <link                                                rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.m
in.css"
          integrity="sha384-
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
          crossorigin="anonymous"                                    />
</head>
<body>
<nav    class="navbar    navbar-expand-lg    navbar-light    bg-light">
    <a       class="navbar-brand"       href="#">OUR       UNIVERSITY</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle                                          navigation">
        <span                          class="navbar-toggler-icon"></span>
    </button>
    <div      class="collapse      navbar-collapse"      id="navbarNav">
        <ul                                          class="navbar-nav">
            <li                  class="nav-item                  active">
                <a  class="nav-link"  th:href="@{/courses}">Courses<span
class="sr-only">(current)</span></a>
            </li>
            <li                  class="nav-item                  active">
                <a class="nav-link" th:href="@{/students}">Students<span
class="sr-only">(current)</span></a>
            </li>
            <li                  class="nav-item                  active">
                <a                                       class="nav-link"
th:href="@{/professors}">Professors<span                    class="sr-
only">(current)</span></a>
            </li>
        </ul>
    </div>
</nav>
<div                                                  align="center">
    <h1>Create new course</h1>
```

```
      <!DOCTYPE                                                    html>
<html                            xmlns:th="https://www.thymeleaf.org">
<head>
  <meta                                                charset="UTF-8">
  <title>All                 university                students</title>
  <meta name="viewport" content="width-device-width, initial-scale=1.0">
  <link                                                rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css">
  <script                          src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.
js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.mi
n.js"></script>
  <script        src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
```

```
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper
.min.js"                                              integrity="sha384-
vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh"
crossorigin="anonymous"></script>
  <script          src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.2/js/bootstrap.min.js"                           integrity="sha384-
alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ"
crossorigin="anonymous"></script>
  <link                                                 rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.m
in.css"
        integrity="sha384-
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
        crossorigin="anonymous"                                          />
</head>
<body>
<nav    class="navbar    navbar-expand-lg    navbar-light    bg-light">
  <a       class="navbar-brand"       href="#">OUR       UNIVERSITY</a>
  <button  class="navbar-toggler"  type="button"  data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle                                              navigation">
    <span                              class="navbar-toggler-icon"></span>
  </button>
  <div      class="collapse      navbar-collapse"      id="navbarNav">
    <ul                                             class="navbar-nav">
      <li                    class="nav-item                    active">
        <a      class="nav-link"      th:href="@{/courses}">Courses<span
class="sr-only">(current)</span></a>
      </li>
      <li                    class="nav-item                    active">
        <a     class="nav-link"     th:href="@{/students}">Students<span
class="sr-only">(current)</span></a>
      </li>
      <li                    class="nav-item                    active">
        <a   class="nav-link"   th:href="@{/professors}">Professors<span
class="sr-only">(current)</span></a>
      </li>
    </ul>
  </div>
</nav>
<div                                                        class="row">
  <div                                                 class="col-md-4">
  </div>
  <div align="center" ></div> <h2> ADD PROFESSOR TO COURSE  </h2>  </div>

<br>

<span           th:if="${professors.size()           ==           0}">
  <div                                                  align="center">
  <h3>  No  one  professor's  in  base!  Want  to  add  first?</h3>
  <br/>
  <a    th:href="@{/professors/new}"    class="btn    btn-primary">Add
Professor</a>
```

```
  <br/>
    </div>
</span>

<span            th:if="${professors.size()            >            0}">

 <div                                                    align="center">
  <h3  th:text="  'No  one  professors  set  for  '+${course.courseName}+'
course    in    base!    Set    one    or    create    new!'">    </h3>
  <br/>
  <a   th:href="@{/professors/new}"   class="btn   btn-info">Create   new
professor</a>
  <br/>
    </div>

  <div                                              class="container">
  <h3>List                        Of                      Professor</h3>
  <br/>
        <div           th:each="p           :           ${professors}">

           <table    class="table    table-bordered    table-striped">

             <thead>
             <tr>
               <th>ID</th>
               <th>Professor                                    name</th>
               <th>Actions</th>
                     </tr>
                     </thead>
                     <tbody>
                       <td         th:text="${p.id}"                  />
                       <td            th:text="${p.name}"             />
                       <td>
                          <form                    th:object="${course}"
                                th:action="@{/courses/set-
professor/{id}/{pId}          (id=${course.id},          pId=${p.id})}"
                                th:method="patch">
                            <button    type="submit"    class="btn    btn-
primary">SET</button>
                         </form>

                       </td>

               </tr>
               </tbody>
             </table>

        </div>
 </div>
</span>

</body>
```

```
      <!DOCTYPE                                                  html>
<html                          xmlns:th="https://www.thymeleaf.org">
<head>
    <meta                                          charset="UTF-8">
    <title>All              university              professors</title>
```

```
    <meta    name="viewport"    content="width-device-width,    initial-
scale=1.0">
    <link                                                rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css">
    <script                          src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.
js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.mi
n.js"></script>
    <script       src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper
.min.js"                                             integrity="sha384-
vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh"
crossorigin="anonymous"></script>
    <script        src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.2/js/bootstrap.min.js"                          integrity="sha384-
alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ"
crossorigin="anonymous"></script>
    <link                                                rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.m
in.css"
          integrity="sha384-
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
          crossorigin="anonymous"                                    />
</head>
<body>
<nav    class="navbar    navbar-expand-lg    navbar-light    bg-light">
    <a       class="navbar-brand"       href="#">OUR       UNIVERSITY</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle                                          navigation">
        <span                        class="navbar-toggler-icon"></span>
    </button>
    <div      class="collapse      navbar-collapse"      id="navbarNav">
        <ul                                          class="navbar-nav">
            <li                 class="nav-item                 active">
                <a  class="nav-link"  th:href="@{/courses}">Courses<span
class="sr-only">(current)</span></a>
            </li>
            <li                 class="nav-item                 active">
                <a class="nav-link" th:href="@{/students}">Students<span
class="sr-only">(current)</span></a>
            </li>
            <li                 class="nav-item                 active">
                <a                                       class="nav-link"
th:href="@{/professors}">Professors<span                    class="sr-
only">(current)</span></a>
            </li>
        </ul>
```

```
    </div>
</nav>

<br>
<div                                                    class="row">
    <div                                              class="col-md-4">
    </div>
    <div                      align="center"                      ></div>
    <h2 th:text=" 'Professor of ' + ${course.courseName}">Hi, User</h2>
</div>

<br>
<div                                                    class="row">
    <div                  class="offset-md-2                  col-md-8">

    <table       class="table       table-bordered       table-striped">

        <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
        </tr>
        </thead>
        <tbody>
            <td               th:text="${professor.id}"               />
            <td              th:text="${professor.name}"              />
            <td>
                <a
th:href="@{/professors/edit/{id}(id=${professor.id})}"  class="btn  btn-
info">Edit</a>

                <a      th:href="@{/courses/delete-professor/{cId}/{pId}
(cId=${course.id},      pId=${professor.id})}"      class="btn      btn-
danger">Delete</a>
            </td>
        </tr>
        </tbody>
    </table>
    </div>
</div>

</div>

</body>
</html>
```

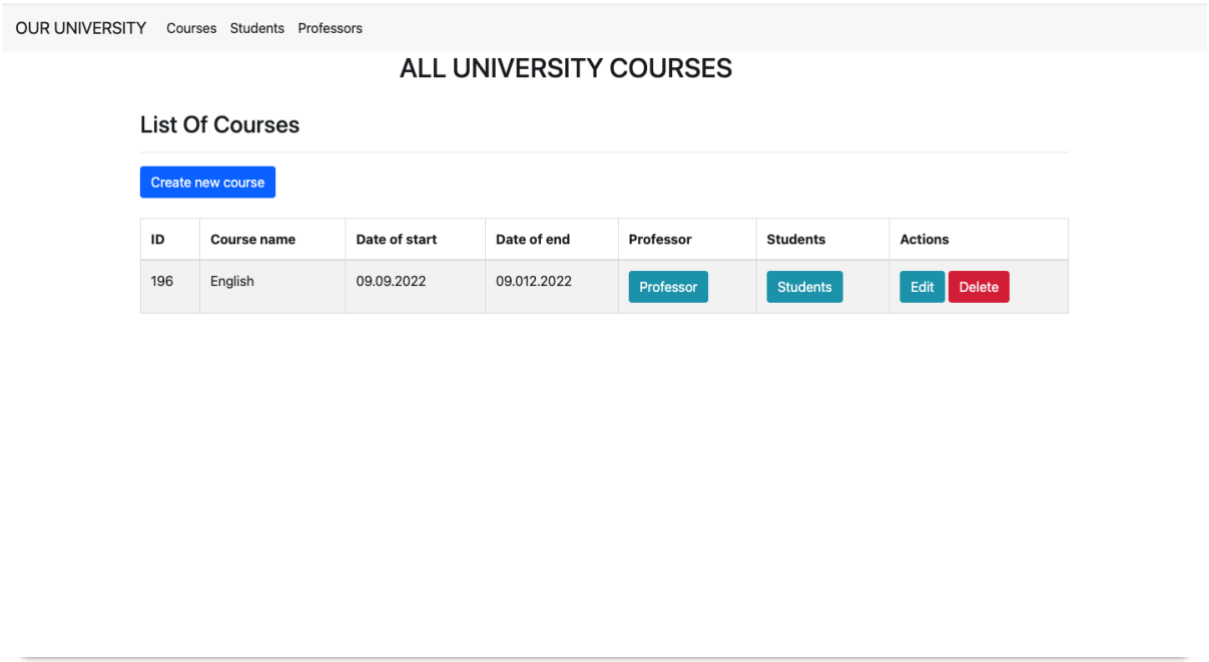
```
      <!DOCTYPE                                                 html>
<html                           xmlns:th="https://www.thymeleaf.org">
<head>
  <meta                                                 charset="UTF-8">
  <title>All                  university                  students</title>
  <meta name="viewport" content="width-device-width, initial-scale=1.0">
  <link                                                 rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css">
  <script                           src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
  <script
```

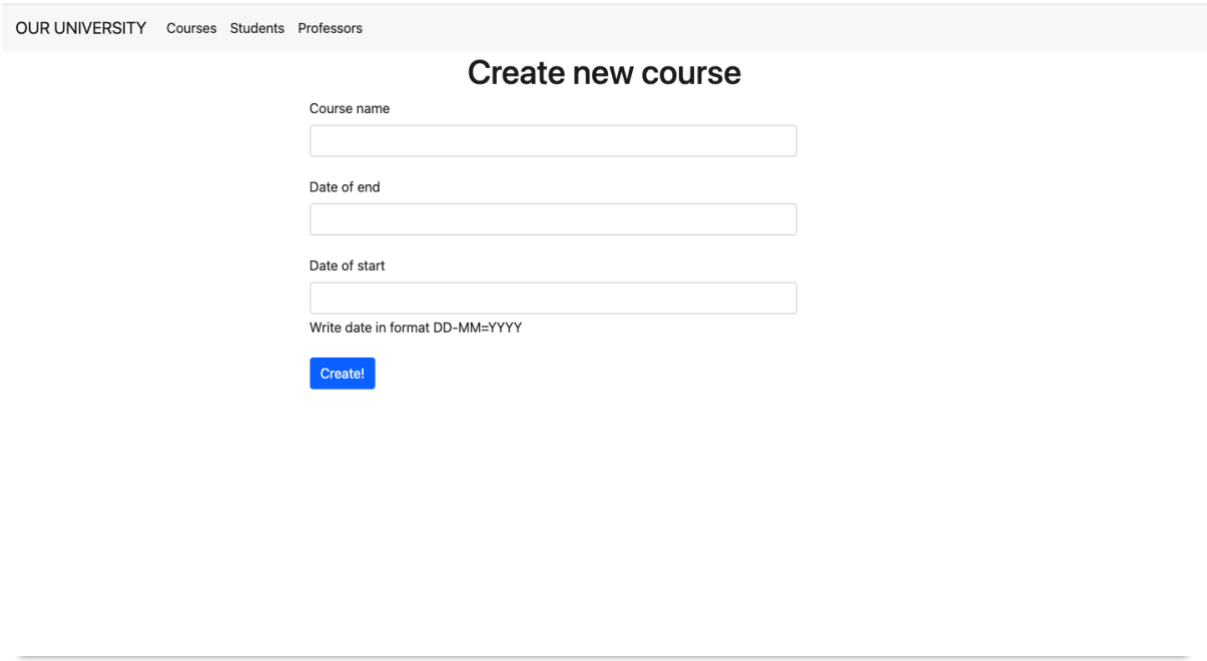
```
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.
js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.mi
n.js"></script>
  <script        src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper
.min.js"                                                integrity="sha384-
vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh"
crossorigin="anonymous"></script>
  <script          src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.2/js/bootstrap.min.js"                             integrity="sha384-
alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ"
crossorigin="anonymous"></script>
  <link                                                  rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.m
in.css"
        integrity="sha384-
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
        crossorigin="anonymous"                                          />
</head>
<body>
<nav    class="navbar    navbar-expand-lg    navbar-light    bg-light">
  <a       class="navbar-brand"       href="#">OUR       UNIVERSITY</a>
  <button  class="navbar-toggler"  type="button"  data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle                                        navigation">
    <span                              class="navbar-toggler-icon"></span>
  </button>
  <div       class="collapse       navbar-collapse"       id="navbarNav">
    <ul                                              class="navbar-nav">
      <li                     class="nav-item                    active">
        <a      class="nav-link"      th:href="@{/courses}">Courses<span
class="sr-only">(current)</span></a>
      </li>
      <li                     class="nav-item                    active">
        <a     class="nav-link"     th:href="@{/students}">Students<span
class="sr-only">(current)</span></a>
      </li>
      <li                     class="nav-item                    active">
        <a   class="nav-link"   th:href="@{/professors}">Professors<span
class="sr-only">(current)</span></a>
      </li>
    </ul>
  </div>
</nav>
<div                                                      class="row">
  <div                                                 class="col-md-4">
  </div>
  <div align="center" ></div> <h2> ADD STUDENTS TO COURSE  </h2>  </div>

<br>
```

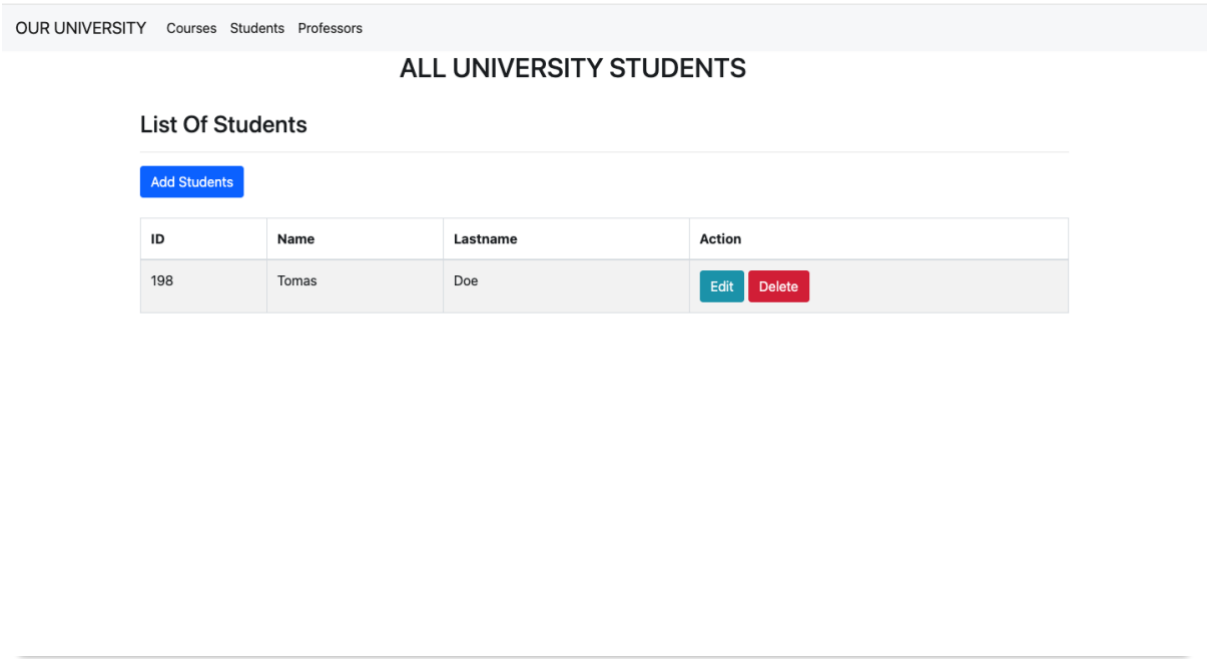
```
<span            th:if="${allStudents.size()            ==            0}">

  <h3 ><center>No one student's in base! Want to add first?</center></h3>
  <br/>
  <center><a   th:href="@{/students/new}"   class="btn   btn-primary">Add
Student</a></center>
  <br/>

</span>
<span            th:if="${allStudents.size()            >            0}">

<div                                                  class="container">

  <h3>List                        Of                        Students</h3>
  <br/>
        <div        th:each="s        :        ${studentNotFromCourse}">
            <table     class="table     table-bordered     table-striped">

              <thead>
              <tr>
                <th>ID</th>
                <th>Student                                        name</th>
                <th>Student                last                    name</th>
                <th>Actions</th>
                      </tr>
                      </thead>
                      <tbody>
                        <td                              th:text="${s.id}"/>
                        <td            th:text="${s.name}"             />
                        <td          th:text="${s.lastName}"           />
                        <td>
                           <form         th:object="${courseForStudent}"
                                 th:action="@{/courses/set-
student/{id}/{sId}       (id=${courseForStudent.id},       sId=${s.id})}"
                                 th:method="patch">
                             <button    type="submit"    class="btn    btn-
primary">ADD</button>
                          </form>

                        </td>

                        </tr>
                </tbody>
              </table>
        </div>

</div>
```

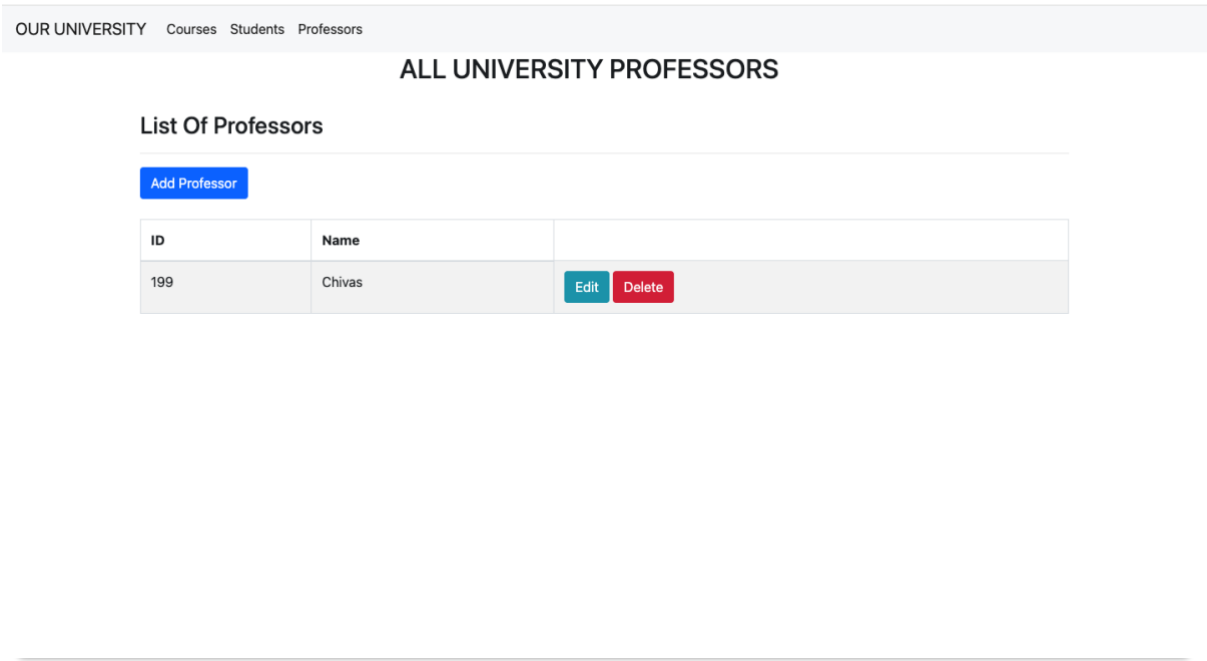
ДОДАТОК Б

Прототип інтерфейсу додатку






OUR UNIVERSITY
Courses
Students
Professors
ALL UNIVERSITY STUDENTS
List Of Students
Add Students
ID
Name
Lastname
Action
198
Tomas
Doe
Edit
Delete


OUR UNIVERSITY
Courses
Students
Professors
ALL UNIVERSITY PROFESSORS
List Of Professors
Add Professor
ID
Name
199
Chivas
Edit
Delete

OUR UNIVERSITY Courses Students Professors

ADD PROFESSOR TO COURSE

No one professors set for English course in base! Set one or create new!

Create new professor

List Of Professor

ID	Professor name	Actions
199	Chivas	SET

OUR UNIVERSITY Courses Students Professors

Students of English

ID	Name	Last name	
198	Tomas	Doe	Edit Delete

Add more students

Додаток В

Реалізація JWT токену

```
@Monitor
@RestController
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RequiredArgsConstructor
@RequestMapping("/api/v1/auth")
@CrossOrigin(origins = "http://localhost:3000")
public class AuthController {

    AuthenticationService authenticationService;

    @PostMapping("/authenticate")
    public ResponseEntity<AuthenticationResponse>
authenticate(@RequestBody AuthenticationRequest request) {
        return
ResponseEntity.ok(authenticationService.authenticate(request));
    }

    @PostMapping("/logout")
    public ResponseEntity<?> logout(@RequestHeader("Authorization")
String token) {
        authenticationService.logout(token);
        return ResponseEntity.ok().build();
    }
}
```

```
@Service
@Slf4j
@RequiredArgsConstructor
public class AuthenticationService {

    private final AccountService service;
    private final TokenService tokenService;
    private final JwtService jwtService;
    private final AuthenticationManager authenticationManager;

    public AuthenticationResponse authenticate(AuthenticationRequest
request) {
        authenticationManager.authenticate(
                new
UsernamePasswordAuthenticationToken(request.getUsername(),
request.getPassword()));

        var user = service.getUserByUserName(request.getUsername());
        var jwtToken = jwtService.generateToken(user);
        var refreshToken = jwtService.generateRefreshToken(user);
        checkAllUsersTokenToExpired(user);
        saveUserToken(user, jwtToken);

        return AuthenticationResponse.builder()
                .accessToken(jwtToken)
                .refreshToken(refreshToken)
                .role(user.getAccountRole())
                .structureUnit(user.getStructureUnit())
                .build();
```

```
    }

    private    void    checkAllUsersTokenToExpired(Account    user)    {
        List<Token>                  allTokensByAccount                  =
tokenService.findAllValidTokensByAccount(user.getId());
        allTokensByAccount.forEach(t     ->     isTokenExpired(t.token));
    }


    private        void        isTokenExpired(String        token)        {
        try                                                              {
            DecodedJWT        decodedJWT        =        JWT.decode(token);
            Date        expiresAt        =        decodedJWT.getExpiresAt();
            if(new                Date().after(expiresAt))                {
                Token     byToken     =     tokenService.findByToken(token);
                byToken.setExpired(true);
                byToken.setRevoked(true);
                tokenService.save(byToken);
            }
        }          catch          (JWTDecodeException          e)          {
            log.error(e.getMessage());
        }
    }

    public          void          logout(String          token)          {
        String          validToken          =          token.substring(7);
        Token     byToken     =     tokenService.findByToken(validToken);
        var   validUserTokens   =   tokenService.findByToken(validToken);
        invalidateAndSaveTokens(validUserTokens);
    }

    private  void  saveUserToken(Account  account,  String  jwtToken)  {
        var              token              =              Token.builder()
                .account(account)
                .token(jwtToken)
                .tokenType(TokenType.BEARER)
                .expired(false)
                .revoked(false)
                .build();
        tokenService.save(token);
    }

    private     void     invalidateAndSaveTokens(Token     token)     {
            token.setExpired(true);
            token.setRevoked(true);
        tokenService.save(token);
    }
}
```

```
        @Component
@Slf4j
@RequiredArgsConstructor
public  class  JwtAuthenticationFilter  extends  OncePerRequestFilter  {

    private            final            JwtService            jwtService;
    private       final       UserDetailsService       userDetailsService;
    private         final         TokenRepository         tokenRepository;
```

```
    @Override
    protected                    void                    doFilterInternal(
          @NonNull               HttpServletRequest               request,
          @NonNull              HttpServletResponse              response,
          @NonNull                 FilterChain                filterChain
    )        throws        ServletException,        IOException        {

       if      (request.getServletPath().contains("/api/v1/auth"))      {
          filterChain.doFilter(request,                        response);
          return;
       }
       final  String  authHeader  =  request.getHeader("Authorization");
       final                          String                          jwt;
       final                       String                       userLogin;
       if  (authHeader  ==  null  ||  !authHeader.startsWith("Bearer  "))  {
          filterChain.doFilter(request,                        response);
          return;
       }
       jwt                   =                  authHeader.substring(7);
       userLogin             =             jwtService.extractLogin(jwt);


       if           (userLogin           !=           null           &&
SecurityContextHolder.getContext().getAuthentication()    ==    null)    {
          UserDetails                   userDetails                    =
this.userDetailsService.loadUserByUsername(userLogin);
          var     isTokenValid     =     tokenRepository.findByToken(jwt)
                .map(t     ->     !t.isExpired()     &&     !t.isRevoked())
                .orElse(false);
          if          (jwtService.isTokenValid(jwt,          userDetails)
                &&                                          isTokenValid
          )                                                          {
             UsernamePasswordAuthenticationToken    authToken    =    new
UsernamePasswordAuthenticationToken(
                   userDetails,
                   null,
                   userDetails.getAuthorities()
             );
             authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authToken);
          }
       }


       filterChain.doFilter(request,                        response);
    }

}
```

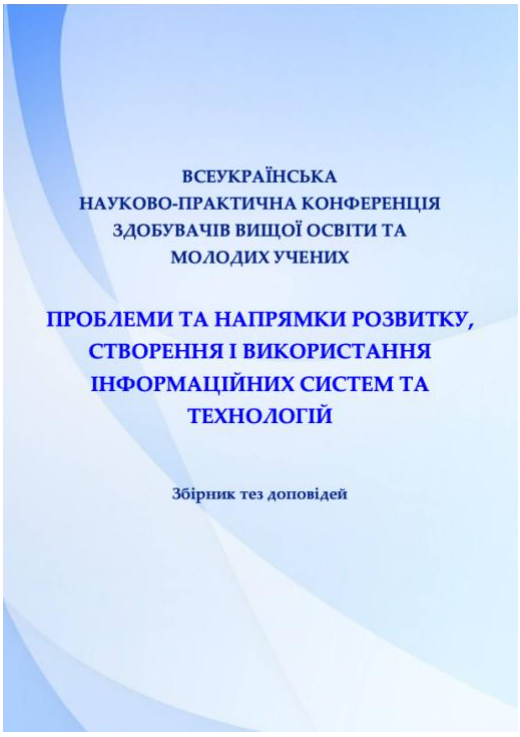
```
       @Service
public                 class                 JwtService                 {

    @Value("${application.security.jwt.secret-key}")
    private                      String                      secretKey;
    @Value("${application.security.jwt.expiration}")
```

```
    private                     long                    jwtExpiration;
    @Value("${application.security.jwt.refresh-token.expiration}")
    private                   long                  refreshExpiration;

    public        String        extractLogin(String        token)        {
       return          extractClaim(token,          Claims::getSubject);
    }

    public   <T>   T   extractClaim(String   token,   Function<Claims,   T>
claimsResolver)                                                          {
       final      Claims      claims      =      extractAllClaims(token);
       return                                claimsResolver.apply(claims);
    }

    public     String     generateToken(UserDetails     userDetails)     {
       return      generateToken(new      HashMap<>(),      userDetails);
    }

    public   String   generateToken(Map<String,   Object>   extraClaims,
UserDetails                          userDetails)                        {
       return   buildToken(extraClaims,   userDetails,   jwtExpiration);
    }

    public   String   generateRefreshToken(UserDetails   userDetails)   {
       return       buildToken(new       HashMap<>(),       userDetails,
refreshExpiration);
    }

    public boolean isTokenValid(String token, UserDetails userDetails) {
       final       String      username      =      extractLogin(token);
       return       (username.equals(userDetails.getUsername()))       &&
!isTokenExpired(token);
    }

    private    String    buildToken(Map<String,    Object>    extraClaims,
UserDetails        userDetails,        long        expiration)        {
       return                                                      Jwts
             .builder()
             .setClaims(extraClaims)
             .setSubject(userDetails.getUsername())
             .setIssuedAt(new          Date(System.currentTimeMillis()))
             .setExpiration(new     Date(System.currentTimeMillis()     +
expiration))
             .signWith(getSignInKey(),         SignatureAlgorithm.HS256)
             .compact();
    }

    private       boolean       isTokenExpired(String       token)       {
       return        extractExpiration(token).before(new        Date());
    }

    private       Date       extractExpiration(String       token)       {
       return        extractClaim(token,        Claims::getExpiration);
    }

    private       Claims       extractAllClaims(String       token)       {
       return                                                      Jwts
             .parserBuilder()
```

```
                .setSigningKey(getSignInKey())
                .build()
                .parseClaimsJws(token)
                .getBody();
    }

    private                Key                getSignInKey()                {
        byte[]      keyBytes      =      Decoders.BASE64.decode(secretKey);
        return                                Keys.hmacShaKeyFor(keyBytes);
    }
}
```

```
        @Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
public                  class                   Token                   {

    @Id
    @GeneratedValue(strategy          =          GenerationType.IDENTITY)
    public                              Long                              id;

    @Column(unique                            =                          true)
    public                           String                            token;

    @Enumerated(EnumType.STRING)
    public       TokenType       tokenType       =       TokenType.BEARER;

    public                           boolean                          revoked;

    public                           boolean                          expired;

    @ManyToOne(fetch                    =                  FetchType.EAGER)
    @JoinColumn(name                      =                    "account_id")
    @JsonIgnore
    public                           Account                          account;
}
```

Публікації



2. Бородін О. І. Використання нейронних мереж для кредитного скорингу підприємства / О. І. Бородін; ЧНУ ім. Петра Могили. – Миколаїв, 2022. – 103 с.

ТЕХНОЛОГІЧНІ РІШЕННЯ СПРОМОЖНІ ЗМЕНШИТИ БІДНОСТІ В КРАЇНАХ, ЩО РОЗВИВАЮТЬСЯ

Лянкало Ю.А.
спеціальність 122 – Комп'ютерні науки, ОПП «Інформаційні управляючі системи і технології», 5 курс
науковий керівник: кандидат технічних наук, доцент, професор кафедри інформаційних систем в економіці Іванченко Г.Ф.
Київський національний економічний університет імені Вадима Гетьмана м. Київ, Україна

Доведено, що надання та забезпечення дешевих, доступних і надійних технологічних рішень може зменшити бідність у країнах, що розвиваються. Вони також забезпечують зростання виробництва енергії та доступ до питної води. Інновації необхідні для покращення життя тих, хто цього потребує. Ось декілька технологічних рішень для подолання бідності в країнах, що розвиваються.

Цифрові пристрої. Людство наразі має більше доступу до цифрових пристроїв, ніж будь-коли. У країнах, що розвиваються, використання цифрових пристроїв дозволить точно збирати дані. Цей збір даних надає можливості для покращення сектору охорони здоров'я та харчування. Наприклад, Гарвардська школа громадської охорони здоров'я ефективно пояснила, чому та як хвороби поширюються в Кенії. Дослідники використовували статистичні дані з цифрових пристроїв, щоб ефективно визначити місце поширення хвороб [1]. У країнах, що розвиваються, цифрові пристрої також можуть допомогти в розвитку дрібним фермерам. Наприклад, WeFarm — це безкоштовна цифрова мережа, яка об'єднує фермерів у Кенії, Уганді та Танзанії. WeFarm використовує штучний інтелект, щоб зв'язувати фермерів зі схожими запитаннями та відповідями [2]. Впровадження WeFarm знизило вартість транзакцій місцевих фермерів. Це також збільшило консультаційні можливості та доходи дрібних фермерів. Таким чином, цифрові пристрої є успішним технологічним рішенням, яке пом'якшує бідність.

Онлайн-навчання. Крім того, онлайн-навчання є одним із інших технологічних рішень, які можуть подолати бідність у країнах, що розвиваються. Покращення освітніх можливостей має важливе значення для загального зростання нації. На жаль, окремі регіони країн, що розвиваються, не мають доступу до послуг особистої освіти. Тому онлайн-навчання долає цю прогалину. Африканський віртуальний університет (AVU) — це некомерційна організація, яка надає курси вищої освіти громадянам країн Африки. AVU пропонує онлайн-курси навчання від 50 університетів [3].

Гідроенергетика. Однією зі стратегій гідроенергетики є створення універсальної греблі. Гідроенергетика також забезпечує громади чистою, дешевою та стабільною енергією. Оцінка проекту підтвердила здатність розвитку гідроенергетики подолати бідність.

Загалом, доступні технологічні рішення зменшують глобальну бідність у країнах, що розвиваються. Оскільки технологія продовжує розвиватися, послуги мають стати менш ексклюзивними і, отже, більш доступними для країн, що розвиваються [4].

Перелік посилань:

1. Harvard School of Public Health. Using cell phone data to curb the spread of malaria [Електронний ресурс] / Harvard School of Public Health – Режим доступу до ресурсу: https://www.hsph.harvard.edu/news/press-releases/cell-phone-data-malaria/.

2. Welcome to Wefarm [Електронний ресурс] – Режим доступу до ресурсу: https://wefarm.com/moja.

3. African Virtual University: Transforming Africa into a Global Knowledge Hub [Електронний ресурс] – Режим доступу до ресурсу: https://www.afdb.org/fr/projects-and-operations/selected-projects/african-virtual-university-transforming-africa-into-a-global-knowledge-hub-88.

4. TECHNOLOGICAL SOLUTIONS TO POVERTY [Електронний ресурс] – Режим доступу до ресурсу: https://borgenproject.org/10-technological-solutions-poverty/.

UNICHECK
by Turnitin

Ім'я користувача:
Інформаційних систем в економіці Шкуратовська Те...

ID перевірки:
1016258531

Дата перевірки:
17.05.2024 21:16:52 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
17.05.2024 22:04:29 EEST

ID користувача:
100005745

Назва документа: Ляпкало Ю. ІУС-601з

Кількість сторінок: 74 Кількість слів: 10873 Кількість символів: 91738 Розмір файлу: 1.17 MB ID файлу: 1016046421

2.8%
Схожість

Найбільша схожість: 0.92% з джерелом з Бібліотеки (ID файлу: 1016022887)

0.95% Джерела з Інтернету	86	Сторінка 76
2.69% Джерела з Бібліотеки	267	Сторінка 76

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	16