НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ Ігор БОЛБОТ	**_____ Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення системи з документообігу в деканаті вищого навчального закладу»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Дмитро ШЕВЧЕНКО**
(підпис)

Консультант бакалаврської кваліфікаційної роботи
к.т.н., доцент _____ **Белла ГОЛУБ**
(підпис)

Виконав _____ **Юрій МИКИТИН**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Микитину Юрію Романовичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення системи з документообігу в деканаті вищого навчального закладу»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): Правила призначення академічних стипендій у Національному університеті біоресурсів і природокористування України від 22.11.2023, Форма додатка до диплома європейського зразка.
Перелік питань, що підлягають дослідженню:
Документообіг в деканаті вищого навчального закладу як об'єкт розробки, проєктування інформаційного забезпечення, розробка програмного забезпечення, рекомендації щодо впровадження та експлуатації системи.
Дата видачі завдання «19» грудня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Дмитро ШЕВЧЕНКО**

Завдання взяв до виконання _____ (підпис) **Юрій МИКИТИН**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Representational State Transfer (REST) – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

API (Application Programming Interface) – інтерфейс доступу до ресурсів програмного застосунку.

MVC – архітектура Model-View-Controller.

HTTP (HyperText Transfer Protocol) – протокол передачі даних Всесвітній павутині (інтернеті).

PDF (portable document format) – формат зчитування документів

МОН – Міністерство освіти і науки України

ЄДЕБО – Єдина державна електронна база з питань освіти

СУБД – Система управління базами даних

ВСТУП

Запорукою успішності сучасного вищого навчального закладу є постійне впровадження сучасних технологій. Вони застосовуються буквально скрізь: в навчальному процесі, в веденні документації, в адміністративних рішеннях. Основна мета таких рішень – прискорити роботу працівників та зменшити вплив людського фактору на важливі процеси. При роботі зі студентами завданнями університету є: зарахувати студента, призначити йому навчальний план, вести облік його успішності, виплачувати стипендію в процесі навчання і супровід випуску та формування додатків до дипломів державного зразка.

За 2025 рік в Національний університет біоресурсів і природокористування України вступило 5494 студенти [1]. А отже, з кожним потрібно виконувати вищевказані дії. Для автоматизації таких процесів також можна застосувати сучасні методи формування документів, збереження та передачі даних. Оскільки, формування документів можна прискорити і акумулювання даних у єдиному інформаційному середовищі, з’являється можливість моніторингу даних та перевірки на прозорість роботи деканатів вищих навчальних закладів.

Об’єктом розробки є програмне забезпечення системи з документообігу в деканаті вищого навчального закладу. При розробці важливо дотримуватись всіх положень і правил в Національному університеті біоресурсів і природокористування України.

Метою такого розробки є автоматизація обліку даних студентів, які є важливими в контексті їх навчання. Важливо, щоб система спрощувала формування документів, які забезпечує деканат. А в результаті підвищила ефективність роботи працівників.

Для такої розробки зручно використовувати об’єктно-орієнтований підхід до системного аналізу. Результати аналізу описано та змодельовано за допомогою UML.

За результатами розробки було подано тези з назвою “ІНФОРМАЦІЙНА УПРАВЛЯЮЧА СИСТЕМА “ЕЛЕКТРОННИЙ ДЕКАНАТ”” на VII

Всеукраїнській науково-практичній конференції студентів і аспірантів “Теоретичні та прикладні аспекти розробки комп’ютерних систем 2025”.

Також було подано тези з назвою “Програмне забезпечення системи з документообігу в деканаті вищого навчального закладу” на VII Всеукраїнській науково-практичній конференції студентів і аспірантів “Теоретичні та прикладні аспекти розробки комп’ютерних систем 2026” з доповіддю.

Записка кваліфікаційної роботи складається зі вступу, чотирьох розділів та висновків. Список використаних джерел та додатки винесені в кінці записки. Загальний обсяг записки – 62 сторінки основного тексту, кількість використаних джерел – 18, кількість додатків – 8, у тому числі Декларація про академічну доброчесність та використання ШІ у додатку Л.

У першому розділі здійснено системний аналіз документообігу в деканаті вищого навчального закладу на прикладі НУБіП України. Визначено процедурні та нормативні особливості ведення контингенту студентів, обліку успішності та нарахування академічних і соціальних стипендій. За допомогою інструментарію UML (діаграм прецедентів, послідовності та активності) формалізовано бізнес-процеси предметної області та побудовано математичну модель розрахунку рейтингового бала здобувачів освіти. На основі отриманих результатів сформульовано комплекс функціональних і нефункціональних вимог до проєктувального програмного забезпечення.

У другому розділі обґрунтовано вибір реляційної СУБД MySQL та інструменту проєктування MySQL Workbench. Розроблено концептуальну та логічну моделі даних у вигляді ER-діаграми, які в процесі нормалізації доведено до вимог 3NF для усунення аномалій. Описано детальну фізичну структуру інформаційної бази даних, специфікації типів полів, індексів та каскадних зв'язків між таблицями, а також реалізовано збережені процедури, тригери та транзакційні механізми забезпечення цілісності даних на рівні СУБД.

У третьому розділі описано безпосередню програмну реалізацію архітектури системи на базі фреймворку Laravel із дотриманням патерну MVC. Деталізовано компонентну структуру ПЗ, де клієнтську частину побудовано на

основі шаблонів Blade та бібліотеки Bootstrap 5, а динамічну взаємодію без перезавантаження сторінок реалізовано через асинхронні запити AJAX. Окрему увагу приділено інтеграційним шлюзам системи: автоматизованій синхронізації з реєстрами ЄДЕБО через REST API, наскрізній авторизації через Google OAuth2.0, імпорту оцінок із платформи E-Learn за допомогою токенів Laravel Sanctum та взаємодії з каталогом LDAP університету. Описано алгоритми низькорівневого формування PDF-документів європейського зразка за допомогою бібліотеки FPDF. Окрема подяка висловлюється Андрющенку Віктору Миколайовичу, Мокрієву Максиму Володимировичу та Понзелю Ярославу Юрійовичу.

У четвертому розділі представлено результати контролю якості та впровадження системи. Описано архітектуру автоматизованого тестування ПЗ на базі PHPUnit, що включає покриття ключових ендпоінтів авторизації 14-ма негативними та позитивними Feature-тестами з ізоляцією бази даних через транзакційні трейти. Наведено результати тримісячної дослідної експлуатації та апробації продукту на базі факультету під час пікових навантажень зимової та літньої екзаменаційних сесій. Обґрунтовано вимоги до апаратно-програмного забезпечення сервера й клієнта, а також описано DevOps-інфраструктуру контейнеризованого розгортання пакета на основі Docker та Docker Compose.

1 ДОКУМЕНТООБІГ В ДЕКАНАТІ ВИЩОГО НАВЧАЛЬНОГО ЗАКЛАДУ ЯК ОБ'ЄКТ РОЗРОБКИ

1.1 Аналіз документообігу в деканаті, як предметної області

Факультет – це структурний підрозділ закладу вищої освіти, що об'єднує не менш як три кафедри та/або лабораторії, які в державних і комунальних закладах вищої освіти в сукупності забезпечують підготовку не менше 200 здобувачів вищої освіти денної та дуальної форм здобуття освіти (крім факультетів вищих військових навчальних закладів, закладів вищої освіти із специфічними умовами навчання, військових навчальних підрозділів закладів вищої освіти, закладів вищої освіти фізичного виховання і спорту, закладів вищої освіти культури та мистецтва). Для вирішення поточних питань діяльності закладу вищої освіти утворюються робочі органи – ректорат, деканати, приймальна комісія, адміністративна рада тощо [3].

Деканат – це керуючий орган факультету, який координує співпрацю студентів, викладачів та адміністрації. Посада працівника деканату називається методист. Орган підпорядковується напряму декану факультету або заступнику декана з навчальної роботи і виховної роботи.

Завдання деканату полягає у видачі довідок, замовленні дипломів, веденні успішності студентів, їх навчальних планів, формуванні та друкуванні відомостей обліку успішності та атестаційних відомостей, складанні рейтингу студентів на основі їх успішності, формуванні реєстрів стипендіатів та ін.

В межах питань документообігу деканат формує такі документи:

- Довідка про навчання;
- Академічна довідка;
- Відомість обліку успішності;
- Атестаційна відомість;
- Витяг з картки навчання;
- Диплом бакалавра/магістра;
- Додаток до диплому бакалавра/магістра;
- Протокол засідання стипендіальної комісії факультету;

- Реєстр осіб з числа студентів кожної зі спеціальностей, кожної освітньої програми, кожного курсу, яким призначена стипендія рішенням стипендіальної комісії факультету;
- Індивідуальний навчальний план студента;

Також, в Національному університеті біоресурсів і природокористування України передбачено надання додаткових освітніх послуг, для оформлення яких формуються наступні документи:

- Заява з проханням надати додаткову освітню послугу з визначених заявником дисциплін;
- Додаток до заяви;
- Договір про надання додаткових освітніх послуг Національним університетом біоресурсів і природокористування України;

Великий обсяг документів та необхідність їх швидкого формування створюють значне навантаження на методистів деканату, що зумовлює потребу в розробці програмного забезпечення системи з документообігу в вищому навчальному закладі.

Детально розглянуто процедуру призначення академічних та соціальних стипендій. Кінцевою метою методиста деканату є формування реєстру осіб з числа студентів конкретної спеціальності, конкретної освітньої програми, конкретного курсу, яким призначена стипендія рішенням стипендіальної комісії факультету, для передачі такого реєстру до загальноуніверситетської системи документообігу. Сформовані реєстри будуть виступати додатками до наказу про призначення стипендії.

Для формування списків студентів, що отримуватимуть стипендію, працівнику деканату потрібно:

- Сформувати індивідуальний навчальний план студента на відповідний семестр. Для цього потрібно сформувати загальний, в якому будуть передбачені всі дисципліни. У тому числі обов’язкові, вибіркові фахові та вибіркові загальноуніверситетські дисципліни.

- Внести оцінки студентів з дисциплін, які є в кожному індивідуальному навчальному плані. А також, додаткові бали.
- Сформувати протокол засідання стипендіальної комісії факультету з рейтингом студентів. Відповідно до ст. 4 п. 5 Правил призначення академічних стипендій у Національному університеті біоресурсів і природокористування України [4], рейтинговий бал студента Rстуд визначається за 100-бальною шкалою як сума двох складових:

– середнє арифметичне значення рейтингів R усіх видів атестації за 100-бальною шкалою

− складання заліків, екзаменів, захисту курсових робіт чи проектів, звітів за навчальну чи виробничу практику

− за результатами навчання в останньому семестрі та складання екзаменаційної сесії (складає 90 % рейтингового бала студента);

– додаткові бали b за участь у науковій, науково-технічній діяльності, громадському житті, творчій та спортивній діяльності (складає 10 % рейтингового бала студента). Визначений рейтинговий бал студента Rстуд округлюється до сотих значень числа (другого знаку після коми). Рейтинговий бал студента розраховується за формулою:

$$Rстуд = \frac{R1+R2+\cdots+Rn}{n} \times 0{,}9 + b,$$

де:

Rn − рейтинги усіх видів атестацій за 100-бальною шкалою в останньому семестрі і під час складання екзаменаційної сесії;

n − кількість атестацій в останньому семестрі і під час складання екзаменаційної сесії;

b – додаткові бали (від 0,1 до 10,0) призначаються за участь у науковій, науково-технічній діяльності, громадському житті, творчій та спортивній діяльності.

- На основі протоколу засідання стипендіальної комісії потрібно сформувати реєстр осіб з числа студентів відповідного курсу, відповідного освітнього ступеня та спеціальності, яким призначена стипендія.

1.2 Моделювання предметної області

Для полегшеного розуміння всього процесу побудовано діаграму прецедентів (рис. 1.1).

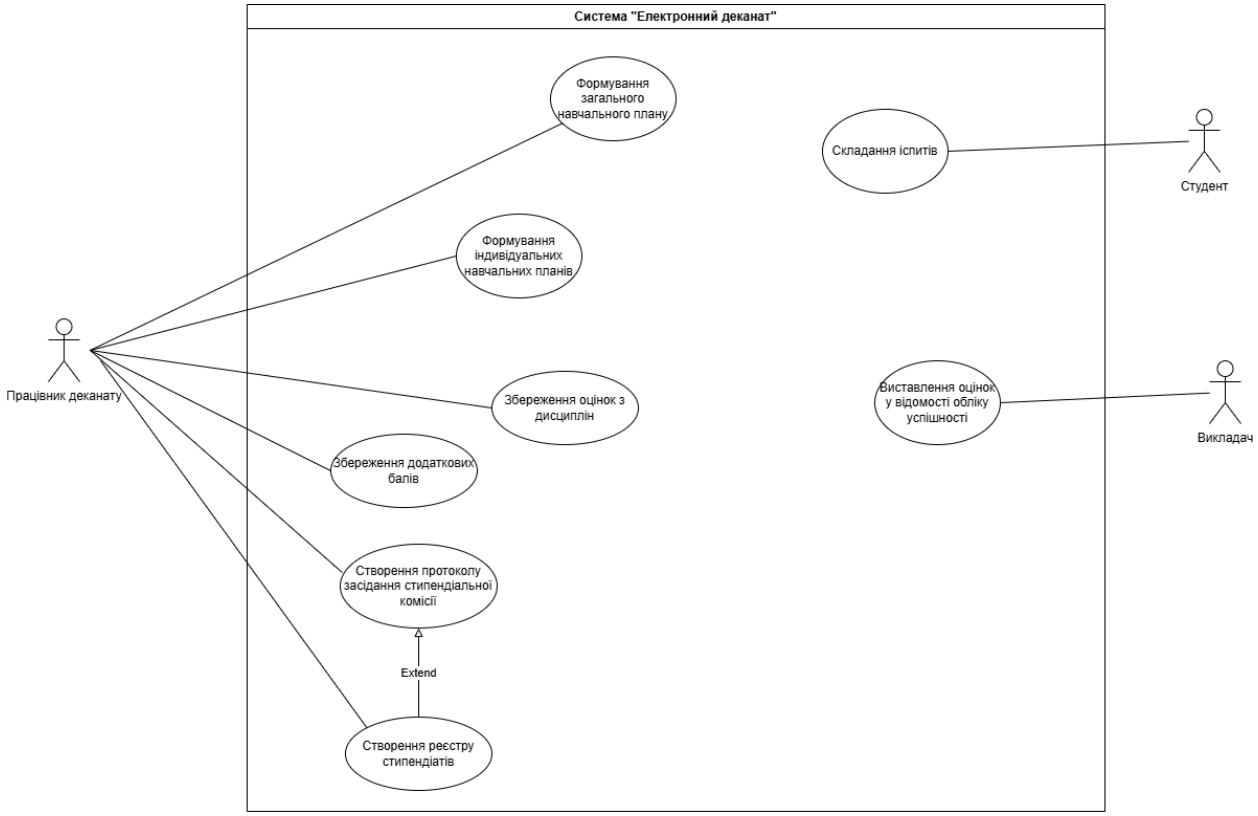


Рис. 1.1 Діаграма прецедентів

Діаграми прецедентів (use-case diagrams) ілюструють та визначають контекст і вимоги як до системи в цілому, так і до її ключових частин [5].

На діаграмі прецедентів (рис. 1.1) показано складання іспитів студентом та виставлення оцінок у відомості обліку успішності викладачем. Це процеси, які виконуються без участі методиста деканату, але методист зберігає відомості з оцінками. Це реалізовано в програмному забезпеченні. Для розуміння послідовності створення реєстру стипендіатів побудуємо діаграму послідовності (рис. 1.2).

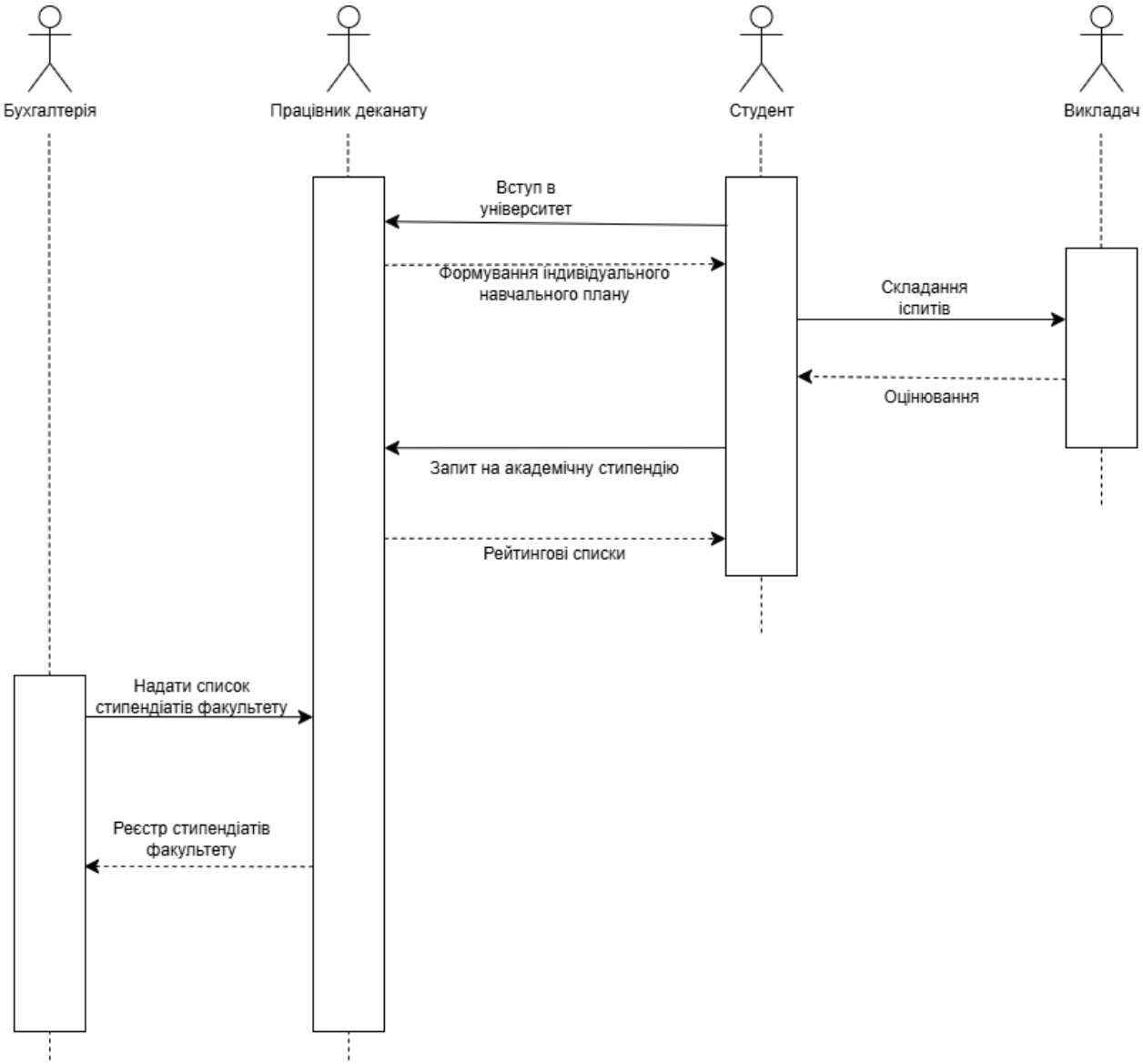


Рис. 1.2 Діаграма послідовності

Діаграма послідовності описує взаємодію, фокусуючись на послідовності повідомлень, якими обмінюються об’єкти, разом із відповідними специфікаціями подій (OccurrenceSpecifications) на лініях життя (Lifelines) [6].

На цій діаграмі було вимушено додано актора “Бухгалтерія”, оскільки, наказ для нарахування стипендії формує саме бухгалтерія. Працівники бухгалтерії не передбачаються користувачами системи документообігу в деканаті. З побудованої діаграми послідовності можна побудувати наступну діаграму активності (рис. 1.3)

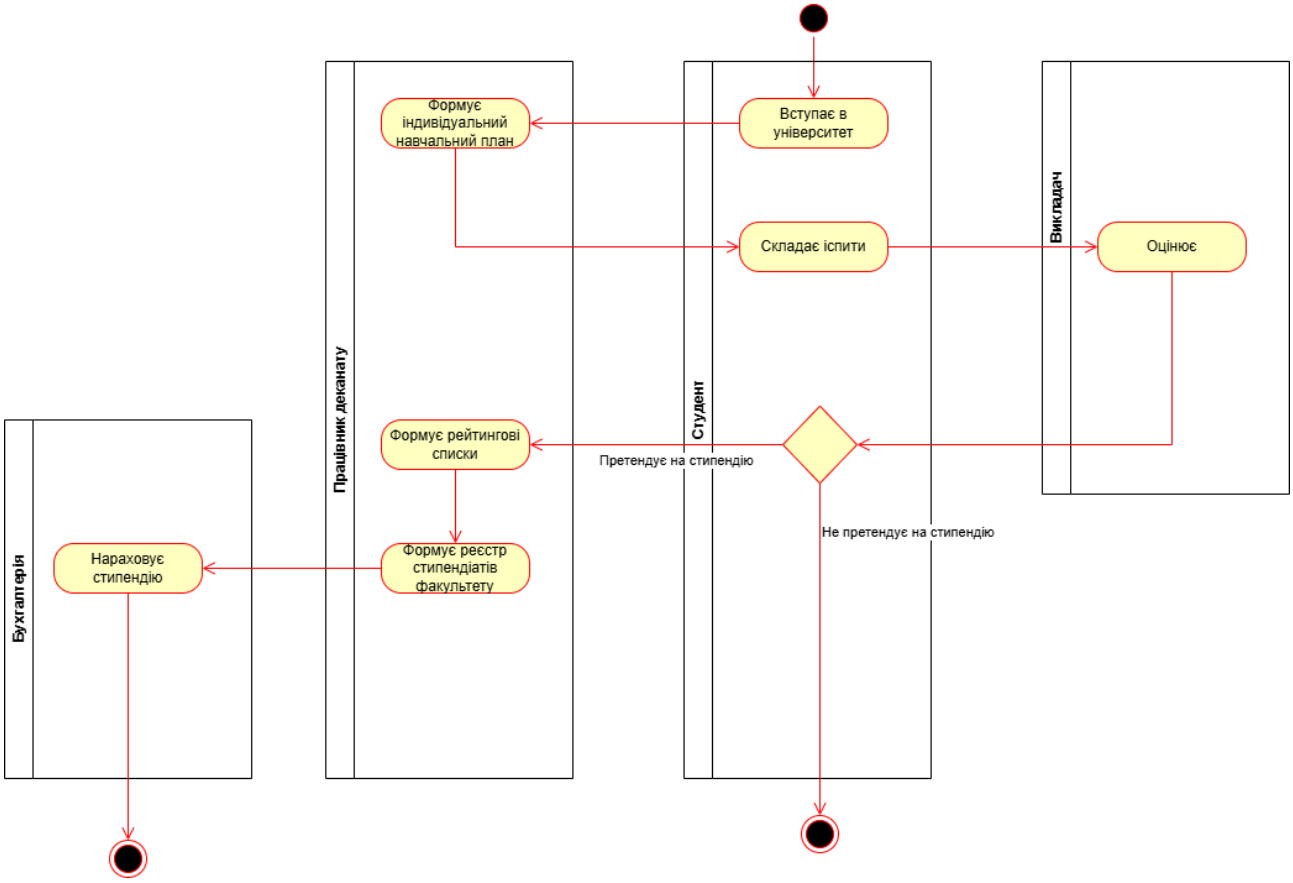


Рис 1.3 Діаграма активності

Отже, все починається зі вступу студента в університет, за його освітньою програмою складається індивідуальний навчальний план. Індивідуальний навчальний план складається з дисциплін трьох видів: обов'язкові, вибіркові фахові та вибіркові загальноуніверситетські. Згідно з правилами призначення академічних стипендій у Національному університеті біоресурсів і природокористування України рейтинг студентів складається з оцінок навчальної роботи і додаткових балів. Наразі, претендувати на академічну стипендію можуть тільки 40% студентів, що навчаються за державним замовленням. Цей рейтинг вкладається в протокол засідання стипендіальної комісії, що є основою для формування реєстру осіб з числа студентів, яким призначена стипендія.

1.3 Постановка завдання

У межах даної роботи потрібно розробити програмне забезпечення системи документообігу в деканаті вищого навчального закладу. Система

повинна підвищити ефективність деканату, полегшити та прискорити його роботу. Основним завданням є переведення ручної роботи в електронний формат.

Система створюється для ефективного збереження персональних даних, координації навчальних етапів та швидкого формування необхідних документів.

Функціональними вимогами до системи є:

- Синхронізація даних студентів, що вступили до навчального закладу, з Єдиною державною електронною базою з питань освіти
- Формування і призначення навчальних планів
- Зберігання успішності студентів
- Генерація документів, що супроводжують навчальний процес, які відповідають положенням і правилам Національного університету біоресурсів і природокористування України
- Авторизація користувачів за допомогою корпоративного Google-акаунту

Нефункціональними вимогами до системи є:

- Підтримка цілісності і захисту даних
- Продуктивність роботи
- Кроссплатформеність інтерфейсу

Вхідними даними системи є персональні дані студента, його успішність та навчальний план. Вихідними даними – документи, що стосуються навчального процесу в вищому навчальному закладі. Важливо використовувати безкоштовні інструменти розробки, щоб не обтяжувати бюджет вищого навчального закладу.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір інструментарію для створення і роботи з інформаційним забезпеченням системи

Для зберігання даних використано реляційну систему управління базою даних MySQL. Цю СУБД було вибрано керівництвом проекту.

СУБД MySQL є безкоштовним продуктом із відкритим кодом, що дозволяє знизити витрати на впровадження системи в вищому навчальному закладі. Нижче наведено систематизований аналіз ключових переваг MySQL, що обґрунтовують її доцільність для застосування у високонавантажених вебзастосунках.

Транзакційна надійність та архітектурна гнучкість. Фундаментальною перевагою MySQL є її мультидвигунова архітектура (Pluggable Storage Engine Architecture), яка дозволяє оптимізувати обробку даних на рівні окремих таблиць залежно від специфіки бізнес-логіки.

Висока продуктивність операцій читання та оптимізація запитів. MySQL демонструє високу детерміновану швидкість під час виконання операцій вибірки даних (SELECT), що робить її еталонним для рішенням систем документообігу із високою інтенсивністю читання.

Оскільки, в деканатах Національного університету біоресурсів і природокористування України працює понад 100 людей, найпростішим рішенням буде використати технології веб-програмування. Така технологія дозволяє користування застосунком за допомогою веб-браузера, без зайвого навантаження на клієнтський пристрій. Також технологія дозволяє забезпечувати кроссплатформеність.

Для вибору мови програмування серверної частини веб-застосунку порівняємо дві мови програмування PHP та ASP .NET.

PHP має менший поріг входу для новачка у веб-розробці завдяки гнучкості програмного коду. Динамічна типізація значно полегшує роботу. В свою чергу ASP .NET пропонує сувору статичну типізацію і жорстке дотримання архітектури. Також, ASP .NET має велику, але закриту екосистему. PHP має

величезну екосистему composer та packagist. Дослідження порталу Wezoom, що складається на основі опитувань на StackOverflow та аналізі репозиторіїв Github вважає, що в 2026 році найкращі мови програмування серверної частини веб-застосунку це PHP та Ruby [11]. Отже вибір очевидний, PHP є добрим варіантом для розробки програмного забезпечення системи документообігу в вищому навчальному закладі.

Для спрощення роботи вибрано фреймворк для розробки. Складено порівняльну таблицю (Таблиця 2.1) з популярними фреймворками для розробки мовою PHP.

Таблиця 2.1

Порівняльний аналіз фреймворків, що працюють з мовою програмування PHP

Критерій	**WordPress**	**Symfony**	**Laravel**
Тип	CMS (система керування контентом)	Компонентний фреймворк	Full-stack фреймворк
Призначення	Блоги, прості сайти, SEO-ресурси	Високонавантажені Enterprise-системи	Сучасні SaaS, сервіси, API, CRM
Швидкість розробки	Дуже висока (якщо брати готове)	Нижча (багато конфігурацій)	Висока (баланс готового та кастомного)
Гнучкість	Низька (обмежена архітектурою CMS)	Максимальна	Висока
Крива навчання	Легко почати, важко "чисто" програмувати	Дуже висока (складна архітектура)	Помірна (дуже зрозуміла документація)

Було використано фреймворк Laravel, який забезпечує реалізацію архітектури MVC.

Архітектура MVC в межах фреймворку Laravel полягає у розділенні системи на три основні компоненти: модель, контролер, представлення. Модель це клас, який відтворює поля таблиці або представлення бази даних в свої атрибути. Оскільки, використовуються веб-технології, бізнес-логіка застосунку впирається в HTTP-запити.

Контролер в Laravel відповідає за обробку таких запитів. Представлення зазвичай містять HTML-код і відповідають за інтерфейс користувача. Фреймворк має власний двигун шаблонів blade, який створений на основі двигуна .NET Razor. Узгодження, що в документації називаються директивами мають префікс "@". Директиви використовуються для визначення структур керування, наслідування і будь-яких користувацьких функцій. Для можливості відправки HTTP-запитів із веб-сторінки застосовується система роутингу. За допомогою цієї системи формується HTTP посилання до конкретного контролера. Роутинг дозволяє формувати групи посилань з контролерами, які знаходяться в одному просторі імен.

Laravel підтримує розробку проміжного програмного забезпечення (middleware). Концепція проміжного програмного забезпечення полягає в тому, що будь-який запис в програмному застосунку проходить через кожен шар такого ПЗ. Зазвичай проміжне програмне забезпечення не вважається складовою логіки застосунку і розробляється таким чином, щоб його можна було застосовувати до будь-якого застоснку [10].

Фреймворк має вбудовані механізми захисту від типових вразливостей (SQL-ін'єкцій, CSRF-атак та XSS), що є критичним при роботі з персональними даними.

Для генерації pdf-документів використаємо клас FPDF, що працює з PHP. FPDF – це клас PHP, який дозволяє генерувати PDF-файли за допомогою чистого PHP, тобто без використання бібліотеки PDFlib. В пакеті fpdf/fpdf, крім самого класу, також є утиліта, яка перетворює .ttf файли шрифтів у .php файл, і дозволяє

його використання класом. В деяких документах, як от в додатках до дипломів, за стандартом МОН вимагається нестандартна нумерація сторінок, а клас це не підтримує. Тому, було додано атрибут класу $hardPageNumber, що буде приймати номер, який буде заданий при створенні нової сторінки. Також додано атрибут $disablePageNumber, який вимикає нумерацію сторінок.

2.2 Логічна модель у вигляді ER-діаграми

Діаграма «сутність-зв'язок» (ER-діаграма або ERD) — це візуальне представлення того, як елементи в базі даних пов'язані між собою. ER-діаграми є спеціалізованим типом блок-схем, що відображають типи відносин між різними сутностями всередині системи. Вони використовують визначений набір символів, зокрема прямокутники, овали та ромби, і з'єднують їх лініями зв'язку [9].

Основною сутністю системи документообігу є студент. Оскільки одна і та сама людина може навчатись на різних освітніх програмах, розіб'ємо на сутність PersonEducationCard і Person. Ці сутності з'єднано зв'язком багато-до-одного. Сутність персона містить персональні дані студента (ІПН, ПІБ, ID персони в ЄДЕБО, дані документа, що посвідчує особу, контактні дані і пільгу, якщо вона є). Оскільки, є визначений список пільг, для студентів, сутність Previlege повинна існувати в системі. Вона міститиме дані про вид пільги, та передбачену соціальну стипендію. Сутність PersonEducationCard міститиме дані про навчання студента на конкретній освітній програмі, основу вступу, ID навчальної картки в ЄДЕБО. Студенти об'єднуються в академічні групи, тому справедливим є існування сутності StudyGroup.

В Національному університеті біоресурсів і природокористування України станом на 2026 рік існує 16 факультетів та ННІ. Сутність Faculty зберігатиме назву факультету, ПІБ декана факультету/директора ННІ. Кожен Faculty має свої спеціальності, сутність Speciality пов'язана з Faculty зв'язком багато-до-одного. Спеціальності розділяються на освітні програми. Кожна Speciality може містити 1 і більше StudyProgram.

Для кожної освітньої програми кожного року вступу формується навчальний план. Сутність Curriculum містить дисципліни навчального плану і прив’язана до PersonEducationCard.

З кожної з дисциплін студент складає іспити, заліки, пише курсові роботи. Це оцінює його знання. Сутність StudentAssessment містить дані про успішність студента за тією чи іншою дисципліною.

З аналізу предметної області, відомо, що для нарахування стипендії формується протокол засідання стипендіальної комісії з рейтингом студентів. Сутність Protocol містить дані про сам протокол, а StudentProtocol містить рейтингові списки студентів, що відповідають кожному протоколу.

На рис. 2.1 показано діаграму сутність-зв’язок, що описує логічну модель даних.

2.3 Розробка структури інформаційної бази

Реляційна система управління базами даних передбачає розміщення даних в таблицях, які пов’язані зв’язком.

З логічної моделі можна виділити такі таблиці:

Previlege – таблиця, що зберігає дані про існуючі пільги. Поля таблиці:

- idPrevilege – первинний ключ таблиці. Унікальний ідентифікатор пільги;
- Name – назва пільги. Не може бути пустим;
- Sum – сума соціальної стипендії, яка виплачується за цією пільгою;

Person – таблиця, що зберігає персональні дані студентів. Поля таблиці:

- idPerson – первинний ключ таблиці. Унікальний ідентифікатор персони;
- idPrevilege – зовнішній ключ таблиці Previlege. Може бути пустим, якщо пільгп не призначена. Цей зв’язок має кратність багато-до-багатьох, тому потребуватиме проміжної таблиці на фізичному рівні;
- PersonCode – унікальний ідентифікатор персони в Єдиній державній електронній базі з питань освіти. Не може бути пустим;

- Citizen – поле, що визначає чи є студент іноземцем. Це важливо в питанні формування додатка до диплому, оскільки, крім документа про попередню освіту, потрібні ще і дані свідоцтва про визнання документу Міністерством освіти і науки України.
- LastName – поле прізвища особи;
- FirstName – поле імені особи;
- MiddleName – поле по-батькові особи;
- BirthDate – дата народження;
- LastName – поле прізвища англійською особи;
- FirstName – поле імені англійською особи;
- MiddleName – поле по-батькові англійською особи;
- Sex – поле, що вказує стать;
- PersonDocumentType – тип документа, що посвідчує особу;
- PersonDocumentNumber – номер документа, що посвідчує особу;
- PersonDocumentSeries – серія документа, що посвідчує особу (за наявності);
- PersonDocumentIssuedBy – орган, що видав документ;
- PersonDocumentIssuedDate – дата видачі документа, що посвідчує особу;
- IPN – ідентифікаційний податковий номер особи;
- PhoneNumber – номер телефону особи;
- Email – персональна електронна пошта особи;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

StudyGroup – таблиця, що групує студентів за академічними групами. Поля таблиці:

- idStudyGroup – первинний ключ таблиці. Унікальний ідентифікатор академічної групи;

- StudyGroupName – назва, або код академічної групи;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

Faculty – таблиця, що зберігає дані про факультети та ННІ вищого навчального закладу. Поля таблиці:

- idFaculty – первинний ключ таблиці. Унікальний ідентифікатор факультету;
- FacultyName – назва факультету;
- FacultyNameEn – назва факультету англійською;
- DeanLastName – прізвище декана факультету. Це і наступні два поля потрібні для створення поля підпису в деяких документах;
- DeanFirstName – ім’я декана факультету;
- DeanMiddleName – по-батькові декана факультету;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;

Поле UserEdited_by не потрібне в цій таблиці, оскільки факультети вносить адміністратор.

Speciality – таблиця, що зберігає дані спеціальностей. Поля таблиці:

- idSpeciality – первинний ключ таблиці. Унікальний ідентифікатор спеціальності;
- idFaculty – зовнішній ключ з таблиці Faculty. Кожен факультет має свої спеціальності. Кратність зв’язку багато-до-одного.
- SpecialityName – назва спеціальності;
- SpecialityNameEn – назва спеціальності англійською мовою;
- SpecialityCode – код спеціальності;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;

Поле UserEdited_by не потрібне в цій таблиці, оскільки спеціальності вносить адміністратор.

StudyProgram – таблиця, що зберігає дані про освітні програми. Поля таблиці:

- idStudyProgram – первинний ключ таблиці. Унікальний ідентифікатор освітньої програми в системі.
- idSpeciality – зовнішній ключ з таблиці Speciality. Деякі спеціальності розподіляються на кілька освітніх програм. Кратність зв’язку багато-до-одного;
- StudyProgramName – назва освітньої програми;
- StudyProgramNameEn – назва освітньої програми англійською мовою;
- StudyProgramCode – код освітньої програми в Єдиній державній електронній базі з питань освіти;
- HigherEducation – освітній ступінь, що здобуває студент після закінчення навчання за освітньою програмою.
- Accreditation – дата, номер та орган, що видав сертифікат акредитації освітньої програми;
- AccreditationEn – те саме, перекладене англійською мовою;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;

Discipline – таблиця, що зберігає список дисциплін, що викладаються в вищому навчальному закладі. Поля таблиці:

- idDiscipline – первинний ключ таблиці. Унікальний ідентифікатор дисципліни;
- DisciplineName – назва дисципліни;
- DisciplineNameEn – назва дисципліни англійською мовою;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;

- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

Curriculum – таблиця, що зберігає дані про навчальні плани. Поля таблиці:

- idCurriculum – первинний ключ таблиці. Унікальний ідентифікатор навчального плану;
- idDiscipline – зовнішній ключ з таблиці Discipline, що є в навчальному плані. Кратність зв'язку багато-до-багатьох. Кожна дисципліна може бути в кількох навчальних планах, а в кожному навчальному плані є багато дисциплін. Потребує проміжної таблиці на фізичному рівні;
- idStudyProgram – зовнішній ключ з таблиці StudyProgram. Кожен навчальний план складається окремо для кожної освітньої програми.
- YearEntry – рік вступу студентів, що навчатимуться за цим навчальним планом;
- EducationTime – загальний час навчання студентів на цьому навчальному плані словами. Наприклад: 3 роки 10 місяців;
- EducationTimeEn – те саме, перекладене англійською мовою;
- BasedOn – основа вступу, для навчання на цьому навчальному плані;
- HigherEducation – освітній ступінь студентів, що будуть навчатися за цим навчальним планом;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

PersonEducationCard – таблиця, що зберігає дані про навчальні картки студентів. Поля таблиці:

- idPersonEducationCard – первинний ключ таблиці. Унікальний ідентифікатор навчальної картки;
- idPerson – зовнішній ключ з таблиці Person. Кратність зв'язку багато-до-одного. Кілька карток можуть стосуватися однієї особи;

- EducationCardCode – код навчання в Єдиній державній електронній базі з питань освіти;
- idStudyGroup – зовнішній ключ з таблиці StudyGroup. Кожен студент є в академічній групі своєї освітньої програми.
- idCurriculum – зовнішній ключ з таблиці Curriculum. Кожен студент навчається на конкретному навчальному плані, але на одному навчальному плані може бути багато студентів. Кратність зв’язку багато-до-одного;
- CompetitionScore – конкурсний бал вступу. На його основі нараховується стипендія в перший семестр навчання;
- PaymentType – тип фінансування навчання. На стипендію претендують тільки ті студенти, що навчаються за державним замовленням;
- EducationForm – форма навчання. Наразі в Національному університеті є три форми навчання: денна, заочна та дистанційна;
- EducationDateBegin – дата початку навчання студента;
- EducationDateEnd – дата кінця навчання студента;
- EducationBase – тип документа, на основі якого студент вступив до вищого навчального закладу;
- EducationDocumentSeries – серія документа, що був основою вступу;
- EducationDocumentNumber – номер документа, що був основою вступу;
- EducationDocumentIssuedBy – орган, що видав такий документ;
- EducationDocumentIssuedByEn – те саме, перекладене англійською мовою;
- EducationDocumentIssuedDate – дата видачі документа, що був основою для вступу;

- HigherEducation – освітній ступінь, що здобуває студент;

- StudentStatus – статус навчання студента у вищому навчальному закладі. Такими статусами є: навчається, закінчив навчання, відрахований, в академічній відпустці;
- Email – корпоративна електронна пошта студента. Важливо для інтеграції з електронним кабінетом студента;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

StudentAssessment – таблиця, що зберігає дані про академічну успішність студента. Поля таблиці:

- idStudentAssessment – первинний ключ таблиці. Унікальний ідентифікатор оцінки;
- idDiscipline – зовнішній ключ з таблиці Discipline. Одна кілька оцінок може бути з однієї і тієї ж дисципліни. Кратність зв’язку багато-до-одного;
- idPersonEducationCard – зовнішній ключ з таблиці PersonEducationCard. Тільки одна оцінка може стосуватися одного студента, але один студент має багато оцінок. Кратність зв’язку багато-до-одного;
- Value – кількість балів;
- NumberExaminationNote – номер відомості обліку успішності, в якій було поставлено цю оцінку;
- Attempt – номер спроби складання іспиту, заліку чи підготовки курсової роботи. Студент, що не склав іспит, залік або не підготував курсову роботу з першої спроби не може претендувати на стипендію;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

Protocol – таблиця, що зберігає дані про протоколи засідань стипендіальної комісії. Поля таблиці:

- idProtocol – первинний ключ таблиці. Унікальний ідентифікатор протоколу;
- StudyGroups – перелік ідентифікаторів академічних груп, які вибрані для цього протоколу;
- Description – опис протоколу засідання стипендіальної комісії. Наприклад: 4 курсу ОС "Бакалавр";
- DateProtocol – дата протоколу;
- NumberProtocol – номер протоколу;
- DateOrder – дата наказу, до якого прив’яжеться реєстр стипендіатів, що сформований на основі цього протоколу;
- NumberOrder – номер цього ж наказу;
- Semester – семестр, за результатами якого формується протокол;
- AcademicYear – навчальний рік, за результатами якого формується протокол;
- IsByCB – поле, що визначає, чи протокол сформований на основі конкрусного балу вступу;
- DatePayStart – дата початку виплат стипендії;
- DatePayEnd – дата кінця виплат стипендії;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

StudentProtocol – таблиця, що зберігає рейтинг студентів, які є у створеному протоколі і вид стипендії, що виплачується. Поля таблиці:

- idStudentProtocol – первинний ключ таблиці. Унікальний ідентифікатор запису в рейтингу;
- idProtocol – зовнішній ключ з таблиці Protocol. Визначає, до якого протоколу стосується даний запис. Кратність зв’язку один-до-багатьох;

- idPersonEducationCard – зовнішній ключ з таблиці PersonEducationCard. Визначає, який студент в даному записі. Кратність зв’язку багато-до-одного;
- numberInorder – порядковий номер студента в рейтингу;
- ZagalRating – загальний рейтинговий бал студента, за яким визначається його місце в рейтингу, якщо рейтинг формується не на основі першого семестру навчання;
- NavchRating – рейтинг навчальної роботи. Частка з формули 1.1.1;
- DodatRating – рейтинг додаткової роботи. Змінна b з формули 1.1.1;
- ScholarshipType – тип стипендії. Може бути соціальна або академічна, якщо студент претендує;
- AcadScholarshipType – тип академічної стипендії, якщо призначена. Може бути звичайна або підвищена;
- HasScholarship – поле, яке визначає, чи взагалі студент претендує на стипендію;
- Note – примітка. В неї записуватимемо, чи мав або має студент академічну заборгованість. Академічна заборгованість означає, що студент не отримав оцінку з дисципліни за першу спробу складання;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

Спроєктована база даних відповідає другій нормальній формі. Кожна з таблиць є атомарною і усі неключові атрибути повністю залежать від первинного ключа. Щоб досягти третьої нормальної форми потрібно в таблиці StudentProtocol прибрати поле ZagalRating, оскільки, його значення складається з суми значень полів NavchRating та DodatRating.

2.3 Схема та фізична структура даних

Для можливості фізичної реалізації бази даних в СУБД MySQL створено такі проміжні таблиці: CurriculumDiscipline – таблиця, що зв’язує Discipline та Curriculum, та PersonPrevilege – таблиця, що зв’язує Previlege та Persons.

Оскільки, важливо розділяти дисципліни за семестрами та навчальними роками в залежності від навчального плану, таблиця CurriculumDiscipline складається з таких полів:

- idCurriculumDiscipline – первинний ключ таблиці. Унікальний ідентифікатор дисципліни навчального плану;
- idDiscipline – зовнішній ключ з таблиці Discipline. Одна дисципліна може міститися в багатьох навчальних планах. Кратність зв’язку багато-до-одного;
- idCurriculum – зовнішній ключ з таблиці Curriculum. Навчальний план складається з багатьох дисциплін навчального плану. Кратність зв’язку багато-до-одного;
- idEcurse – поле для інтеграції системи з навчальним порталом. Для багатьох дисциплін є відповідний електронний курс на порталі;
- AcademicYear – навчальний рік, в якому викладається/викладатиметься дисципліна;
- Semester – семестр, в якому викладається/викладатиметься дисципліна. Може бути тільки 1 або 2;
- Mandatory – вид дисципліни. Може бути: обов’язкова, вибіркова фахова або вибіркова загальноуніверситетська;
- Lectures – кількість годин, виділених на лекції;
- LaboratoryWork – кількість годин, виділених на лабораторні роботи;
- PracticalLessons – кількість годин, виділених на практичні заняття;
- IndependentWork – кількість годин, виділених на самостійну роботу;
- EducationalPracticeHours – кількість годин, виділених на навчальну практику;

- ManufacturingPractice – кількість годин, виділених на виробничу практику;
- AmountWeeks – кількість тижнів, в яких викладається дисципліна;
- WeeklyHours – кількість годин на один тиждень;
- ExaminationForm – форма контролю знань. Може бути: іспит, залік, курсова робота;
- IsInAddition – поле, що визначає, чи є дисципліна в додатку до диплома. Якщо в навчальному одна і та ж дисципліна викладається кілька семестрів, в додатку до диплома буде тільки остання;
- created_at – дата створення запису в таблиці;
- updated_at – дата оновлення запису в таблиці;
- UserEdited_by – користувач, що вніс останні зміни в поточний запис;

Таблиця PersonPrevileges має зовнішні ключі з таблиць Previlege і Person, та об’єднує їх під одним ключовим полем.

Оскільки, була створена таблиця CurriculumDiscipline, успішність можна записувати по конкретній дисципліні навчального плану студента. Зв’язок StudentAssessment та Discipline змінено на зв’язок між StudentAssessment та CurriculumDiscipline. Для досягнення третьої нормальної форми було прибрано поле ZagalRating із таблиці StudentProtocol, що існувало в логічній моделі. Після цих змін, спроєктована схема бази даних готова до реалізації на фізичному рівні. У додатку Ж зображено схему даних, що приведена до третьої нормальної форми та готова до реалізації на фізичному рівні.

Для адміністрування бази даних використано веб-інтерфейс phpMyAdmin. Проте, оскільки phpMyAdmin погано працює із завантаженням великих масивів даних, для резервного копіювання використано MySQL Workbench. Цей інструмент дозволяє швидко експортувати та імпортувати дані. Резервне копіювання здійснюється раз на місяць. Адміністратор експортує дані з кожної таблиці у окремі .sql файли, що містять запит видалення таблиці при існуванні,

створення нової таблиці з визначеною структурою, а також запит вставлення експортованих даних.

Фреймворк Laravel для розробки бази даних використовує міграції та фасад Schema. Але перед цим обов'язково потрібно вказати в конфігураційному файлі database.php у масиві підключень. В розробленому програмному забезпеченні елемент масиву 'default' призначений для підключення до бази даних за замовчуванням, проте їх можна створювати безліч. Моделі та міграції, якщо не вказувати конкретне підключення, працюватимуть саме з підключенням 'default'. Для налаштування підключення до бази даних використовуються обов'язкові параметри. До таких параметрів належать:

- Driver – безпосередньо СУБД, що використовується, у даному випадку mysql;
- Host – зазвичай, IP адреса сервера, на якому знаходиться база даних. Оскільки у розробленому ПЗ, база даних розгорнута локально на сервері, значенням цього параметра є localhost;
- Port – зазвичай стандартний порт бази даних 3306;
- Database – назва бази даних;
- Username – користувач бази даних. Для кожного застосунку Laravel створюється окремий користувач бази даних, яким користується він користується для підключення. Якщо це база даних за замовчуванням, то зазвичай надаються всі права крім створення нових користувачів і надання ролей;
- Password – пароль користувача бази даних;

Також є необов'язкові поля, що стосуються кодування символів, SSL-сертифікату та ін.

Кожен окремий клас міграції містить два методи up() та down(). В методі up() визначається все, що повинен зробити клас при запуску мігрування. Метод down() використовується для визначення дій при відкаті мігрування. Міграції використовують для створення таблиць та додавання полів у них, створення

зв’язків між таблицями, їх індексація а також створення представлень, тригерів і т.д. В розробленому програмному забезпеченні міграції використано для створення таблиць та зв’язків між ними при запуску та їх видалення при відкаті. Для всіх цих операцій використовується фасад Schema, що є базовим фасадом фреймворку Laravel.

Фасад Schema в Laravel клас, який абстрагує взаємодію з системою управління базами даних (СУБД) на рівні опису структури даних (Data Definition Language – DDL). В його основі лежить побудовник запитів DDL, що надає розробнику незалежний від конкретної СУБД інструментарій для створення, зміни та видалення таблиць, індексів та зв’язків.

Приклад створення таблиці studyprogram за допомогою класу міграції наведено в додатку А.

2.4 Робота з базою даних

Роботу з базою даних в Laravel забезпечує Eloquent ORM.

Eloquent ORM (Object-Relational Mapping) – це вбудована в фреймворк Laravel об'єктно-реляційна система відображення, яка реалізує архітектурний шаблон Active Record.

Для коректної роботи системи створюються класи моделей, що визначають таблицю, до якої прив’язана, та її структуру. Приклад моделі studyprogram наведено у додатку Б.

Реалізація шаблону Active Record означає, що для створення нового запису в таблиці studyprogram достатньо створити новий об’єкт цієї моделі, задати його атрибути та викликати метод save(), який є його деструктором, а атрибути впишуться в базу даних у поля, що відповідають назві атрибута. Щоб змінити дані запису, достатньо викликати вбудований конструктор find(), задавши параметром ідентифікатор запиту, змінити атрибут класу та викликати метод save().

Також ця об’єктно-реляційна система має побудовник запитів. Що значно спрощує роботу з даними. Для задання умови, замість WHERE мови SQL можна

викликати метод where() моделі та задати потрібні параметри, що є повним аналогом. Так само це працює і з іншими SQL командами. Якщо потрібно сконструювати колекцію об’єктів, використовується метод get().

Перевагами використання Eloquent ORM також є методи firstOrCreate(), createOrIgnore() та ін. Метод first() вибирає перший об’єкт із колекції. Якщо такого запису немає firstOrCreate() його створює, якщо є, повертає його. А, якщо створення виконується методом createOrIgnore(), то у разі помилки цілісності даних, запис не створиться.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір інструментарію для розробки клієнтської частини програмного забезпечення

У розділі 2.1 аргументовано вибір фреймворку Laravel. Він забезпечує роботу шаблонізатора blade. Цей шаблонізатор забезпечує наслідування між представленнями. Тому в розробленому ПЗ, для виключення дублювання коду користувацького інтерфейсу, було виділено три основні елементи: навігаційне меню зверху, бокове меню і частину динамічного контенту. Ці елементи будуть частиною основного blade-шаблону, що міститиме в коді підключення потрібних бібліотек, файлів стилів та скриптів. За результатами такого аналізу побудовано діаграму компонентів, що на рис. 3.1.

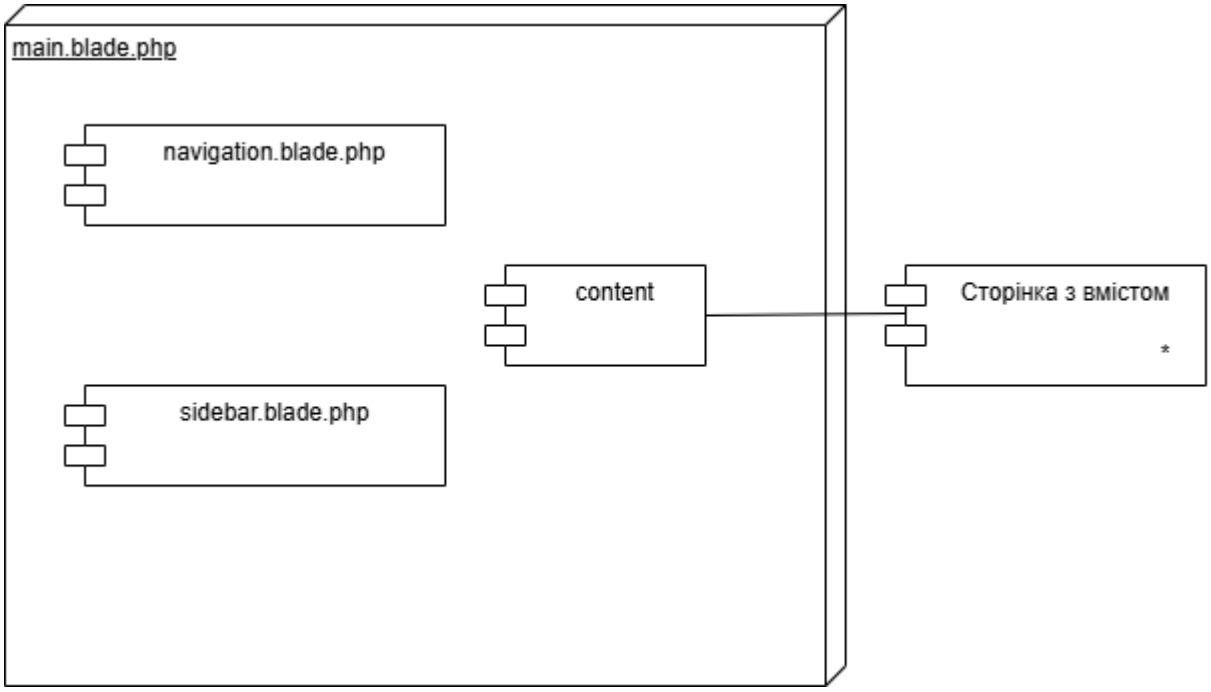


Рис. 3.1 Схема ієрархії та наслідування Blade-шаблонів користувацького інтерфейсу

Для формування інтерфейсу користувача вибрано мову гіпертекстової розмітки HTML. За допомогою цієї мови можна сформувати “скелет” веб-сторінок застосунку. Вона дозволяє використання спеціальних тегів, що робить код зрозумілим для браузерів і допомагає правильно відображати складні форми з даними. У цій системі CSS (Cascading Style Sheets) відповідає за візуальне

оформлення та зручність користування. Стилі дозволять створити єдину систему корпоративного стилю для всіх сторінок системи, а бібліотека bootstrap полегшить вибір кольорів та вигляду форм для заповнення. Використання JavaScript дозволяє виконувати асинхронні запити, що дозволяє оновлювати частину сторінки без повного перезавантаження всієї сторінки, що значно пришвидшить роботу в умовах великого обсягу даних.

3.2 Абстракції предметної області

Абстракція в об’єктно-орієнтованому програмуванні – це виділення загальних характеристик об’єктів, їхніх властивостей і методів, при ігноруванні деталей реалізації. Цей процес дає змогу створювати простіші моделі складних систем, які містять тільки необхідні елементи для розв’язання задачі. Наприклад, ми можемо створювати моделі комп’ютерів, турбін або людського тіла, без згадки окремих деталей їхньої структури та функціонування [7].

З проведеного аналізу та моделювання предметної області в межах документів для нарахування стипендії виділено такі абстракції:

- Працівник деканату – в межах програмного забезпечення є користувачем;
- Студент – абстракція, навколо якої будується все програмне забезпечення;
- Навчальний план – список і порядок дисциплін, за яким навчається студент;
- Протокол засідання стипендіальної комісії – документ, що містить рейтинговий список і є основою для створення реєстру стипендіатів;
- Реєстр стипендіатів – документ, що є підставою для нарахування стипендії;

Абстракції, їх властивості та обов’язки показано на рис. 3.2.

Рис. 3.2 Абстракції

3.3 Аналіз інтеграцій програмного забезпечення системи документообігу в деканаті вищого навчального закладу

Визначено асоціації ключових об’єктів системи, що взаємодіють для формування відомості обліку успішності, використовуючи діаграму класів (рис. 3.3).

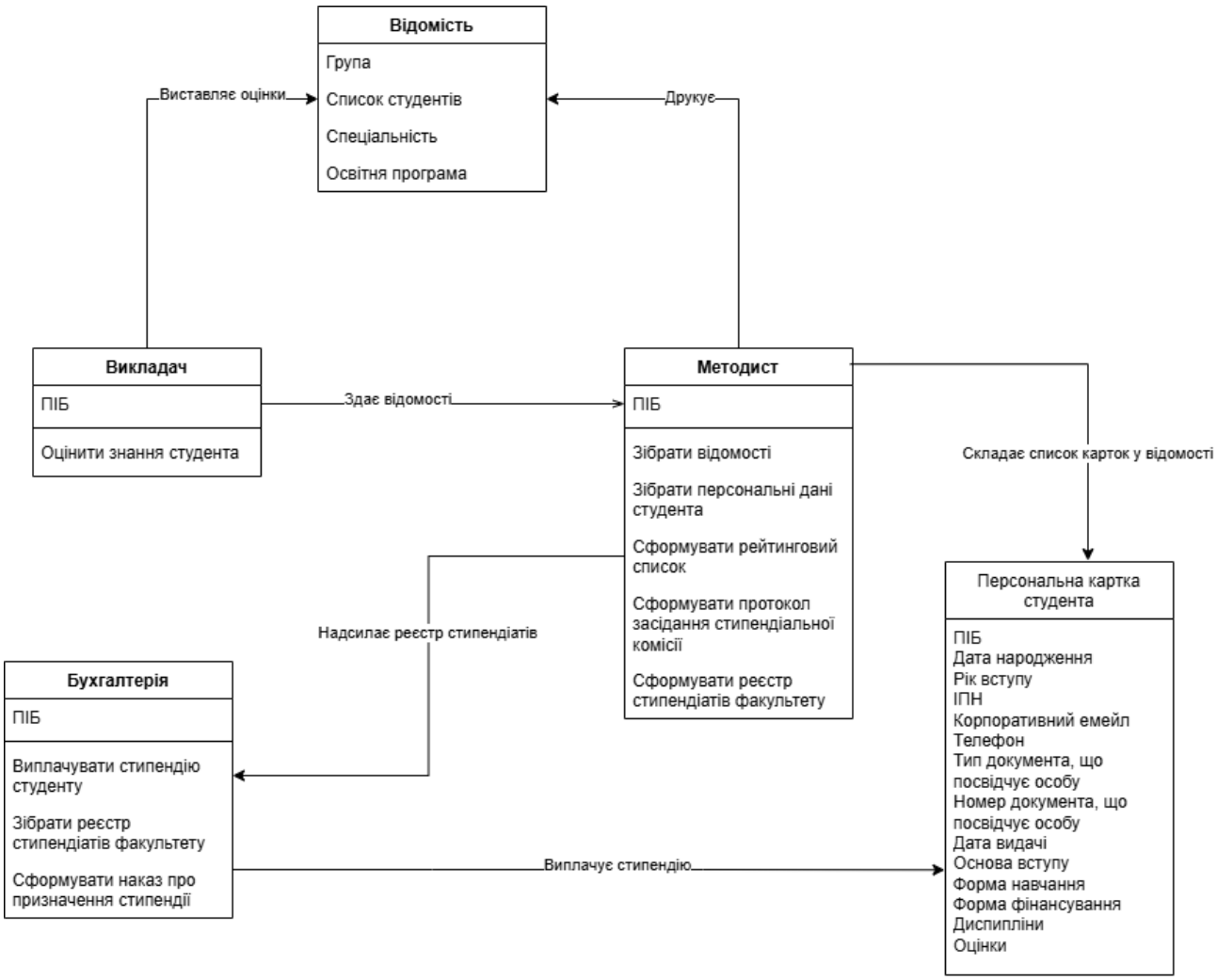


Рис. 3.3 Діаграма класів

Компонентний склад та специфікація класів. Модель складається з п'яти взаємопов'язаних класів, які архітектурно поділяються на дві категорії: активні сутності (ролі/актори системи) та пасивні сутності (інформаційні структури/документи).

А. Класи операційного та адміністративного персоналу (Актори)

1. Викладач: Об'єкт предметної області, що ініціює внесення первинних академічних даних.
    - Атрибути: ПІБ (ідентифікатор суб'єкта).
    - Операції: Оцінити знання студента (метод, що запускає процес атестації).
2. Методист: Центральний керуючий клас системи, який консолідує потоки документів у деканаті.

- Атрибути: ПІБ.
- Операції: Реалізує бізнес-функції збору відомостей та персональних даних, генерації рейтингових списків, протоколювання засідань стипендіальної комісії та формування підсумкових реєстрів.

3. Бухгалтерія: Сервісний клас, що відповідає за фінансово-економічний супровід студентського контингенту.
   - Атрибути: ПІБ.
   - Операції: Нарахування стипендій, агрегація факультетських реєстрів та видання наказів про призначення виплат.

Б. Інформаційні класи

1. Відомість: Документ суворої звітності, що відображає зріз успішності. Не містить власних методів, функціонуючи як структура даних із атрибутами: Група, Список студентів, Спеціальність, Освітня програма.
2. Персональна картка студента: Реляційне ядро моделі, що містить вичерпний масив атрибутів профілю здобувача вищої освіти:
   - Анкетні та контактні дані: ПІБ, Дата народження, Рік вступу, ІПН, Корпоративний емейл, Телефон, Дані документа, що посвідчує особу.
   - Академічні та фінансові метрики: Основа вступу, Форма навчання, Форма фінансування, Дисципліни, Оцінки.

Семантика зв'язків та функціональні цикли. Зв'язки між класами реалізовані за допомогою спрямованих асоціацій. Написи на лініях зв'язку визначають семантичні ролі та напрямки передачі керування або даних у системі. Модель чітко описує два основні життєві цикли документообігу деканату:

А. Цикл обліку успішності (Атестаційний цикл)

- Клас Викладач здійснює запис у зазначену сутність Відомість (роль: «Виставляє оцінки»).
- Заповнена Відомість передається від класу Викладач до класу Методист (роль: «Здає відомості»), що фіксує закриття відомості в деканаті.

- Клас Методист має зворотний безпосередній вплив на сутність Відомість через операцію видачі або друку (роль: «Друкує»).

Б. Стипендіальний та обліковий цикл

- Процес адміністрування даних починається з того, що Методист модифікує Персональну картку студента (роль: «Складає список карток у відомості»).
- На основі накопичених оцінок у картках студентів Методист генерує реєстр і передає керування класу Бухгалтерія (роль: «Надсилає реєстр стипендіатів»).
- Клас Бухгалтерія закриває фінансовий цикл, взаємодіючи з Персональною карткою студента (роль: «Виплачує стипендію»), що призводить до зміни фінансового статусу в профілі студента.

Очевидно, що всі дані, які застосовуються, стосуються студента та його навчальної картки. Найактуальнішим джерелом даних про студентів є Єдина державна електронна база з питань освіти. Ця база надає широкий список REST-методів, що дозволяють додавати, редагувати та зчитувати дані про студентів.

ЄДЕБО. Для синхронізації даних використано вбудований планувальник завдань Laravel у взаємодії з cron сервера. Cron це звичайний планувальник завдань операційної системи Ubuntu, що встановлена на сервері. Цей планувальник щохвилини ініціює виконання планувальника завдань Laravel, а той в свою чергу перевіряє системний час і у визначений планувальником час запускає команду синхронізації. Для запобігання конфлікту даних, сайт зупиняється на технічні роботи з повідомленням про синхронізацію даних. Лістинг коду частини команди, що синхронізує студентів, що набувають ступінь вищої освіти Бакалавр наведений у додатку В.

Google OAuth2.0. У Національному університеті біоресурсів і природокористування України існує власний корпоративний домен Google-акаунтів. Кожен методист деканату має свій корпоративний акаунт, тому авторизацію користувачів реалізовано саме через OAuth2.0 від Google. У поєднанні з системою авторизації Laravel Socialite, було реалізовано наступний алгоритм реєстрації та авторизації:

1. Адміністратор реєструє корпоративну пошту методиста в списку користувачів. Вказання пароля не потрібне, оскільки користувачу достатньо використовувати пароль від Google-акаунту.

2. На сторінці входу користувач натискає одну єдину кнопку “Вхід через Google”.

3. Socialite перенаправляє користувача на систему авторизації Google, де він вибирає акаунт, за допомогою якого може авторизуватись та вводить пароль за потреби.

4. Якщо користувач зареєстрований у корпоративному домені університету, Google відправляє дані його акаунту назад на сайт.

5. Сайт перевіряє, чи зареєстрований користувач із вказаною електронною поштою. Якщо зареєстрований – він авторизований.

E-Learn. Європейський зразок додатку до диплома вимагає відомості про програму, накопичені індивідуальні кредити та отримані бали/оцінки в пункті 4.3. Тому збереження оцінок є ключовою функціональною вимогою до програмного забезпечення.

Більшість дисциплін мають відповідні електронні курси на навчальному порталі університету E-Learn. На цьому порталі студенти можуть переглянути матеріали лекційних занять та виконати практичні або лабораторні завдання, що поставлені викладачем. А в результаті отримати оцінки за свою роботу.

У взаємодії з розробниками та адміністраторами порталу, було розроблено інтеграцію, що дозволяє викладачеві напряму відправляти оцінки за курс в деканат. На діаграмі послідовності в додатку Г показано алгоритм роботи перенесення оцінок з навчального порталу в розроблену систему.

Фреймворк Laravel містить окремий файл маршрутів (роутів), що призначені для приймання запитів за допомогою API. Для отримання оцінок з навчального порталу було використано саме технологію передачі даних за допомогою цього інтерфейсу. Оскільки, цей інтерфейс дозволяє створювати записи в базі даних, авторизація користувачів інтерфейсу є обов’язковою. Для цього було використано Laravel Sanctum, який генерує токен певного терміну дії

для користувача, що авторизувався. Використовуючи цей токен, користувач може користуватися тільки методами API.

Електронний кабінет студента. У взаємодії з розробником Електронного кабінету студента, було організовано інтеграцію з електронним кабінетом студента. Цей сервіс дозволяє студенту переглядати свою успішність, індивідуальний навчальний план, розклад та формувати свою вибіркову складову навчального плану. Для цього у базі даних програмного забезпечення з документообігу в деканаті вищого навчального закладу створено окремого користувача з правами перегляду даних.

Електронний кабінет студента ідентифікує навчальну картку користувача за корпоративною поштою. Отримує усі дані про здобувача та показує їх у кабінеті [14].

База даних користувачів в домені Національного університету біоресурсів і природокористування України. Оскільки, для інтеграції з електронним кабінетом студента, потрібна його корпоративна електронна пошта, було реалізовано синхронізацію корпоративних електронних пошт з бази даних користувачів. Ця база даних є ієрархічною і для підключення використовується протокол LDAP.

LDAP (Lightweight Directory Access Protocol — полегшений протокол доступу до каталогів) являє собою довершений, гнучкий та широко підтримуваний стандартний механізм взаємодії з серверами каталогів. Найчастіше він застосовується для автентифікації та систематизації даних про користувачів, групи та прикладні програми. Водночас сервер каталогів LDAP слугує універсальним сховищем даних загального призначення, що зумовлює доцільність його впровадження у широкому спектрі програмних рішень [15].

Програмне забезпечення через LDAP звертається до бази даних користувачів. Знаходить кожного користувача за ідентифікатором навчальної картки ЄДЕБО та зберігає у свою базу даних його корпоративну електронну пошту. Це стається після кожної синхронізації даних про студентів з ЄДЕБО.

3.4 Організаційна структура програмного забезпечення

Оскільки, фреймворк Laravel реалізує архітектуру MVC, використано саме цей патерн для розробки програмного забезпечення. Проте, оскільки це вебзастосунок, все зводиться до роботи з папкою /public, зокрема файлом index.php. Це основний файл програмного забезпечення, який запускає увесь застосунок та динамічно змінюється у процесі роботи. Окрім очевидних папок /models, /views, /controllers, фреймворк надає величезну структуру допоміжних засобів. Розглянемо деякі з них.

Зберігання файлів в програмному забезпеченні використано виключно для зберігання рисунків і скріншотів, що використовуються в інструкціях і повідомленнях, а також для зберігання логів системи. Для роботи зі сховищем, фреймворк містить фасад Storage, який має широкий спектр методів для зберігання, кодування, редагування та видалення файлів. Після збереження, фасад надає повний шлях до файлу, що можна використати, щоб потім програмно звернутись до нього ж.

Логування. фреймворк також має свій фасад та файл конфігурації. В цьому файлі можна створити кілька каналів логування, зі своєю назвою, окремим файлом та типом логу.

В програмному забезпеченні системи з документообігу в деканаті вищого навчального закладу реалізовано три канали логування. Стандартний канал, що містить логування щодо авторизації веб-користувачів, повідомлення про помилки у разі їх виникнення, а також інформацію про синхронізацію даних. Канал API, що зберігає логи про процеси в інтеграції з навчальним порталом. Третій канал логування створений для відслідковування видалення даних у базі даних. Це потрібно для відстеження та збереження даних у разі злому користувача, або його навмисної шкідливої діяльності.

Керування зовнішніми залежностями. Оскільки мова програмування PHP має розгалужену екосистему пакетів, у фреймворку передбачено директорію /vendor. У ній зберігаються всі сторонні бібліотеки, зокрема FPDF, що інтегрована для автоматизації генерації PDF-документів (наказів, відомостей

тощо). Використання менеджера пакетів Composer дозволяє забезпечити цілісність версій усіх компонентів системи

База даних. Папка /database містить три папки, що дозволяють швидко розгорнути базу даних для основних потреб застосунку. Факторії /factories використовують для генерації випадкових (фейкових) даних, що є корисним для тестування роботи застосунку. Міграції, що дозволяють швидко побудувати структуру бази даних. Це застосовується для перенесення застосунку з одного сервера на інший. Сідери, те саме, що і факторії, тільки вже генерують чіткі дані, що вказані у файлі.

Конфігурація. Весь фреймворк потребує конфігураційних файлів для керування різними частинами програмного забезпечення. Вони розміщені у папці /config. Редагуванням файлів в цій папці можна регулювати роботу застосунку, авторизації, взаємодії з базою даних та ін. Розглянемо конфігураційний файл database.php. В цьому файлі визначається перелік підключень до баз даних та їх параметри. Задається основне підключення до бази даних, з яким потім працюють моделі. Цей файл автоматично вписує параметри в клас PDO.

PHP Data Objects (PDO) – це об’єктно-орієнтований інтерфейс програмування застосунків, що забезпечує легковажний та універсальний рівень абстракції для взаємодії між мовою програмування PHP та різними СУБД [16].

Середовище. Для основного підключення до бази даних використано глобальну змінну. Такі глобальні змінні можна записувати у .env файл, що розміщений у корені проекту. Також, сюди записані: основна url адреса застосунку, IP-адреса хосту, дані підключення до бази даних, часовий пояс додатку та дані, що стосуються роботи з Google OAuth2.0.

Хостинг. Для хостингу застосунку використано віртуальну машину на фізичному сервері, яка повністю виділена під програмне забезпечення. На цій віртуальній машині розгорнута база даних та веб-застосунок. HTTP підключення забезпечується Apache. Також адміністратори програмного забезпечення мають доступ за допомогою SSH та SFTP.

3.5 Сесія та система глобальних фільтрів

Сесія в мові програмування PHP це асоціативний масив, що містить змінні, доступні для поточного сценарію [18]. У випадку розробленого ПЗ, сесія доступна під час усього часу роботи користувача у застосунку.

Laravel дозволяє зберігати сесію у вигляді файлу на сервері, cookie-файлу на пристрої користувача, базі даних або масиві php [10]. Вибір способу зберігання сесій є важливим аспектом під час розроблення програмного забезпечення системи документообігу деканату закладу вищої освіти, оскільки від нього залежить безпека, стабільність та ефективність роботи вебзастосунку.

Зберігання сесій у cookie-файлах передбачає розміщення даних безпосередньо на стороні користувача. Такий підхід характеризується обмеженим обсягом пам’яті та нижчим рівнем безпеки, оскільки дані можуть бути перехоплені або змінені користувачем. Для інформаційних систем деканату, які працюють із персональними даними студентів, наказами, відомостями успішності та іншою службовою документацією, використання cookie як основного механізму зберігання сесій є недостатньо надійним.

Використання бази даних для зберігання сесій забезпечує централізований доступ до інформації та спрощує масштабування системи. Проте такий підхід створює додаткове навантаження на СУБД, що може негативно впливати на швидкодію вебзастосунку при значній кількості одночасних користувачів. Оскільки база даних у системі документообігу вже активно використовується для обробки навчальної та адміністративної інформації, додаткове зберігання сесій може зменшувати продуктивність системи.

Зберігання сесій у PHP-масиві використовується переважно під час тестування програмного забезпечення, оскільки дані існують лише в межах одного HTTP-запиту та не зберігаються між зверненнями користувача до системи. Через це такий механізм не може бути використаний у реальному середовищі експлуатації.

Найбільш доцільним рішенням для ПЗ є файловий механізм зберігання сесій на сервері. У цьому випадку сесійні дані розміщуються у спеціальних

файлах серверної операційної системи, що забезпечує високий рівень безпеки, оскільки інформація не передається безпосередньо клієнту. Крім того, файлове зберігання не створює додаткового навантаження на базу даних та не потребує складного адміністрування. Такий підхід забезпечує стабільну роботу системи, достатню швидкодію та надійний захист даних користувачів, що є особливо важливим для інформаційних систем закладів вищої освіти.

Фреймворк Laravel має два види сесії: довгострокова і короткострокова. Довгострокова сесія зберігає значення змінної до тих пір, поки її не видалити. Короткострокова – зберігає значення змінної тільки на один HTTP-запит. При виконанні наступного змінна буде видалена.

За замовчуванням сесія зберігає URL поточної сторінки, попередньої сторінки, дані авторизованого користувача, csrf-токен та короткострокову сесію. Елементи конфігурації також описані там. Тому, важливо, щоб сесія зберігалась саме на сервері у зашифрованому вигляді.

Система авторизації фреймворку полягає у валідації користувачів з таблиці бази даних. Оскільки, в результаті аналізу предметної області, було визначено три ролі: адміністратор, методист та користувач API, створено таблицю ролей і зв’язано з таблицею користувачів зв’язком один-до-багатьох. Це дозволить розширити типи користувачів у випадку масштабування застосунку. Кожен методист повинен бачити тільки дані студентів свого факультету. Тому встановлено зв’язок один-до-багатьох з таблицею факультетів. Таким же чином створено зв’язок навчальної картки студента із факультетом.

Для глобальної фільтрації за факультетом, у сесію вписується ідентифікатор факультету користувача, а при запитах до бази даних автоматично додано умову пошуку за факультетом. Це потрібно для зручнішого фільтру за факультетом для адміністратора. У навігаційному меню адміністратор має можливість вибрати факультет, дані якого йому потрібно переглянути.

Станом на 01.05.2026 у Національному університеті біоресурсів і природокористування України 11804 студента здобувають ступінь вищої освіти бакалавр, а магістр – 3036. Тому, важливо передбачити глобальний фільтр за

ступенем вищої освіти, що здобувається. За аналогією із фільтрацією факультету, кожен вебкористувач може вибрати ступінь вищої освіти у навігаційному меню. Це також зберігається у сесії. За цим параметром можна фільтрувати список студентів, список навчальних планів, список успішності академічних груп та ін.

3.6 Функціональна структура програмного забезпечення

За проведеним аналізом предметної області було визначено такі кластери: модуль перегляду списку студентів, модуль формування академічних груп, модуль створення і внесення навчального плану з можливістю призначення його студенту або групі студентів, модуль обліку успішності з можливістю формування відомостей обліку успішності і атестаційних відомостей, модуль формування протоколу засідання стипендіальної комісії з можливістю формування реєстру стипендіатів, модуль формування диплома та модуль формування додатку до диплома.

Модуль перегляду списку студентів. Розроблено інтерфейс користувача, що виводить таблицю зі списком студентів. Таблиця передбачає фільтрацію за прізвищем, роком вступу, типом фінансування, пільгою та ін. Прямо на цій сторінці користувач може призначити навчальний план студенту, синхронізувати дані конкретного студента з ЄДЕБО достроково та переглянути успішність його академічної групи.

Для роботи фільтрів використано асинхронні запити ajax, що є частиною javascript. Цей запит формує json масив з переліком фільтрів та відправляє за відповідним маршрутом. Контролер за цим маршрутом обробляє запит, формує HTML зі списком студентів та повертає його назад до ajax-запиту. Далі цей запит очищує таблицю, що вже бачить користувач і замінює на новий список, сформований контролером.

Також у цій таблиці реалізовано можливість сформувати академічну довідку відповідному студенту. При цьому, програмне забезпечення робить API-запит до ЄДЕБО, та повертає номер довідки у форму, якщо такий існує.

Список студентів дозволяє сформувати документи для призначення додаткових освітніх послуг. Їх поля автоматично заповнюються елементами даних. Залишається тільки роздрукувати та поставити підпис.

У модулі навмисно не реалізовано функціонал ручного додавання або редагування персональних даних студентів з метою мінімізації людського фактора та запобігання внесенню помилкових відомостей методистом. Цілісність та актуальність даних забезпечується виключно шляхом автоматизованої синхронізації з першоджерелом – ЄДЕБО.

Модуль перегляду списку навчальних планів. Інтерфейс модуля перегляду навчальних планів спроєктовано за аналогічним принципом. Функціоналом списку є можливість перегляду списку дисциплін конкретного навчального плану. Система підтримує функції створення та модифікації навчальних планів. Таблиця може бути відфільтрована за спеціальністю, освітньою програмою та роком вступу. Є можливість призначення навчального плану групі, копіювання навчального плану для нового року вступу разом зі списком дисциплін. Реалізовано можливість видалення навчального плану, проте це дозволено тільки у тому випадку, коли він не призначений жодному студенту. При цьому дисципліни видаляться каскадно. Реалізовано функціонал заповнення пунктів додатку до диплома, які є різними для різних навчальних планів. До таких пунктів належать:

- 2.2 Основна (основні) галузь (галузі) знань за кваліфікацією
- 2.3 Найменування та статус закладу, який присвоїв кваліфікацію
- 3.1 Рівень кваліфікації згідно з Національною рамкою кваліфікацій
- 3.3 Вимоги до вступу кваліфікацій
- 4.2 Програмні результати навчання (перший абзац)
- 4.2 Набуті компетенції

Ці дані зберігаються у таблицю Supplement зі зв’язком з таблицею Curriculum, що не була показана на рис. 2.2.

Модуль обліку успішності. Цей модуль зі сторони інтерфейсу користувача передбачає таблицю, що показана у додатку Д. Для кожної дисципліни передбачена можливість внести оцінки. Якщо оцінка менша, ніж 60 балів, або студент не з’явився на іспит, біля оцінки з’являється знак плюс у квадраті. Ця кнопка відкриває модальне вікно, де є можливість внести оцінки за наступними спробами складання іспиту, заліку чи захисту курсової роботи. Кожне модальне вікно завантажується за допомогою асинхронного запиту ajax.

Інтерфейс модуля містить елементи керування для ініціалізації генерації відповідних документів. Також передбачена можливість формування Excel-таблиці з успішністю. Це використовується деканатами для надсилання успішності студентам, які перевіряють свою успішність.

Успішність можна переглядати посеместрово. Також, у випадку, коли відрахованого студента методист не виключив з групи, він може додатково переглядати його успішність ввімкнувши відповідний функціонал.

Модуль формування протоколу засідання стипендіальної комісії. Для фільтрації академічних груп було розроблено вікно з вибором спеціальності, освітньої програми, роком вступу та формою навчання. Після вибору користувач бачить список груп, що відповідають заданим параметрам. Він вибирає перелік груп, для яких хоче сформувати або переглянути протокол засідання стипендіальної комісії. За допомогою асинхронного запиту йому показується список таких протоколів, якщо вже були сформовані.

В правому верхньому куті картки зі списком протоколу розміщено кнопку створення нового протоколу. Після натискання цієї кнопки, з’являється модальне вікно з вибором семестру і навчального року, за результатами якого потрібно сформувати відповідний документ. Коли користувач вибрав семестр і навчальний рік, йому відкривається вікно з полями, що потрібні для протоколу, а також рейтинговий список студентів, що формується відповідно до описаного в розділі 1. Скриншот рейтингового списку показано в додатку Е.

Студентам, що не мають пільги, але претендують на стипендію автоматично встановлено призначення академічної стипендії. Студент, що має

пільгу, проте претендує на академічну стипендію, може відмовитись від неї. В такому випадку академічна стипендія буде нарахована наступному студенту в рейтинговому списку, що не претендує на таку. Для цього зроблено відповідну кнопку плюс в квадраті. Цю кнопку можна побачити навпроти студента Філоненко Іван Олександрович у додатку Е.

Студент може мати кілька пільг. Тому розроблено функціонал вибору пільги, за якою студент отримуватиме соціальну стипендію.

Після вибору або внесення усіх параметрів користувач натискає кнопку Зберегти у правому верхньому куті вікна.

Збережений протокол відображається у списку протоколів. За збереженими даними можна сформувати PDF протоколу, а також реєстр стипендіатів на основі сформованого протоколу.

Для формування PDF створюється новий об’єкт класу FPDF з параметрами орієнтації сторінок, мірою довжини та форматом паперу. Для цього об’єкта викликається метод AddPage(), що додає нову сторінку. Оскільки форма протоколу не передбачає нумерацію сторінок, встановлюється $disablePageNumber = true. Для використання шрифтів, спочатку потрібно їх підключити. Для цього використовується метод AddFont(), що приймає назву шрифта і назву файлу. Ідентифікатор шрифту визначається розробником і використовується під час форматування текстових комірок.

У бібліотеці FPDF позиціонування елементів базується на двовимірній системі координат, де початком відліку (0,0) є верхній лівий кут сторінки. Оскільки зміщення вниз по сторінці традиційно інтерпретується в коді програми через додатні значення координати *y*, реалізовано лінійне відображення точок у межах робочої області документа. Для того, щоб написати текст на сторінці, потрібно:

- Встановити курсор у потрібній точці;
- Створити комірку, вказавши її розміри, товщину межі (якщо межа непотрібна – 0), вказати текст та його вирівнювання;

Комірка створиться у четвертій чверті системи координат, де курсор вважатиметься точкою відліку для неї.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Етап тестування та контролю якості (Quality Assurance, QA) є критично важливою фазою життєвого циклу розробки програмного забезпечення, спрямованою на верифікацію, валідацію та оцінку відповідності розробленого програмного забезпечення вихідним технічним і функціональним вимогам.

Головною метою цього етапу є виявлення дефектів, архітектурних невідповідностей та логічних помилок на всіх рівнях функціонування системи для забезпечення її надійності, безпеки та стабільності перед розгортанням у продуктивному середовищі.

Реалізація процесу тестування у межах проєкту має системний характер і передбачає послідовне виконання таких підетапів:

1. Аналіз вимог та планування тестування. На основі технічного завдання та специфікацій прецедентів визначаються критерії якості, обсяг тестового покриття та розробляється стратегічний документ – тест-план.
2. Проєктування тестових сценаріїв. На цьому кроці створюється набір архітектурних тест-кейсів, що містять чіткі передумови, послідовності кроків та очікувані результати. Також формуються масиви тестових даних для симуляції різних станів системи.
3. Виконання тестування та фіксація дефектів. Здійснюється безпосередній запуск тестів (ручний або автоматизований). У разі виявлення розбіжностей між фактичним та очікуваним результатами генеруються звіти про дефекти у системі відстеження задач, які передаються на розгляд розробникам.
4. Повторне та регресійне тестування. Після модифікації вихідного коду та усунення помилок проводиться ретест для підтвердження виправлення дефекту, а також регресійне тестування з метою перевірки, що внесені зміни не порушили цілісність та коректність роботи раніше верифікованих

модулів системи. зміни не порушили цілісність та коректність роботи раніше верифікованих модулів системи.

Для забезпечення високої надійності, модульності та спрощення процесу регресійного тестування розробленого програмного забезпечення було впроваджено практику автоматизованого тестування. Фреймворк Laravel має глибоку інтеграцію з інструментом PHPUnit, який є галузевим стандартом для об'єктно-орієнтованого тестування у середовищі PHP.

PHPUnit — це фреймворк для модульного тестування програм мовою PHP, який використовується для автоматизованої перевірки правильності роботи окремих частин програмного забезпечення (unit-тестів) [19].

У межах проєкту архітектура тестів була розділена на дві ключові категорії, що конфігуруються у файлі phpunit.xml:

1. Unit-тести (Модульні тести): Націлені на ізольовану перевірку невеликих частин коду (методів, допоміжних класів, алгоритмів валідації) без залучення бази даних, мережевих запитів чи інших зовнішніх сервісів.
2. Feature-тести (Функціональні/Інтеграційні тести): Призначені для верифікації більших фрагментів коду, які моделюють поведінку кількох компонентів одночасно (наприклад, повний цикл обробки HTTP-запиту від контролера до бази даних та повернення JSON-відповіді).

Архітектура та інструменти тестування в Laravel

Перевагою використання вбудованого інструментарію Laravel є наявність розширеного набору методів ствердження (*Assertions*) та допоміжних трейтів, які суттєво спрощують тестування вебдодатків:

- Тестування HTTP-шару: За допомогою вбудованих методів симулюються HTTP-запити ($this->get(), $this->post()) до API-ендпоінтів або маршрутів вебсторінок. Перевірка результатів здійснюється через спеціалізовані асерції, такі як $response->assertStatus(200) для перевірки HTTP-коду відповіді або $response->assertJsonStructure() для валідації структури отриманих JSON-даних.

- Ізоляція бази даних: Для запобігання забрудненню продуктивної чи локальної бази даних під час запусків тестів використовується трейт RefreshDatabase. Він автоматично запускає міграції перед початком тестів і обгортає кожен окремий тест у в окрему транзакцію бази даних, яка скасовується після його завершення. Це забезпечує ідемпотентність та незалежність тестових сценаріїв.
- Генерація фікстур (Фабрики моделей): Для наповнення бази даних тестовими сутностями застосовуються фабрики (Model Factories) разом із бібліотекою Faker. Це дозволяє динамічно генерувати реалістичні та унікальні масиви даних (імена, електронні адреси, ідентифікатори) для симуляції великої кількості записів.
- Макетування (Mocking): У випадках, коли тестуванню підлягає підсистема, що залежить від сторонніх сервісів (наприклад, зовнішні API погоди, сервіси перекладу або відправки електронної пошти), використовується вбудований у Laravel механізм макетування об'єктів (Mockery). Це дозволяє замінити реальні класи «заглушками», які повертають заздалегідь визначений результат, запобігаючи виконанню реальних мережевих запитів під час тестування.

Розглянемо тестування API для функціоналу отримання оцінок з навчального порталу. Для верифікації API-ендпоінту, за допомогою якого здійснюється автентифікація та авторизація зовнішнього навчального порталу як сервісного користувача системи, було розроблено функціональний Feature-тест. Для авторизації такого користувача використовуються такі параметри:

- Grant_type – тип авторизації
- Username – ім’я користувача
- Password – пароль
- App_key – ключ додатку, що згенерований окремо для навчального порталу

Виділено такі тест-кейси:

1. Позитивний сценарій (Positive TestCase)

- Тест-кейс 1: Успішна автентифікація користувача (test_success)
  - Вхідні дані: Валідний JSON-payload, що містить усі обов'язкові поля (grant_type, username, password, app_key) з коректними типами та значеннями знайденого в системі користувача.
  - Очікуваний результат: HTTP-статус 200 OK. Відповідь містить фрагменти з токеном доступу (access_token), даними користувача (user) та типом токена Bearer.

2. Негативні сценарії: Валідація структури JSON та обов'язкових полів (HTTP 422 Unprocessable Entity)

- Тест-кейс 2: Відсутність обов'язкового поля пароля (test_BadJsonNoPass)
  - Вхідні дані: Payload без ключа password.
  - Очікуваний результат: HTTP-статус 422. Повідомлення про помилку структури/валідації з чітким зазначенням відсутності поля password.
- Тест-кейс 3: Відсутність обов'язкового поля логіна (test_BadJsonNoUsername)
  - Вхідні дані: Payload без ключа username.
  - Очікуваний результат: HTTP-статус 422. Повідомлення про помилку валідації з масивом помилок для поля username.
- Тест-кейс 4: Відсутність обов'язкового ключа додатка (test_BadJsonNoAppKey)
  - Вхідні дані: Payload без ключа app_key.
  - Очікуваний результат: HTTP-статус 422. Повідомлення системи про відсутність обов'язкового поля app_key.
- Тест-кейс 5: Відсутність обов'язкового типу авторизації (test_BadJsonNoGrantType)
  - Вхідні дані: Payload без ключа grant_type.
  - Очікуваний результат: HTTP-статус 422. Валідаційна помилка, яка вказує на відсутність поля grant_type.
- Тест-кейс 6: Передача надлишкових/зайвих полів у запиті (test_BadJsonRedutantKeys)

- Вхідні дані: Payload містить додатковий непередбачений ключ extra_key.
- Очікуваний результат: HTTP-статус 422. Повідомлення про те, що JSON містить зайві поля: extra_key.

3. Негативні сценарії: Перевірка типів даних та форматів (HTTP 422 Unprocessable Entity)

- Тест-кейс 7: Невідповідність типу даних поля пароля (test_NotValidPassword)
  - Вхідні дані: Передача в полі password цілого числа (235) замість рядка.
  - Очікуваний результат: HTTP-статус 422. Помилка: "Поле password повинне бути рядком".
- Тест-кейс 8: Невідповідність формату логіна (test_NotValidUser)
  - Вхідні дані: Передача числа (3485) замість рядка у форматі email в полі username.
  - Очікуваний результат: HTTP-статус 422. Помилка: "Поле username повинно бути адресою ел. пошти."
- Тест-кейс 9: Невідповідність типу даних типу авторизації (test_NotValidGrantType)
  - Вхідні дані: Передача числа (3426) в полі grant_type.
  - Очікуваний результат: HTTP-статус 422. Помилка: "Поле grant type повинне бути рядком".
- Тест-кейс 10: Невідповідність типу даних ключа додатка (test_NotValidAppKey)
  - Вхідні дані: Передача числа (38426) в полі app_key.
  - Очікуваний результат: HTTP-статус 422. Помилка: "Поле app key повинне бути рядком".

4. Негативні сценарії: Бізнес-логіка, автентифікація та права доступу (HTTP 401 / 403)

- Тест-кейс 11: Спроба входу неіснуючого користувача (test_NotExistingUser)
  - Вхідні дані: Рядок username має валідний формат email, але такого користувача немає в базі даних.

- Очікуваний результат: HTTP-статус 401 Unauthorized. JSON-відповідь: ['error' => 'Такого користувача не існує'].

- Тест-кейс 12: Передача некоректного ключа додатка (test_BadAppKey)
  - Вхідні дані: Невалідний/неправильний рядок у полі app_key.
  - Очікуваний результат: HTTP-статус 403 Forbidden. JSON-відповідь: ['error' => 'Невірний ключ додатку'].
- Тест-кейс 13: Передача невірного пароля (test_BadPassword)
  - Вхідні дані: Існуючий username, але помилкове значення у полі password.
  - Очікуваний результат: HTTP-статус 401 Unauthorized. JSON-відповідь: ['error' => 'Невірний пароль'].
- Тест-кейс 14: Обмеження доступу за роллю користувача (test_CanNot)
  - Вхідні дані: Валідні дані користувача, який не має прав доступу до API (наприклад, звичайний користувач замість сервісного акаунту).
  - Очікуваний результат: HTTP-статус 403 Forbidden. JSON-відповідь: ['error' => 'Доступ тільки для API користувачів'].

Успішне проходження комплексу автоматизованих Unit-тестів дозволило перейти до етапу дослідної експлуатації системи в продуктивному середовищі з використанням реальних масивів даних. Процес практичної апробації розробленого функціоналу було реалізовано у три послідовні етапи.

Етап 1: Первинне пілотне тестування (Зимова екзаменаційна сесія)

На першій фазі дослідного впровадження було обмежено коло користувачів для проведення контрольованого експерименту.

- Об'єкт дослідження: Студенти та навчальні процеси спеціальностей 151 «Автоматизація та комп’ютерно-інтегровані технології» та 126 «Інформаційні системи та технології».
- Часовий інтервал: Період зимової екзаменаційної сесії.
- Результати та виявлені дефекти: За результатами першого етапу критичних програмних збоїв у роботі архітектури чи бази даних виявлено не було. Проте було зафіксовано вразливості процесного

(людського) характеру. Зокрема, виявлено затримки у верифікації та обробці даних, зумовлені невчасним виконанням регламентних дій працівниками деканатів у середовищі системи.

Етап 2: Оптимізація та організаційні заходи

Перед запуском наступної фази тестування було проведено роботу з усунення виявленого «вузького місця» (bottleneck) в управлінському ланцюгу:

- Дія: Проведено комплексно-пояснювальну роботу та інструктаж із заступниками деканів з навчальної роботи щодо регламенту, таймінгів та специфіки взаємодії з інтерфейсом системи.
- Мета: Навчання персоналу та підвищення операційної швидкості обробки інформації для виключення затримок, спричинених людським фактором.

Етап 3: Масштабування та фінальна апробація (Літня екзаменаційна сесія)

Оскільки результати тестування лише на двох спеціальностях у межах однієї сесії не давали репрезентативної вибірки для оцінки системної стабільності та навантаження, було прийнято рішення про масштабування експерименту.

- Об'єкт дослідження: Усі факультети та спеціальності закладу вищої освіти (загальноуніверситетський рівень).
- Часовий інтервал: Період літньої екзаменаційної сесії.
- Результати: Масштабування системи на рівень усього університету під час пікового навантаження (екзаменаційної сесії) дозволило отримати повну та об’єктивну картину функціонування продукту, підтвердити його відмовостійкість, масштабованість та готовність до фінальної промислової експлуатації.

Практична апробація та валідація функціональних можливостей інтерфейсу користувача розробленої системи здійснювалися впродовж тримісячного періоду дослідного впровадження. У якості експертів та активних

тестувальників виступили працівники деканатів, які за предметною областю є основними акторами системи.

Проведення тривалого експлуатаційного тестування дозволило мінімізувати вплив "ефекту новизни" та оцінити реальну ефективність автоматизації документообігу. За результатами тримісячного моніторингу взаємодії користувачів із веб-інтерфейсом було оптимізовано форми введення даних, покращено механізми фільтрації та підтверджено високу інтуїтивність архітектури користувацького інтерфейсу, що дозволило знизити середній час обробки інформаційних запитів працівниками.

4.2 Вимоги до апаратного і програмного забезпечення. Опис інсталяційного пакету

За проведеним аналізом побудовано діаграму розгортання, що зображена на рис. 4.1.

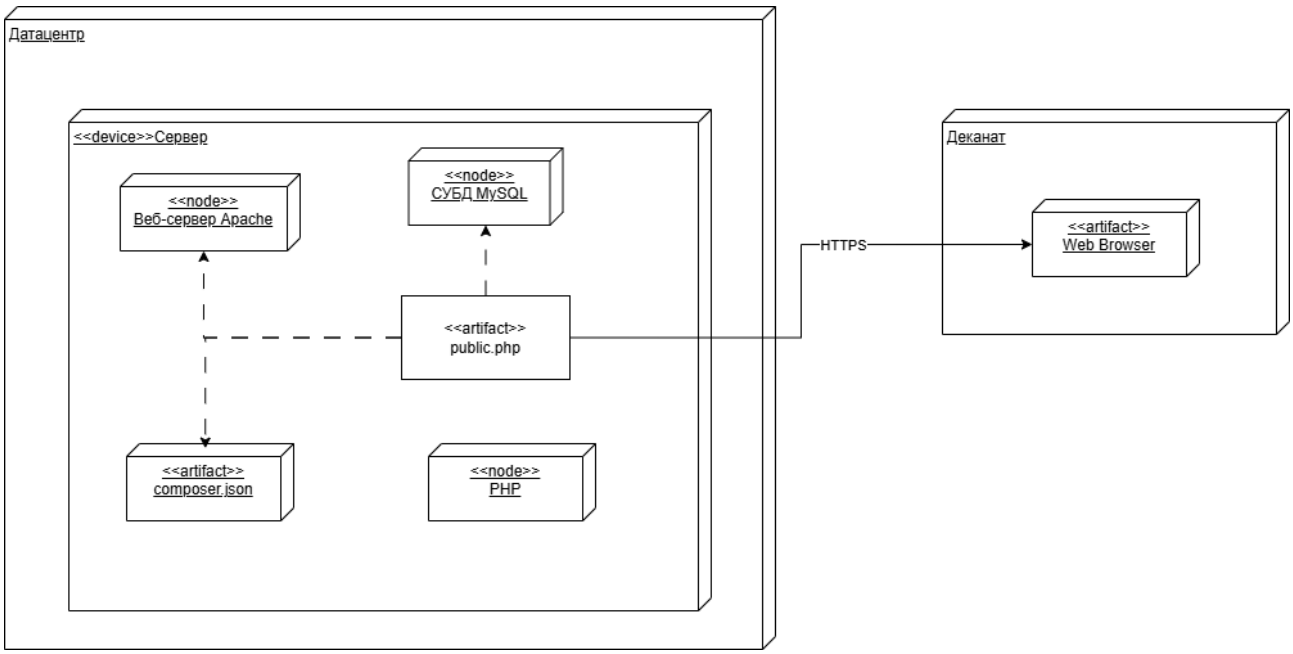


Рис. 4.1 Діаграма розгортання

Діаграма розгортання – діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об’єкти, що виконуються на цих вузлах.

Функціонування розробленої автоматизованої інформаційної системи базується на клієнт-серверній архітектурі. Для забезпечення коректної роботи

системи, її відмовостійкості та швидкодії, встановлено вимоги до апаратного та програмного забезпечення для серверної та клієнтської частин.

Для роботи системи передбачено існування сервера з даними та клієнтського пристрою із встановленим веб-браузером.

Мінімальні системні вимоги до сервера:

- Оперативна пам’ять: 32 ГБ
- Постійна пам’ять: залежить від обсягу даних, що потрібно зберігати. Від 20 до 40 ГБ
- Процесор: 64-бітний, частота від 2 ГГц
- Мережеве підключення

Програмне середовище сервера потребує встановлення будь-якої сучасної операційної системи сімейства Ubuntu Server 22.04 LTS, вебсервера Apache або Nginx, інтерпретатора PHP версій 8.1-8.3 із необхідними модулями розширення, а також реляційної СУБД MySQL версії 8.0 або вище.

Apache HTTP Server — це кросплатформний вебсервер з відкритим вихідним кодом, який забезпечує обробку HTTP-запитів, підтримує модульну архітектуру та використовується для розміщення вебзастосунків і вебсайтів. Сервер є одним із найпоширеніших рішень для хостингу PHP-застосунків, зокрема розроблених із використанням фреймворку Laravel [20].

Оскільки архітектура додатка передбачає товстий сервер і тонкий клієнт, основні обчислення проходять на сервері. Клієнтська частина функціонує всередині веб браузера, тому вимоги до апаратного забезпечення користувача є мінімальними та відповідають специфікаціям сучасного системного ПЗ.

- Апаратні вимоги: Процесор із частотою від 1.6 ГГц, оперативна пам'ять від 4 ГБ, монітор/екран із роздільною здатністю не менше 1024x768 пікселів (інтерфейс системи адаптований під мобільні пристрої).
- Програмні вимоги: Будь-яка сучасна операційна система (Windows 10/11, Linux, macOS, Android, iOS), що підтримує встановлення актуальних версій

веб браузерів із підтримкою стандарту HTML5, CSS3 та рушія JavaScript (Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari).

Для спрощення процесу розгортання, масштабування та міграції інформаційної системи на нові серверні потужності, інсталяційний пакет розроблено у двох конфігураціях:

1. Контейнеризований пакет (Docker-інфраструктура): До складу інсталяційного пакета входить набір конфігураційних файлів архітектури розгортання:

- Dockerfile — інструкція для побудови базового шару серверного середовища з усіма необхідними розширеннями PHP;
- docker-compose.yml — маніфест для автоматичного створення, лінкування та запуску ізольованих контейнерів вебсервера (Nginx/Apache), інтерпретатора (PHP-FPM) та сервера баз даних (MySQL).

Перевага: Пакет дозволяє розгорнути систему однією командою docker-compose up -d на будь-якому сервері без необхідності ручного налаштування ПЗ.

2. Стандартний архів вихідного коду (для традиційного хостингу): У разі розгортання без використання контейнеризації, інсталяційний пакет містить:

- Директорії вихідного коду проєкту (структура архітектури MVC Laravel);
- Файл конфігурації залежностей composer.json (пакети авторизації, генерації PDF-звітів тощо);
- Файл змінних оточення .env, що містить шаблони параметрів підключення до бази даних та генерації ключів безпеки додатку;
- Скрипти міграцій баз даних (database/migrations) для автоматичного створення структури таблиць, зв'язків та тригерів, а також сідери (database/seeders) для первинного наповнення системи адміністративними ролями.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи на тему «Програмне забезпечення системи з документообігу в деканаті вищого навчального закладу» було здійснено повний цикл розробки автоматизованої інформаційної системи, що охоплює етапи аналізу предметної області, проєктування бази даних, розробки прикладного програмного забезпечення та підготовки інсталяційних рішень для подальшого розгортання та експлуатації в Національному університеті біоресурсів і природокористування України.

Проведений аналіз діяльності та документообігу в деканаті дозволив визначити ключові напрями автоматизації, які включають облік контингенту студентів, адміністрування індивідуальних навчальних планів, фіксацію академічної успішності та спроб складання контрольних заходів, а також автоматизацію стипендіального забезпечення. На основі виявлених потреб та діючих Правил призначення академічних стипендій НУБіП було сформульовано технічне завдання, визначено функціональні та нефункціональні вимоги, а також розроблено концепцію програмного забезпечення.

Було побудовано логічну модель даних у вигляді ER-діаграми, яку в процесі деталізації специфікацій та усунення надлишковості (зокрема, винесення розрахункового поля загального рейтингу) було приведено до вимог третьої нормальної форми (3NF). Спроєктовану структуру бази даних реалізовано в середовищі СУБД MySQL, де для кодування символів української мови використано набір utf8mb4_unicode_ci, а цілісність даних на фізичному рівні підтримано механізмом зовнішніх ключів із каскадними операціями.

Прикладне програмне забезпечення розроблено на основі фреймворку Laravel (PHP), що забезпечило реалізацію архітектурного патерну MVC. Клієнтська частина базується на шаблонізаторі Blade, мові розмітки HTML5, стилях CSS3 та бібліотеці Bootstrap 5, що гарантує кросплатформеність та адаптивність інтерфейсу. Завдяки впровадженню асинхронних запитів JavaScript (AJAX) реалізовано динамічну фільтрацію масивів даних без повного

перезавантаження сторінок, що суттєво оптимізувало роботу користувача-методиста. Для автоматизації генерації друкованих форм документів європейського зразка (додатків до дипломів, протоколів, відомостей) у систему інтегровано об'єктно-орієнтовану бібліотеку FPDF.

Окремою науково-практичною та інноваційною складовою розробленої системи є її глибока інтеграція в інформаційне середовище університету за допомогою шлюзів та інтерфейсів:

- Реалізовано REST API клієнт під керуванням планувальника завдань Laravel та системного демона cron сервера для автоматизованої двосторонньої синхронізації даних із Єдиною державною електронною базою з питань освіти (ЄДЕБО), що виключило людський фактор при ручному введенні даних;
- Інтегровано протокол Google OAuth2.0 (через пакет Laravel Socialite) для безпечної наскрізної автентифікації співробітників за допомогою корпоративних облікових записів;
- Розроблено захищений токенами Laravel Sanctum API-шлюз для автоматичного імпорту оцінок із навчального порталу E-Learn;
- Організовано взаємодію за протоколом LDAP з ієрархічною базою користувачів домену НУБіП для синхронізації корпоративних пошт студентів та забезпечення роботи Електронного кабінету студента.

Якість та відмовостійкість розробленого рішення підтверджено шляхом комплексного автоматизованого тестування (Unit та Feature-тестів на базі PHPUnit з використанням Mocking-заглушок та ізольованих транзакцій бази даних), а також тримісячною дослідною експлуатацією в продуктивному середовищі університету під час зимової та літньої екзаменаційних сесій.

Для спрощення розгортання та забезпечення мобільності продукту підготовлено Docker-інфраструктуру (Dockerfile та маніфест docker-compose.yml), що дозволяє розгорнути ізольовані контейнери додатка, вебсервера та СУБД однією командою на будь-яких потужностях.

У результаті виконаної роботи всі поставлені цілі та завдання були повністю досягнуті. Розроблена система успішно автоматизує ключові процеси документообігу деканату, поєднуючи високий рівень інформаційної безпеки (через виділені канали логування операцій видалення), стабільність під піковими навантаженнями сесій та архітектурну гнучкість, і є повністю готовою до промислової експлуатації в реальних умовах ЗВО.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Реєстр суб'єктів освітньої діяльності. Кількість осіб, зарахованих на навчання: Єдина державна електронна база з питань освіти. URL: https://registry.edbo.gov.ua/opendata/entrant/ (дата звернення: 23.04.2026)

2. Посібник із побудови схем UML і моделювання баз даних: Microsoft, Марін Перес. URL: https://www.microsoft.com/uk-ua/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата звернення: 23.04.2026)

3. Про вищу освіту: Закон України від 01.07.2014 №1556-18. Відомості Верховної Ради України. 2014. №37-38. ст. 2004

4. Правила призначення академічних стипендій у Національному університеті біоресурсів і природокористування України від 22.11.2023.

5. Use-case diagrams: IBM. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (дата звернення 24.04.2026)

6. "Sequence Diagrams". Unified Modeling Language 2.5.1 OMG Document Number formal/2017-12-05: Object Management Group Standards Development Organization (OMG SDO). Грудень 2017. с. 595

7. Абстракція в програмуванні допомагає чи тільки ускладнює процес?: FoxMinded. URL: https://foxminded.ua/abstraktsiia-v-prohramuvanni/ (дата звернення: 23.04.2026)

8. Про ЄДЕБО: ЄДЕБО. URL: https://info.edbo.gov.ua/about/ (дата звернення: 23.04.2026)

9. What is an entity relationship diagram?: IBM. URL: https://www.ibm.com/think/topics/entity-relationship-diagram (дата звернення 24.04.2026)

10. Matt Stauffer: Laravel Up & Running A Framework for Building Modern PHP Apps. Київ: O'REILLY, 2024, 540 с.

11. Сучасні мови програмування: список, рейтинг і види у 2026 році: WeZoom. URL: https://wezom.com.ua/ua/blog/naykraschi-movi-programuvannya-2025-roku-

yak-vibir-tehnologiy-mozhe-vplinuti-na-uspih-vashogo-dodatku (дата звернення 28.04.2026)

12. What is FPDF?: FPDF Library. URL: https://www.fpdf.org/ (дата звернення: 27.04.2026)

13. Форма додатка до диплома європейського зразка: Міністерство освіти і науки України. URL: https://mon.gov.ua/static-objects/mon/sites/1/vishcha-osvita/2021/02/Formy%20dokumentiv%20pro%20vyshchu%20osvitu/forma-dodatka-do-dyploma-yevropeyskoho-zrazka.docx (дата звернення 28.04.2026)

14. Понзель Я. Ю. РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ПОБУДОВИ ІНДИВІДУАЛЬНОГО ПЛАНУ НАВЧАННЯ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ": дис. докт. філософії: 122. Київ, 2025. 239 с.

15. LDAP.com Lightweight Directory Access Protocol. URL: https://ldap.com/ (дата звернення 28.04.2026)

16. Benchmarking PHP–MySQL Communication: A Comparative Study of MySQLi and PDO Under Varying Query Complexity: Andrijević, N. URL: https://www.mdpi.com/2079-9292/15/1/21 (дата звернення 28.04.2026)

17. Rumbaugh James; Jacobson Ivar; Booch Grady: The unified modeling language reference manual. Масачусетс: Addison-wesley, 1999, 584 с.

18. PHP Documentation : official website. URL: https://www.php.net/docs.php (дата звернення: 29.04.2026).

19. PHPUnit: official website. URL: http://www.phpunit.de/ (дата звернення: 29.04.2026).

20. Apache Software Foundation. Apache HTTP Server Documentation. URL: https://httpd.apache.org/docs/ (дата звернення: 29.04.2026).

ДОДАТКИ

ДОДАТОК А

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('studyprogram', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('faculty_id');
            $table->string('StudyProgramID')->nullable();
            $table->string('StudyProgramName')->nullable();
            $table->string('StudyProgramNameEn')->nullable();
            $table->set('HigherEducation', ['1', '2', '3']);
            $table->string('Accreditation')->nullable();
            $table->string('AccreditationEn')->nullable();
            $table->timestamps();
            $table->string('UserEdited_by');

            $table->index('faculty_id', 'studyprogram_faculty_idx');
            $table->foreign('faculty_id', 'studyprogram_faculty_fk')->on('faculties')->references('id');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('studyprograms');
    }
};
```

ДОДАТОК Б

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class StudyProgram extends Model
{
    protected $table = 'studyprograms';
    protected $guarded = [];
    use HasFactory;

    public function faculty(): BelongsTo
    {
        return $this->BelongsTo(Faculty::class, 'faculty_id', 'id');
    }
}
```

ДОДАТОК В

```
<?php

namespace App\Console\Commands;

use App\Facades\AdminZone\Settings;
use Illuminate\Console\Command;
use Illuminate\Support\Carbon;
use Illuminate\Support\Facades\Artisan;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Log;

class StudentsSync extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'students-sync:run';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Students synchronization with EDBO';

    /**
     * Execute the console command.
     */

    private function SyncBakNavch($token, $Pages): void
```

```
    {
        $counter = 0;
        $i = 0;
        $this->line('Бакалаври');
        do {
            $response2 = Http::retry(3, 20000)->withHeaders([
                'Authorization' => 'Bearer ' . $token,
                'Content-Type' => 'application/json; charset=utf-8',
            ])->timeout(120)-
>post('http://edbogate.nubip.edu.ua:8080/data/EDEBOWebApi/api/studentEducations/list', [
                'UniversityId' => 7,
                'QualificationGroupId' => 1,
                'HistoryFilterId' => -1,
                'pageSize' => 20,
                'pageNo' => $i,
            ]);
            $StudentsList = $response2->json();
            if (Carbon::parse($StudentsList[0]['modifyDate'])->startOfDay()->lte(Carbon::parse('2021-
01-01 10:00:00')->startOfDay()) == true) {
                break;
            }
            foreach ($StudentsList as $student) {
                $counter++;
                $this->output->write("\r$counter");
                if (DB::table('personeducationcards')->where('EducationCardID',
$student['educationId'])->exists()) {
                    $record = DB::table('personeducationcards')
                        ->where('EducationCardID', $student['educationId'])
                        ->select('ModifyDate')
                        ->first();

                    if (Carbon::parse($student['modifyDate'])->startOfDay()->lte(Carbon::parse($record-
>ModifyDate)->startOfDay()) == true) {
                        continue;
                    }
```

```
            }

            if($student['historyTypeId'] == 16 || $student['historyTypeId'] == 60 ||
$student['historyTypeId'] == 62 || $student['historyTypeId'] == 63 || $student['historyTypeId'] ==
65){
                DB::table('personeducationcards')->where('EducationCardID', $student['educationId'])-
>update([
                    'StudentStatus' => '3',
                    'updated_at' => now(),
                    'UserEdited_by' => 'Оновлено через API',
                    'ModifyDate' => $student['modifyDate'],
                ]);
                continue;
            }
            elseif($student['historyTypeId'] == 110){
                DB::table('personeducationcards')->where('EducationCardID', $student['educationId'])-
>update([
                    'StudentStatus' => '1',
                    'updated_at' => now(),
                    'UserEdited_by' => 'Оновлено через API',
                    'ModifyDate' => $student['modifyDate'],
                ]);
                continue;
            }
            elseif($student['historyTypeId'] == 100 || $student['historyTypeId'] == 101 ||
$student['historyTypeId'] == 111){
                DB::table('personeducationcards')->where('EducationCardID', $student['educationId'])-
>update([
                    'StudentStatus' => '0',
                    'updated_at' => now(),
                    'UserEdited_by' => 'Оновлено через API',
                    'ModifyDate' => $student['modifyDate'],
                ]);
                continue;
            }
```

```
            Log::info( 'Навчальна картка ' . $student['educationId'] . '. Код персони ' .
$student['personCodeU']);
            // 1. studentEducations/info
            $response2 = Http::retry(3, 20000)->withHeaders([
               'Authorization' => 'Bearer ' . $token,
               'Content-Type' => 'application/json; charset=utf-8',
            ])->timeout(120)->post(
               'http://edbogate.nubip.edu.ua:8080/data/EDEBOWebApi/api/studentEducations/info',
               ['EducationId' => $student['educationId']]
            );
            $student['fullInfo'] = $response2->successful() ? $response2->json() : null;

            // 2. physPersons/documents
            try{
               $response2 = Http::retry(3, 20000)->withHeaders([
                  'Authorization' => 'Bearer ' . $token,
                  'Content-Type' => 'application/json; charset=utf-8',
               ])->timeout(120)->post(

'http://edbogate.nubip.edu.ua:8080/data/EDEBOWebApi/api/physPersons/documents',
                  ['PersonCode' => $student['personCodeU']]
               );
               $student['documents'] = $response2->successful() ? $response2->json() : null;
            }
            catch (\Exception $e){
               Log::warning('Не вдалося отримати документи', [
                  'personCode' => $student['personCodeU'],
                  'error' => $e->getMessage(),
               ]);
               continue;
            }

            // 3. studentEducations/history/list
            $response2 = Http::retry(3, 20000)->withHeaders([
               'Authorization' => 'Bearer ' . $token,
```

```
            'Content-Type' => 'application/json; charset=utf-8',
        ])->timeout(120)->post(

'http://edbogate.nubip.edu.ua:8080/data/EDEBOWebApi/api/studentEducations/history/list',
            ['EducationId' => $student['educationId']]
        );
        $student['history'] = $response2->successful() ? $response2->json() : null;

        // 4. physPersons/info
        try{
            $response2 = Http::retry(3, 20000)->withHeaders([
                'Authorization' => 'Bearer ' . $token,
                'Content-Type' => 'application/json; charset=utf-8',
            ])->timeout(120)->post(
                'http://edbogate.nubip.edu.ua:8080/data/EDEBOWebApi/api/physPersons/info',
                ['PersonCode' => $student['personCodeU']]
            );
            $student['personInfo'] = $response2->successful() ? $response2->json() : null;
        }
        catch (\Exception $e){
            Log::warning('Не вдалося отримати фізичну особу', [
                'personCode' => $student['personCodeU'],
                'error' => $e->getMessage(),
            ]);
            continue;
        }
        $dataParsedPerson = [];
        $dataParsedPerson['PersonCode'] = $student['personId'];
        if (!isset($student['fullInfo']['countryName'])) {
            continue;
        }
        if ($student['fullInfo']['countryName'] == 'Україна') {
            $dataParsedPerson['Citizen'] = '1';
        } else {
            $dataParsedPerson['Citizen'] = '0';
```

```
            }
            $Apostrof = array("`", "’");
            $dataParsedPerson['LastName'] = str_replace($Apostrof,
"",$student['personInfo']['lastName']);
            $dataParsedPerson['FirstName'] = str_replace($Apostrof,
"",$student['personInfo']['firstName']);
            $dataParsedPerson['MiddleName'] = str_replace($Apostrof,
"",$student['personInfo']['middleName']);
            $dataParsedPerson['Birthday'] = date('d.m.Y',
strtotime($student['personInfo']['birthday']));
            $dataParsedPerson['LastNameEn'] = $student['personInfo']['lastNameEn'];
            $dataParsedPerson['FirstNameEn'] = $student['personInfo']['firstNameEn'];
            $dataParsedPerson['MiddleNameEn'] = $student['personInfo']['middleNameEn'];
            $dataParsedPerson['SexId'] = ($student['personInfo']['sexId'] == 1 ? '1' : '2');
            foreach ($student['documents'] as $Document) {
               if ($Document['idPersonDocument'] == $student['personInfo']['id_PersonDocument'])
{
                  $dataParsedPerson['PersonDocumentType'] =
$Document['personDocumentTypeName'];
                  $dataParsedPerson['PersonDocumentSeries'] = $Document['documentSeries'];
                  $dataParsedPerson['PersonDocumentNumber'] = $Document['documentNumbers'];
                  $dataParsedPerson['IssuedBy'] = $Document['documentIssued'];
                  $dataParsedPerson['IssuedDate'] = date('d.m.Y',
strtotime($Document['documentDateGet']));
               }
            }
            $dataParsedPerson['IPN'] = $student['personInfo']['ipnNumber'];
            $dataParsedPerson['UserEdited_by'] = 'Оновлено через API';

            if(DB::table('persons')->where('PersonCode', '=', $student['personId'])->exists()){
               $dataParsedPerson['updated_at'] = now();
               DB::table('persons')->where('PersonCode', '=', $student['personId'])-
>update($dataParsedPerson);
            }
            else{$dataParsedPerson['created_at'] = now();
```

```
            DB::table('persons')->insert($dataParsedPerson);
        }
        $dataEducationCardParsed = [];
        $person = DB::table('persons')->where('PersonCode', '=', $student['personId'])->select('id',
'LastName', 'FirstName', 'MiddleName')->first();
        $dataEducationCardParsed['person_id'] = $person->id;
        $dataEducationCardParsed['EducationCardID'] = $student['educationId'];
        if(!isset($student['history'][0]['facultyName'])){
            continue;
        }
        if (DB::table('studyprograms')
            ->join('faculties', 'studyprograms.faculty_id', '=', 'faculties.id')
                ->where('faculties.FacultyName', '=', $student['history'][0]['facultyName'])
                ->where('studyprograms.StudyProgramID', '=',
$student['universityStudyProgramId'])->exists()) {
            $FacultyAndStudyProgram = DB::table('studyprograms')
                ->join('faculties', 'studyprograms.faculty_id', '=', 'faculties.id')
                ->where('faculties.FacultyName', '=', $student['history'][0]['facultyName'])
                ->where('studyprograms.StudyProgramID', '=',
$student['universityStudyProgramId'])
                ->select('studyprograms.id', 'studyprograms.faculty_id',
'studyprograms.speciality_id')->first();
            $dataEducationCardParsed['faculty_id'] = $FacultyAndStudyProgram->faculty_id;
            $dataEducationCardParsed['StudyProgram_id'] = $FacultyAndStudyProgram->id;
            $dataEducationCardParsed['speciality_id'] = $FacultyAndStudyProgram-
>speciality_id;
            if (DB::table('specializations')->where('StudyProgram_id', '=',
$FacultyAndStudyProgram->id)->exists()) {
                $Specialization = DB::table('specializations')->where('StudyProgram_id', '=',
$FacultyAndStudyProgram->id)->select('id')->first();
                $dataEducationCardParsed['specialization_id'] = $Specialization->id;
            }
        } else {
            Log::info('Освітньої програми з ID: ' . $student['universityStudyProgramId'] . 'на
факультеті' . $student['history'][0]['facultyName'] . 'немає в базі даних');
```

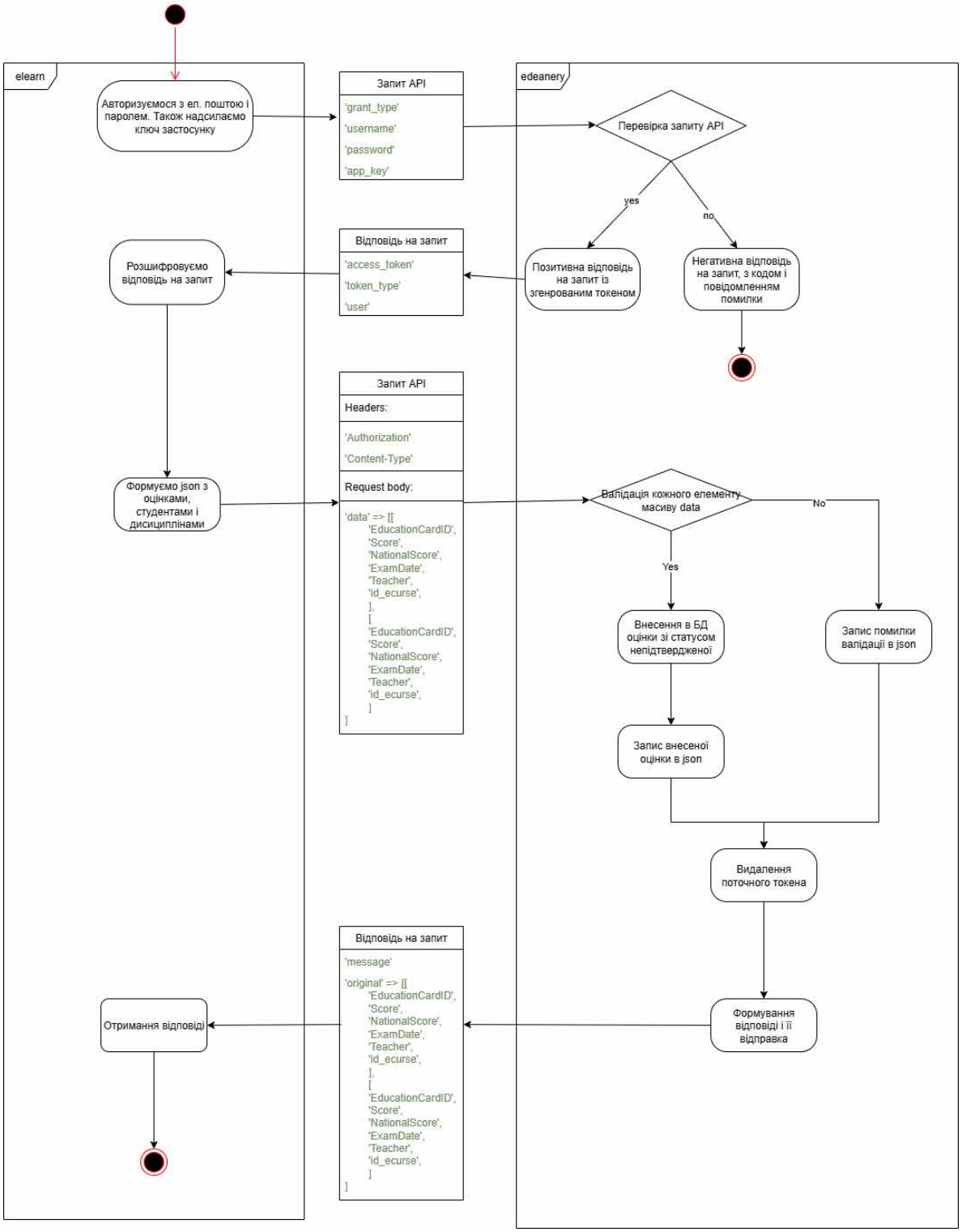
```
            continue;
        }

        $dataEducationCardParsed['CB'] = (string)$student['konkursValue'];
        switch ($student['history'][0]['personEducationPaymentTypeName']) {
            case 'Контракт':
                $dataEducationCardParsed['PaymentTypeId'] = '2';
                break;
            case 'Бюджет':
                $dataEducationCardParsed['PaymentTypeId'] = '1';
                break;
        }
        $dataEducationCardParsed['EducationForm'] =
$student['history'][0]['educationFormName'];
        $dataEducationCardParsed['YearEntry'] =
substr($student['fullInfo']['educationDateBegin'], 0, 4);
        $dataEducationCardParsed['EducationDateBegin'] = date('d.m.Y',
strtotime($student['fullInfo']['educationDateBegin']));
        $dataEducationCardParsed['EducationDateEnd'] = date('d.m.Y',
strtotime($student['fullInfo']['educationDateEnd']));
        $EducationBasetemp1 = explode('; ', $student['fullInfo']['eduDocInfo']);
        $EducationBasetemp2 = explode('  ', $EducationBasetemp1[0]);
        $dataEducationCardParsed['EducationBase'] = $EducationBasetemp2[0];
        $EducationBaseSeriaNomer = explode(' ', $EducationBasetemp2[1]);
        if ($dataParsedPerson['Citizen'] == '1') {
            if (isset($EducationBaseSeriaNomer[1])) {
                $dataEducationCardParsed['DocumentSeries'] = $EducationBaseSeriaNomer[0];
                $dataEducationCardParsed['DocumentNumber'] = $EducationBaseSeriaNomer[1];
            } else {
                $EducationBaseSeriaNomer = explode('/', $EducationBasetemp2[1]);
                if (isset($EducationBaseSeriaNomer[1])) {
                    $dataEducationCardParsed['DocumentSeries'] = $EducationBaseSeriaNomer[0];
                    $dataEducationCardParsed['DocumentNumber'] = $EducationBaseSeriaNomer[1];
                } else {
                    $dataEducationCardParsed['DocumentNumber'] = $EducationBaseSeriaNomer[0];
```

```
                }
            }
            $cleaned = str_replace('Ким видано: ', '', $EducationBasetemp1[2]);
            $escaped = addslashes($cleaned);
            $dataEducationCardParsed['IssuedBy'] = $escaped;
            $dataEducationCardParsed['IssuedByYear'] = $EducationBasetemp1[1];
        }
        switch ($student['history'][0]['qualificationGroupName']) {
            case 'Бакалавр':
                $dataEducationCardParsed['HigherEducation'] = '1';
                break;
            case 'Магістр':
                $dataEducationCardParsed['HigherEducation'] = '2';
                break;
            case 'Доктор філософії':
                $dataEducationCardParsed['HigherEducation'] = '3';
                break;
        }
        $dataEducationCardParsed['StudentStatus'] = '2';
        $dataEducationCardParsed['UserEdited_by'] = 'Оновлено через API';
        $dataEducationCardParsed['ModifyDate'] = $student['modifyDate'];
        //dd($dataEducationCardParsed);
        if(DB::table('personeducationcards')->where('EducationCardID', '=',
$student['educationId'])->exists()) {
            $StudyProgram = DB::table('studyprograms')
                ->join('faculties', 'studyprograms.faculty_id', '=', 'faculties.id')
                ->where('faculties.FacultyName', '=', $student['history'][0]['facultyName'])
                ->where('studyprograms.StudyProgramID', '=',
$student['universityStudyProgramId'])
                ->select('studyprograms.id', 'studyprograms.faculty_id',
'studyprograms.speciality_id')->first();
            $EducationCard = DB::table('personeducationcards')->where('EducationCardID', '=',
$student['educationId'])->first();
            if(DB::table('personeducationcards')->where('EducationCardID', '=',
$student['educationId'])
```

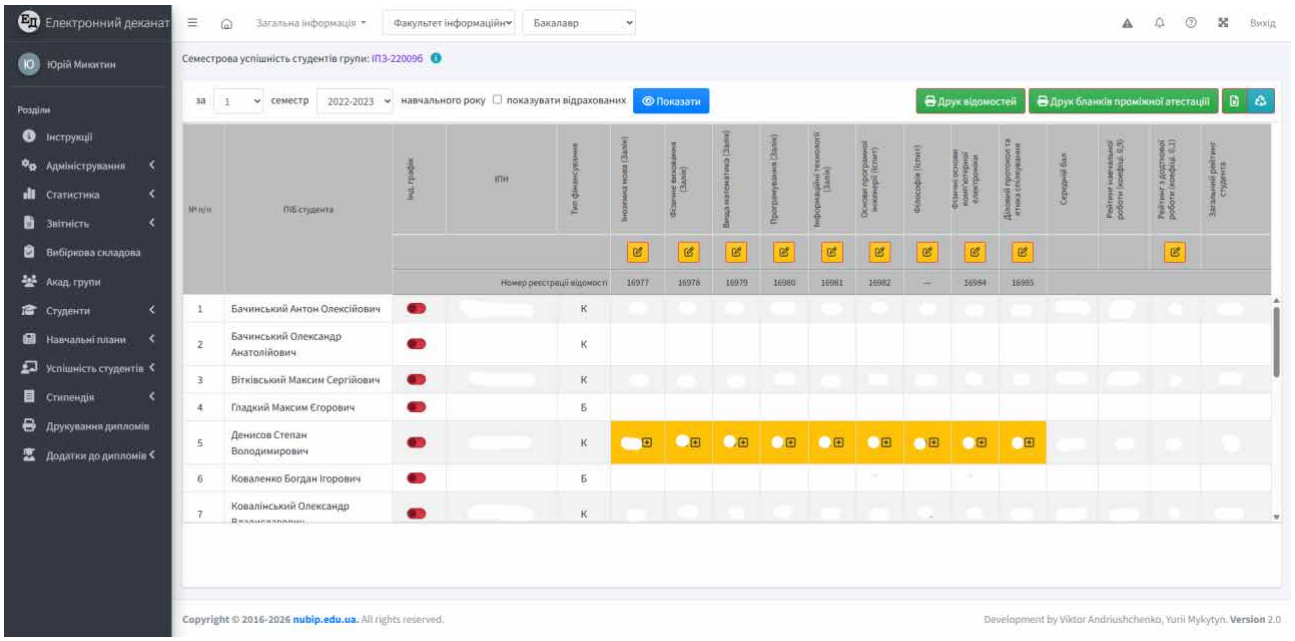
```
                ->where('StudyProgram_id', '!=', $StudyProgram->id)->exists()){
            DB::table('EducationCardCurriculumDiscipline')->where('personeducationcards_id',
'=', $EducationCard->id)->delete();
            DB::table('personeducationcards')->where('EducationCardID', '=', $EducationCard-
>id)->update([
                'StudyGroup_id' => NULL,
                'curriculum_id' => NULL,
            ]);
        }
        $dataEducationCardParsed['updated_at'] = now();
        DB::table('personeducationcards')->where('EducationCardID', '=',
$student['educationId'])->update($dataEducationCardParsed);
    }
    else{
        $dataEducationCardParsed['created_at'] = now();
        DB::table('personeducationcards')->insert($dataEducationCardParsed);
    }
    $personeducationcard = DB::table('personeducationcards')
        ->join('studyprograms', 'studyprograms.id', '=',
'personeducationcards.StudyProgram_id')
        ->where('EducationCardID', '=', $student['educationId'])-
>select('personeducationcards.id AS PersonEduCardID', 'studyprograms.StudyProgramName',
'studyprograms.StudyProgramNameEn')->first();
    // Закічення обробки даних personeducationcard
    // Обробка даних duration
    if (!DB::table('duration')
        ->where('personeducationcard_id', '=', $personeducationcard->PersonEduCardID)
        ->exists()) {
        $dataDurationParsed['created_at'] = now();
        $dataDurationParsed['personeducationcard_id'] = $personeducationcard-
>PersonEduCardID;
        $dataDurationParsed['EducationDateBegin'] = date('d.m.Y',
strtotime($student['fullInfo']['educationDateBegin']));
        $dataDurationParsed['EducationDateEnd'] = date('d.m.Y',
strtotime($student['fullInfo']['educationDateEnd']));
```

```
                $dataDurationParsed['University'] = 'Національний університет біоресурсів і природокористування України';
                $dataDurationParsed['UniversityEn'] = 'National University of Life and Environmental Sciences of Ukraine';
                $dataDurationParsed['EducationProgram'] = $personeducationcard->StudyProgramName;
                $dataDurationParsed['EducationProgramEn'] = $personeducationcard->StudyProgramNameEn;
                $dataDurationParsed['UserEdited_by'] = 'Оновлено через API';
                DB::table('duration')->insert($dataDurationParsed);
            }
        }
        $i++;
        usleep(2);
    } while ($i <= $Pages);
    Log::info('Бакалаври, що навчаються синхронізовані');
    return;
  }
```

ДОДАТОК Г


elearn
Авторизуємося з ел. поштою і паролем. Також надсилаємо ключ застосунку
Запит API
'grant_type'
'username'
'password'
'app_key'
edeanery
Перевірка запиту API
yes
no
Позитивна відповідь на запит із згенрованим токеном
Негативна відповідь на запит, з кодом і повідомленням помилки
Відповідь на запит
'access_token'
'token_type'
'user'
Розшифровуємо відповідь на запит
Формуємо json з оцінками, студентами і дисциплінами
Запит API
Headers:
'Authorization'
'Content-Type'
Request body:
'data' => [[
'EducationCardID',
'Score',
'NationalScore',
'ExamDate',
'Teacher',
'id_ecurse',
],
[
'EducationCardID',
'Score',
'NationalScore',
'ExamDate',
'Teacher',
'id_ecurse',
]
]
Валідація кожного елементу масиву data
No
Yes
Внесення в БД оцінки зі статусом непідтвердженої
Запис помилки валідації в json
Запис внесеної оцінки в json
Видалення поточного токена
Формування відповіді і її відправка
Відповідь на запит
'message'
'original' => [[
'EducationCardID',
'Score',
'NationalScore',
'ExamDate',
'Teacher',
'id_ecurse',
],
[
'EducationCardID',
'Score',
'NationalScore',
'ExamDate',
'Teacher',
'id_ecurse',
]
]
Отримання відповіді

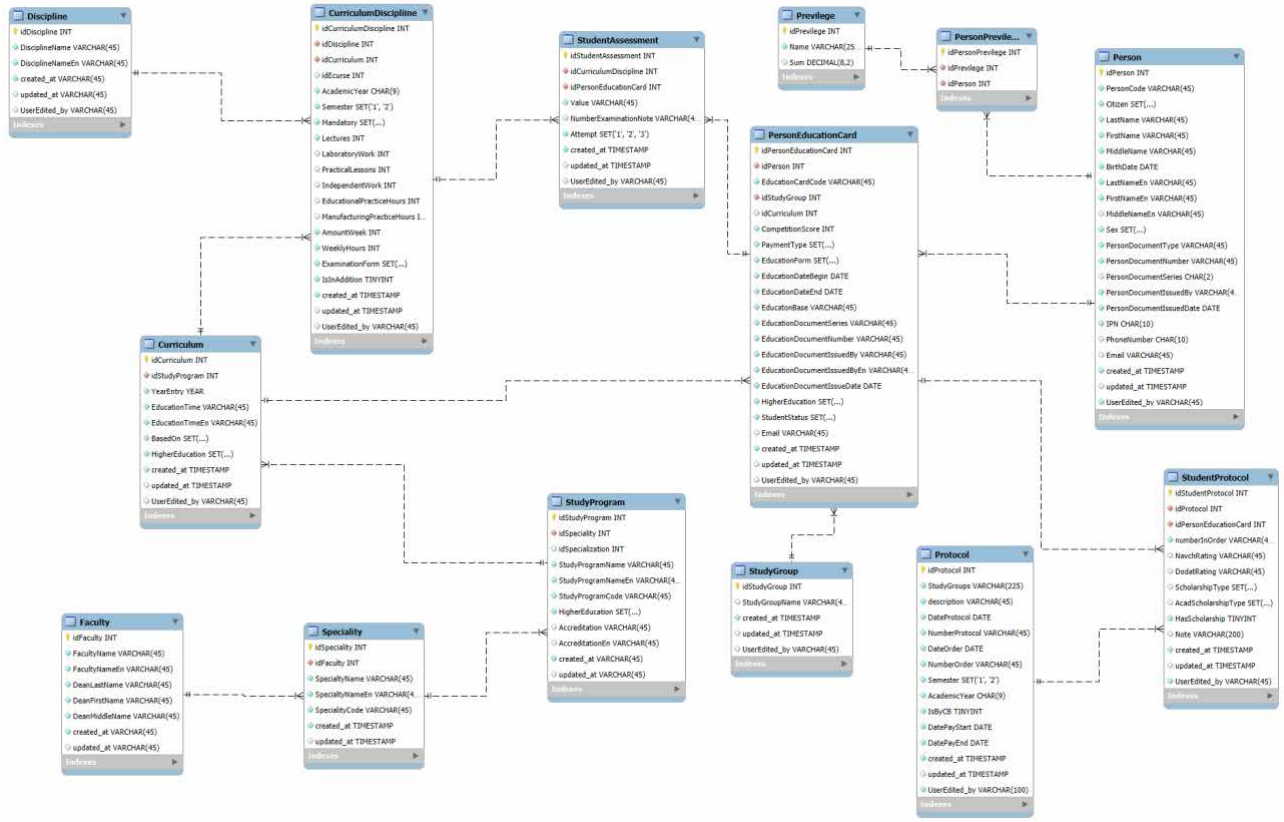
ДОДАТОК Д


Електронний деканат
Загальна інформація
Факультет інформаційн
Бакалавр
Вихід
Юрій Микитин
Семестрова успішність студентів групи: ІП3-220096
Розділи
Інструкції
Адміністрування
Статистика
Звітність
Вибіркова складова
Акад. групи
Студенти
Навчальні плани
Успішність студентів
Стипендія
Друкування дипломів
Додатки до дипломів
за 1 семестр 2022-2023 навчального року показувати відрахованих Показати
Друк відомостей
Друк бланків проміжної атестації
№ п/п
ПІБ студента
Інд. графік
ІПН
Тип фінансування
Номер реєстрації відомості
16977
16978
16979
16980
16981
16982
16984
16985
1 Бачинський Антон Олексійович К
2 Бачинський Олександр Анатолійович К
3 Вітківський Максим Сергійович К
4 Гладкий Максим Єгорович Б
5 Денисов Степан Володимирович К
6 Коваленко Богдан Ігорович Б
7 Ковалінський Олександр К
Copyright © 2016-2026 nubip.edu.ua. All rights reserved.
Development by Viktor Andriushchenko, Yurii Mykytyn. Version 2.0

ДОДАТОК Е

№ п/п	ПІБ студента	ІПН	Рейтинговий бал (всього, до 100 балів)	Бали за навчальні досягнення (до 90 балів)	Додаткові бали (до 10 балів)	Наявність пільг (які саме)	Примітка
Студенти, які навчаються за державним замовленням							
1	Прокопенко Павло Геннадійович	[redacted]	87.3	87.3	0		академічна (звичайна)
2	Закомірний Максим Сергійович	[redacted]	86.51	86.51	0		академічна (звичайна)
3	Стрішенець Ангеліна Олександрівна	[redacted]	86.29	86.29	0		академічна (звичайна)
4	Троян Олександр Володимирович	[redacted]	86.29	86.29	0		академічна (звичайна)
5	Литовка Альбіна Сергіївна	[redacted]	83.59	83.59	0	○ Студенти, зареєстровані як внутрішньо переміщені особи (до 23 років) (1180.00 грн) ○ Студенти, які проживають у населених пунктах на лінії зіткнення (1180.00 грн)	○ академічна (звичайна) ○ соціальна
6	Авдієвський Максим Олександрович	[redacted]	81.9	81.9	0		академічна (звичайна)
7	Гладкий Максим Єгорович	[redacted]	79.88	79.88	0		академічна (звичайна)
8	Філоненко Іван Олександрович		79.88	79.88	0		⊡
9	Леонтович Олександр Сергійович		67.73	67.73	0	Студенти, батьки яких мають статус учасника бойових дій (до 23 років) (1180.00 грн)	○ академічна (звичайна) ○ соціальна
10	Панасюк Анна Костянтинівна		74.48	74.48	0		має/мав академічну заборгованість
11	Спєлов Сергій Сергійович		67.95	67.95	0	Студенти, які проживають у населених пунктах на лінії зіткнення (1180.00 грн)	має/мав академічну заборгованість
						Діти осіб, визнаних постраждалими учасниками	

ДОДАТОК Ж


Discipline
idDiscipline INT
DisciplineName VARCHAR(45)
DisciplineNameEn VARCHAR(45)
created_at VARCHAR(45)
updated_at VARCHAR(45)
UserEdited_by VARCHAR(45)
CurriculumDiscipline
idCurriculumDiscipline INT
idDiscipline INT
idCurriculum INT
idCurse INT
AcademicYear CHAR(9)
Semester SET('1', '2')
Mandatory SET(...)
Lectures INT
LaboratoryWork INT
PracticalLessons INT
IndependentWork INT
EducationalPracticeHours INT
ManufacturingPracticeHours I...
AmountWeek INT
WeeklyHours INT
ExaminationForm SET(...)
IsInAddition TINYINT
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
StudentAssessment
idStudentAssessment INT
idCurriculumDiscipline INT
idPersonEducationCard INT
Value VARCHAR(45)
NumberExaminationNote VARCHAR(4...
Attempt SET('1', '2', '3')
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
Previlege
idPrevilege INT
Name VARCHAR(25...
Sum DECIMAL(8,2)
PersonPrevile...
idPersonPrevilege INT
idPrevilege INT
idPerson INT
Person
idPerson INT
PersonCode VARCHAR(45)
Citizen SET(...)
LastName VARCHAR(45)
FirstName VARCHAR(45)
HiddleName VARCHAR(45)
BirthDate DATE
LastNameEn VARCHAR(45)
FirstNameEn VARCHAR(45)
HiddleNameEn VARCHAR(45)
Sex SET(...)
PersonDocumentType VARCHAR(45)
PersonDocumentNumber VARCHAR(45)
PersonDocumentSeries CHAR(2)
PersonDocumentIssuedBy VARCHAR(4...
PersonDocumentIssuedDate DATE
IPN CHAR(10)
PhoneNumber CHAR(10)
Email VARCHAR(45)
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
PersonEducationCard
idPersonEducationCard INT
idPerson INT
EducationCardCode VARCHAR(45)
idStudyGroup INT
idCurriculum INT
CompetitionScore INT
PaymentType SET(...)
EducationForm SET(...)
EducationDateBegin DATE
EducationDateEnd DATE
EducatonBase VARCHAR(45)
EducationDocumentSeries VARCHAR(45)
EducationDocumentNumber VARCHAR(45)
EducationDocumentIssuedBy VARCHAR(45)
EducationDocumentIssuedByEn VARCHAR(4...
EducationDocumentIssueDate DATE
HigherEducation SET(...)
StudentStatus SET(...)
Email VARCHAR(45)
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
Curriculum
idCurriculum INT
idStudyProgram INT
YearEntry YEAR
EducationTime VARCHAR(45)
EducationTimeEn VARCHAR(45)
BasedOn SET(...)
HigherEducation SET(...)
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
StudyProgram
idStudyProgram INT
idSpeciality INT
idSpecialization INT
StudyProgramName VARCHAR(45)
StudyProgramNameEn VARCHAR(4...
StudyProgramCode VARCHAR(45)
HigherEducation SET(...)
Accreditation VARCHAR(45)
AccreditationEn VARCHAR(45)
created_at VARCHAR(45)
updated_at VARCHAR(45)
StudyGroup
idStudyGroup INT
StudyGroupName VARCHAR(4...
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
Protocol
idProtocol INT
StudyGroups VARCHAR(225)
description VARCHAR(45)
DateProtocol DATE
NumberProtocol VARCHAR(45)
DateOrder DATE
NumberOrder VARCHAR(45)
Semester SET('1', '2')
AcademicYear CHAR(9)
IsByCB TINYINT
DatePayStart DATE
DatePayEnd DATE
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(100)
StudentProtocol
idStudentProtocol INT
idProtocol INT
idPersonEducationCard INT
number1nOrder VARCHAR(4...
NavchRating VARCHAR(45)
DodatRating VARCHAR(45)
ScholarshipType SET(...)
AcadScholarshipType SET(...)
HasScholarship TINYINT
Note VARCHAR(200)
created_at TIMESTAMP
updated_at TIMESTAMP
UserEdited_by VARCHAR(45)
Faculty
idFaculty INT
FacultyName VARCHAR(45)
FacultyNameEn VARCHAR(45)
DeanLastName VARCHAR(45)
DeanFirstName VARCHAR(45)
DeanMiddleName VARCHAR(45)
created_at VARCHAR(45)
updated_at VARCHAR(45)
Speciality
idSpeciality INT
idFaculty INT
SpecialityName VARCHAR(45)
SpecialityNameEn VARCHAR(4...
SpecialityCode VARCHAR(45)
created_at TIMESTAMP
updated_at TIMESTAMP

ДОДАТОК Л

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій
Декларація про академічну доброчесність та використання ШІ

Я, Микитин Юрій Романович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення системи з документообігу в деканаті вищого навчального закладу» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що:

інструменти ШІ Gemini були використані для:

▪ перекладу іншомовних джерел;

▪ допомоги у написанні/відлагодженні програмного коду;

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___»_____ 2026 р. _____ **Юрій МИКИТИН**

(підпис)