МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Козак Роман Ігорович

Серверний модуль для системи анонімного голосування та оцінки викладачів університету засобами Node.js / Server module for the system of anonymous voting and university lecturers evaluation based on Node.js

напрям підготовки: 6.050102 - Комп’ютерна інженерія
фахове спрямування - Комп’ютерні системи та мережі
Бакалаврська робота

Виконав студент групи КСМ 41/1
Козак Роман Ігорович

Науковий керівник:
Вовкодав О.В.

Тернопіль - 2018

РЕЗЮМЕ

Бакалаврська робота містить 52 сторінки пояснюючої записки, 3 рисунки, 7 таблиць, 2 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою даної роботи є розробка програмного рішення для зберігання та керування даними для оцінювання викладачів вищих навчальних закладів.

Предметною областю даної роботи є вищий навчальний заклад, який має чітку структуру підрозділів і достатню кількість кваліфікованих викладачів.

Спроектовано базу даних що містить наступні сутності: користувач, факультет, кафедра, викладач, коментар, вчене звання, науковий ступінь.

Сформовано основні функціональні вимоги для серверного модуля системи оцінювання викладачів: надсилання клієнту даних про викладачів, факультети та кафедри, поточні оцінки викладачів, перевірка достовірності клієнта, а також його прав на читання чи зміну існуючих даних. Відповідно до наведених вимог побудовано загальний алгоритм роботи системи.

Програмне забезпечення розроблено на основі ітеративної моделі розробки. Ітеративний підхід передбачає створення програмного продукту не за один довгий період часу, а за короткі ітерації. При розробці серверного модуля враховано специфіку роботи з Node.js та фрейворком Express.

Ключові слова:. NODE.JS, EXPRESS, ОЦІНЮВАННЯ, ВИКЛАДАЧ, ФАКУЛЬТЕТ.

RESUME

The bachelor's thesis contains 52 pages of explanatory note, 3 figures, 7 tables, 2 appendices. Volume of graphic material 2 sheets of A3 format.

The purpose of this work is to develop a software solution for data storage and management for the evaluation of teachers of higher education institutions.

The subject area of this work is a higher education institution, which has a clear structure of units and a sufficient number of qualified teachers.

A database containing the following entities has been designed: user, faculty, department, teacher, commentary, academic title, scientific degree.

The main functional requirements for the server module of the teacher assessment system are formed: sending the client data on teachers, faculties and departments, current teacher assessments, checking the client's authenticity, as well as his rights to read or change existing data. According to the given requirements the general algorithm of work of system is constructed.

The software is based on an iterative development model. The iterative approach involves creating a software product not for one long period of time, but for short iterations. When developing the server module, the specifics of working with Node.js and the Express framework are taken into account.

Keywords:. NODE.JS, EXPRESS, EVALUATION, LECTURER, FACULTY.

ЗМІСТ

ВСТУП

Дана науково-дослідницька робота присвячена темі «Проектування серверного модуля для системи анонімного голосування та оцінки викладачів університету засобами Node.js».

Виведення української освіти на новий рівень вимагає впровадження інноваційних рішень в освітній процес. Для вирішення питань ефективної взаємодії зі студентами можна використати передові підходи та напрацювання в сфері управління з інших галузей економіки, які наразі найбільш активно працюють в напрямку вдосконалення існуючих методів. Таким чином, за рахунок мінімальних капіталовкладень в освіту можна підготувати велику кількість кваліфікованих кадрів.

На даний момент проблеми педагогічного процесу в українській освіті не розкриваються в повній мірі і кількість робіт по цьому питанню досить обмежена. Існуючі моделі оцінювання викладачів передбачають оцінювання лише керівником і колегами, що є доволі суб’єктивним і не дозволяє отримати повного бачення ситуації.

Актуальність даної роботи зумовлена тим, що без розробки зручних моделей управління та зворотнього зв’язку неможлива прозора та ефективна педагогічна робота у вищих навчальних закладах.

Предметною областю даної роботи є вищий навчальний заклад, який має чітку структуру підрозділів і достатню кількість кваліфікованих викладачів.

Метою даної роботи є розробка програмного рішення для зберігання та керування даними для оцінювання викладачів вищих навчальних закладів.

Для вирішення задачі оцінювання викладачів використано п’ятибальну систему оцінювання за критеріями. Основними принципами є об’єктивність, достовірність, доступність та прозорість. Застосування даної системи паралельно з відгуками від керівника і колег дозволить проводити оцінювання за методом 360 градусів, що є більш ефективним за існуючі методи.

1 ПОСТАНОВКА ЗАДАЧІ ТА СУЧАСНИЙ СТАН ПРОБЛЕМИ

1.1 Сутність освіти

Поняття освіти є досить складним і багатогранним. В загальному освіта – це процес засвоєння людиною систематичних знань та навичок, розвиток розуму і почуттів, формування світогляду. Поняття «освіта» передбачає не лише знання та вміння, як результат навчання, але й уміння критично мислити, аналізувати, оцінювати та творити.

Основні теорії формування змісту освіти склалися в кінці XVIII - на початку XIX ст. Вони отримали назву матеріальної і формальної теорії формування змісту освіти. Першу ще називають теорією дидактичного матеріалізму або енциклопедизму. Її прихильники вважали, що головна мета навчання полягає в передачі студентам якомога більшого обсягу знань з різноманітних галузей науки [1].

Прихильниками матеріальної теорії формування змісту освіти були відомі педагоги XIX ст. Енциклопедична модель використовувалася в привілейованих навчальних закладах Європи, в класичних гімназіях. Своїх прихильників вона має і сьогодні.

На противагу енциклопедизму формальна теорія формування змісту освіти або дидактичний формалізм, трактувала навчання лише як засіб розвитку здібностей і пізнавальних інтересів учнів. Тому головним критерієм при доборі навчальних предметів повинна слугувати розвиваюча цінність навчальної дисципліни, найбільш представлена в математиці і класичних мовах. Теоретичною основою дидактичного формалізму є положення про перенесення знань і умінь, набутих в одній галузі діяльності, в іншу.

Прихильником дидактичного формалізму був Геракліт, який вважав, що «багато знання розуму не додає». Аналогічну позицію займав Цицерон. В Новий час теорію дидактичного формалізму, прин-циповою основою якої була філософія І. Канта, а також неогуманізм. В Німеччині аналогічні погляди

пропагував А. Дістервег у своєму «Посібнику до освіти німецьких учителів» (1835).

Заслуга представників формальної теорії змісту освіти полягає в тому, що вони звернули увагу на необхідність розвитку здібностей і пізнавальних інтересів учнів, їхньої уваги, пам'яті, уяви, мислення. Слабкість цієї теорії була обумовлена тим, що в програмах навчання в першу чергу були відображені математика і класичні мови. Однак, на зразок того, як пізнання фактів (предметів, явищ, подій і процесів) впливає на формування мислення, так і розвиток мислення обумовлює можливості оволодіння учнем знаннями фактологічного характеру. Ця двобічна діалектична залежність не була достатньо чітко осмислена ні представниками енциклопедизму, що визначали навчання через зміст, ні прихильниками формалізму, які переоціню-вали в навчанні значення суб'єктивно-процесуального аспекту [2].

Обидві теорії були піддані глибокій науковій критиці К. Ушинським. Він писав, що формальний розвиток розуму є несуттєвою ознакою і розум розвивається тільки в дійсно реальних знаннях. Школа, на його думку, повинна збагачувати людину знаннями і одночасно привчати її користуватися цим багатством. Починаючи з К. Ушинського, в педагогіці утверджується ідея забезпечення єдності матеріального і формального підходів до добору змісту освіти.

В сучасній педагогічній науці існують різні концепції змісту освіти, корені яких сягають все тих же теорій формальної та матеріальної освіти. Кожна з них пов’язана з певним трактуванням місця і функції людини у світі та суспільстві [4].

Одна з концепцій змісту освіти трактує його як адаптовані основи наук, що вивчаються в школі, залишаючи поза увагою такі якості особистості, як здібність до творчості, уміння реалізувати свободу вибору, справедливе ставлення до людей. Дана практика спрямована на пристосування учнів і

студентів до науки та виробництва, але не до повноцінного самостійного життя в демократичному суспільстві.

Інша концепція розглядає зміст освіти як сукупність знань, вмінь та навичок, які повинні засвоїти учні. Передбачається, що оволодіння знаннями та вміннями дозволить людині адекватно функціонувати всередині існуючої суспільної структури. Достатньо вимагати від людини, щоб вона знала і вміла – не більше.

В сучасних умовах всього цього недостатньо. Вирішення завдань, пов'язаних з функціонуванням окремих сфер діяльності суспільства, вимагає від учнів не лише оволодіння певним навчальним змістом, але й розвиток у них таких якостей, як сила волі, відповідальність [3].

Протягом останніх десятиліть XX століття у світлі ідеї гуманізації освіти все більше утверджувався особистісно-орієнтовний підхід до виявлення сутності освіти. Так, І. Лернер і М. Скаткін розуміли під змістом освіти педагогічно адаптовану систему знань, умінь і навичок, досвіду творчої діяльності і емоційно-вольового відношення, засвоєння якого покликано забезпечити формування всебічно розвиненої особистості, підготовленої до відтворення і розвитку матеріальної і духовної культури суспільства. Отже, при особистісно-орієнтованому підході до визначення сутності змісту освіти абсолютною цінністю є не відчужені від особистості знання, а сама людина. Такий підхід забезпечує свободу вибору змісту освіти з метою задоволення освітніх, духовних, культурних і життєвих потреб особистості, гуманне ставлення до особистості, становлення її індивідуальності і можливі самореалізації в культурно-освітньому просторі.

Професія викладача вищого навчального закладу – одна із найбільш творчих і складних професій, в яких поєднано науку та мистецтво. Ця професія споріднена з працею письменника (творчість у підготовці матеріалу), режисера і постановника (створення замислу і його реалізація),

актора (в педагогічній діяльності інструментом є особистість викладача), педагога, психолога та науковця.

Діяльність викладача вищої школи має високу соціальну значущість і займає одне з центральних місць у державотворенні, формуванні національної свідомості і духовної культури українського суспільства. Професійна педагогічна діяльність викладача може розглядатися як цілісна динамічна система, яка містить п'ять структурних елементів: суб'єкт педагогічного впливу, об'єкт педагогічного впливу, предмет їх спільної діяльності, цілі навчання, засоби педагогічної комунікації. Ці компоненти складають систему, бо ні один з них не може бути замінений іншим або їх сукупністю. Всі вони знаходяться у прямій та зворотній взаємозалежності.

Отже, праця викладача вищого навчального закладу являє собою свідому, доцільну діяльність щодо навчання, виховання і розвитку студентів. Вона є двобічною - спеціальною та соціально-виховною, найважливішими передумовами ефективності педагогічної праці.

Таким чином, праця викладача вищого навчального закладу – це висококваліфікована розумова праця щодо підготовки й виховання кадрів спеціалістів вищої кваліфікації з усіх галузей економіки, інтелектуальної еліти суспільства, української інтелігенції. В ній органічно поєднані знання та ерудиція вченого і мистецтво педагога, висока культура та інтелектуальна, моральна зрілість, усвідомлення обов'язку і почуття відповідальності.

Викладач вищого навчального закладу виконує такі функції:

- організаторську (керівник, провідник у лабіринті знань, умінь, навичок);
- інформаційну (носій найновішої інформації);
- трансформаційну (перетворення суспільно значущого змісту знань в акт індивідуального пізнання);
- орієнтовно-регулятивну (структура знань педагога визначає структуру знань студента);

- мобілізуючу (переведення об'єкту виховання у суб'єкт, самовиховання, саморуху, самоутвердження).

Конкретний зміст праці, права та обов'язки професора, доцента, викладача вищого навчального закладу визначає статут відповідного навчального закладу.

1.2 Оцінювання навчально-педагогічної діяльності викладача

Такий вид діяльності, як освітній процес, повинен проходити об’єктивне оцінювання. Оцінювання надає можливість удосконалити існуючі моделі роботи, сприяє прозорості роботи викладачів.

Відсутність науково обґрунтованих чітких показників та критеріїв оцінювання породжує формалізм, необ’єктивність в оцінюванні педагогічної діяльності, що веде до недостатнього рівня їхнього професіоналізму, і в кінцевому результаті негативно впливає на якість освіти. З урахуванням цього виникає необхідність дослідження проблеми оцінювання педагогічної діяльності викладача в вищих навчальних закладах [5].

Педагогічна діяльність безпосередньо регламентується нормативно-правовими документами в галузі освіти. У науково-педагогічних джерелах існує безліч параметрів оцінювання результативності педагогічної діяльності, що піддаються кількісному визначенню, вимірюванню та порівняльному аналізу на засадах кваліметричного підходу. Розробленням кваліметричного стандарту діяльності педагогічного колективу і педагога зокрема займалися В. Беспалько, Г. Дмитренко, Г. Єльникова. Систему параметрів визначення ефективності навчально-виховного процесу залежно від діяльності вчителя розроблено академіком Ю. Бабанським. Однак проблема оцінювання теоретичної та практичної підготовки вчителів є актуальною і в нинішніх умовах розвитку освіти.

Обґрунтування критеріїв та показників ефективності навчально-виховного процесу ускладнюється специфікою педагогічної діяльності та особливостями теорії. Не все в педагогічній діяльності може бути оцінено з позицій загальнозначущих положень. Знання про людину, на які спирається сучасна педагогічна наука, ніколи не можуть бути кінцевими та абсолютними. Вчитель як професіонал не може покластися лише на здоровий глузд та інтуїцію, він має керувати навчально-виховним процесом, а не творити його стихійно. Водночас наука не відповідає на всі питання, які виникають у роботі педагога, а наукове обґрунтування критеріїв оцінювання його професійної діяльності обмежене рівнем розвитку фахових дисциплін.

Обґрунтування критеріїв та показників ефективності навчально-виховного процесу ускладнюється специфікою педагогічної діяльності та особливостями теорії. Не все в педагогічній діяльності може бути оцінено з позицій загальнозначущих положень. Знання про людину, на які спирається сучасна педагогічна наука, ніколи не можуть бути кінцевими та абсолютними. Викладач як професіонал не може покластися лише на здоровий глузд та інтуїцію, він має керувати навчально-виховним процесом, а не творити його стихійно. Водночас наука не відповідає на всі питання, які виникають у роботі педагога, а наукове обґрунтування критеріїв оцінювання його професійної діяльності обмежене рівнем розвитку фахових дисциплін.

У всьому світі передбачаються дії щодо підвищення вимог до якості діяльності вчителів. Так, у США, наприклад, розгорнута ціла кампанія «Проти нудних методів навчання», створено систему екзаменів і тестів для вчителів, звертається увага на диференціацію в оплаті залежно від розвитку їхньої майстерності. Останньому приділяється особлива увага. На думку американського вченого Д. Дорнема, без диференціації в оплаті педагогів залежно від рівня їхньої майстерності країна не зрушить з місця у реформі в системі освіти [3, с.48]. Зазначене питання широко обговорюється на міжнародних конференціях, проводяться дискусії, на яких робляться спроби

виробити нові критерії оцінки діяльності навчальних закладів загалом та «осучаснити» традиційні підходи до оцінювання діяльності педагогів [6].

Незважаючи на активні пошуки варіантів оцінювання ефективності результатів педагогічної діяльності, до цього часу це питання не знайшло такого розв'язання, яке б влаштовувало не лише адміністраторів, але й педагогів. У загальному плані проблема підвищення якості праці педагогічних працівників є постійною та універсальною в усіх країнах, тому що завжди існує суперечність між вимогами держави і суспільства до школи та її реальними можливостями виконання цих вимог. Проблема якості педагогічної праці визнається на всіх рівнях управління освітою. Ця проблема містить у собі принаймні три аспекти: якість підготовки майбутніх педагогів; правила і методика відбору вчителів на роботу як умова якості їхньої професіональної діяльності; оцінка педагогічної діяльності (атестація) в навчальному закладі або в незалежному від нього «центрі якості» чи асоціації спеціалістів у сфері освіти. Як стверджує у своєму дослідженні Т. Казаріцька, оцінювання педагогічної діяльності вчителя виконує дві функції:

- орієнтувальну, яка полягає в можливості визначати напрями і способи подальшої діяльності;
- стимулювальну, що виявляється в спонуканні людини до діяльності в певному напрямку [7].

Сфера застосування оцінювання педагогічної діяльності вчителя досить широка, вона використовується для розв'язання таких питань, як:

- підбір і розстановка нових працівників;
- кар'єрне зростання;
- раціоналізація засобів і методів роботи, управлінських процедур;
- удосконалення організації праці;
- побудова ефективної системи мотивації трудової діяльності;
- оцінювання ефективності навчання працівників;
- модернізація планів і програм підвищення кваліфікації.

Таким чином, оцінювання педагогічної діяльності викладача займає пріоритетне місце в системі освіти. Як засвідчує досвід, у практиці не існує чітких критеріїв оцінювання педагогічної діяльності вчителя.

1.3 Аналіз існуючих програмних рішень

Аналізуючи поточну ситуацію на ринку, варто зазначити, що систем анонімного оцінювання викладачів досить небагато. Здебільшого такі системи англомовні й орієнтовані на ринки США, Великобританії та інших європейських країн.

Однією з них є RateYourLecturer. Система є зручною у користуванні з приємним дизайном та простою навігацією. Недоліком є те, що вона не орієнтована на українського користувача і не пристосована до реалій українського ринку. Також вона не є анонімною, для оцінювання викладача необхідно авторизуватися.

Іншим аналогом є RateMyTeachers – система оцінювання, орієнтована переважно на середні загальноосвітні школи США. На відміну від попередньої оцінювання може відбуватися анонімно. Система зручна та інтуїтивно зрозуміла, однак в ній відсутня українська мова, відповідно вона не адаптована до українського ринку.

RateMyProfessors – ще одна англомовна система оцінювання викладачів з сучасним та зручним дизайном. Від користувача не вимагається бути зареєстрованим для того, щоб залишити свою оцінку. Недоліком виглядає занадто велика кількість рекламних оголошень на веб-сторінках.

СтудЗона – одна з небагатьох систем на пострадянському просторі. Її вирізняє велика база навчальних закладів та викладачів, зрозумілими рейтингами та списками. Недоліком СтудЗони є застарілий дизайн, який вже не відповідає вимогам часу.

ПрофессорРейтинг – одна з найбільш вдалих реалізацій ідей анонімного оцінювання. Це система з хорошим дизайном та навігацією. Головна сторінка зображена на рисунку 1.1. Єдиним та все ж значним недоліком є продуктивність і надійсність системи. Часто сторінки довго завантажуються або ж користувач бачить помилки замість очікуваної сторінки.

Рисунок 1.1 – Головна сторінка ПрофессорРейтинг

Провівши аналіз існуючих аналогів, можна дійти до висновку, що у всіх є вагомі недоліки, уникнення яких дозволить зробити новий та успішний продукт.

1.4 Постановка задачі

Однією з важливих закономірностей сучасної епохи, що характеризується світовими глобалізаційними процесами, є зростання ролі освіти як основи прогресу суспільства. У зв’язку з цим перед сучасним вищим навчальним закладом постає завдання розроблення та впровадження у систему управління новітніх технологій, комплексних критеріїв визначення якості

освітнього процесу, процедури оцінювання досягнень закладу освіти та педагогічної діяльності викладачів.

Теорія і практика оцінювання навчально-педагогічної діяльності в Україні нині перебувають на етапі модернізації. Враховуючи міждисциплінарний характер оцінювання навчально-педагогічної діяльності, окремих методик, притаманних лише оцінюванню, воно не має. Та всі методики базуються на певних принципах, дотримання яких є складовою оцінювання й важливим інструментом гарантії якості, а значить, підвищення об’єктивності і збільшення практичної значущості оцінювання. Ефективне впровадження цих принципів сприятиме вдосконаленню системи оцінювання навчально-педагогічної діяльності.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ СЕРВЕРНОГО МОДУЛЯ

2.1 Алгоритмічне забезпечення системи

Алгоритмічне забезпечення – це набір алгоритмів, які дозволяють вирішити задачі функціонального характеру інформаційної системи [8].

Основними функціональними вимогами для серверного модуля системи оцінювання викладачів є надсилання клієнту даних про викладачів, факультети та кафедри, поточні оцінки викладачів, перевірка достовірності клієнта, а також його прав на читання чи зміну існуючих даних. До нефункціональних вимог варто віднести надійсніть та швидкодію системи.

Відповідно до наведених вимог можна побудувати загальний алгоритм роботи системи, який зображений на рисунку 3.2.

Спочатку система приймає запит від клієнта і перевіряє чи це не запит на захищений ресурс. Якщо це так, то перевіряються дані авторизації, тобто чи має клієнт права для доступу до цього ресурсу. Залежно від результатів перевірки клієнт отримує потрібні дані, або ж отримує інформацію про помилку.

У випадку, якщо клієнт запитує незахищений ресурс, то тут можливі також кілька варіантів. Клієнт може робити запит на аутентифікацію і тоді система перевіряє правильність логіна та пароля і відправляє йому відповідь з випадково згенерованим токеном аутентифікації. Якщо ж запит не на аутентифікацію, то це може бути запит на завантаження файла. Система повинна коректно обробити такий випадок, а в разі помилки надіслати всю інформацію про неї.

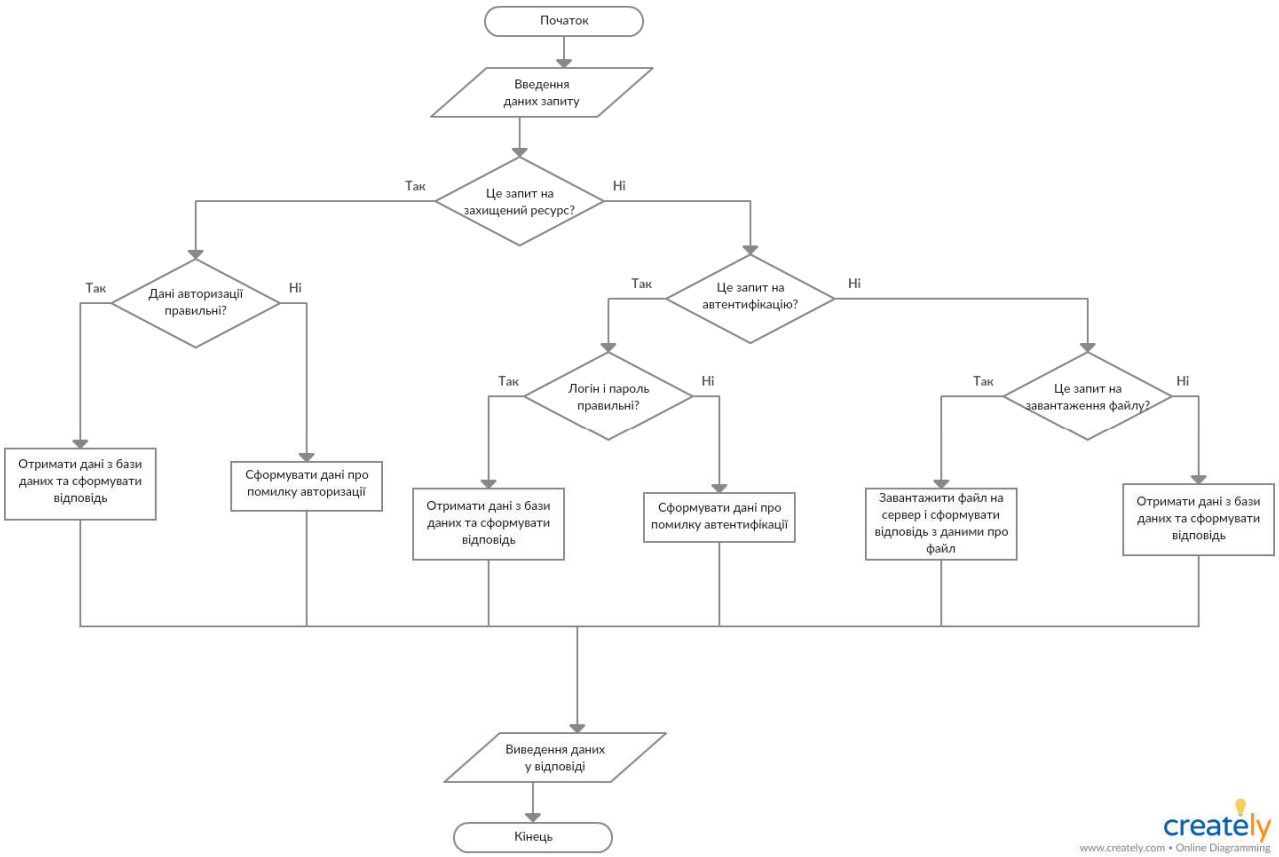

Рисунок 2.1 – Загальний алгоритм роботи системи

У всіх інших випадках система просто формує дані про запитуваний ресурс або ж помилку у випадку невірних вхідних даних.

Останнім кроком є надсилання даних, сформованих на попередніх етапах, клієнту через певний інтерфейс між ними.

Щоб краще зрозуміти, як користувач буде взаємодіяти з системою, варто використати графічне зображення у вигляді діаграми варіантів використання.

Діаграма варіантів використання, або діаграма прецендентів – в стандарті UML діаграма, яка зображає взаємозв’язки між акторами та варіантами використання проектованої системи. [9]

Даний тип діаграми є графом, що складається з акторів, варіантів використання та зв’язків між ними. [10]

У стандарті UML існує кілька типів зв’язків між акторами та варіантами використання:

- асоціації (англ. association);
- включення (англ. include);
- розширення (англ. extend);
- загальнення (англ. generalization) [11].

Загальна суть діаграми використання полягає в тому, щоб представити проектовану систему як множину сутностей чи акторів, які взаємодіють з системоюю за допомогою варіантів використання. Варіант використання являє собою певний набір дій, які повинна виконати система при взаємодії з актором [12].

Проектована система оцінювання викладачів включає в себе кілька акторів, які зможуть працювати з нею, зокрема:

- адміністратор;
- модератор;
- студент;
- викладач;
- гість.

Відповідно до наведених вище акторів будуть розподілені ролі користувачів у системі.

Варіанти використання системи розподілені на три типи, які покривають три різних бізнес-процеси:

- пошук інформації;
- оцінювання викладачів;
- керування інформацією.

Пошук інформації відповідає за отримання інформації про факультети, кафедри та викладачів університету. Дана частина системи в подальшому може інтегруватися зі сторонніми системами університету.

Оцінювання викладачів – це процес додавання оцінки для викладача, він є одним з ключових процесів системи.

Керування інформацією здійснюється персоналом університету і включає в себе оновлення існуючих даних, додавання нових, модерацію та видалення неактуальної інформації. Детальна діаграма варіантів використання зображена на рисунку 3.3.

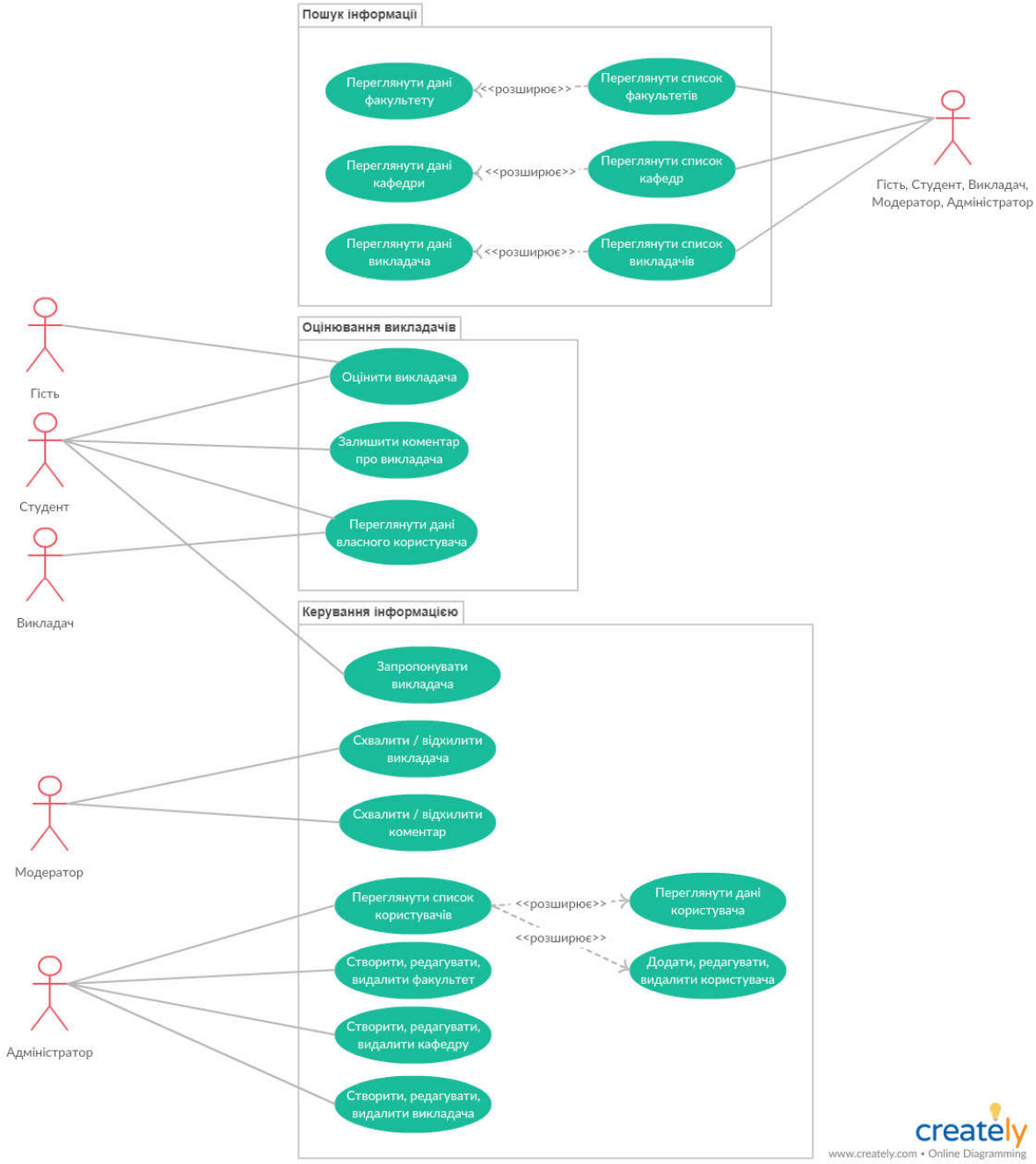

Рисунок 2.2 – Діаграма використання системи

Оскільки система позиціонується як анонімна система оцінювання викладачів, то доступ до оцінювання мають і неавторизовані користувачі (гості). Також оцінювати викладачів можуть авторизовані студенти. До пошуку інформації маюють доступ всі типи акторів. Керування інформацією здійснюється виключно персоналом університету, тобто адміністраторами та модераторами.

2.2 Проектування компонентів системи

Для реалізації програмного рішення необхідно підібрати цілий набір інструментів. Потрібно обрати сервер, технологію та мову програмування, систему управління базами даних та допоміжні компоненти.

В якості сервера можна використати популярний Apache, nginx або ж середовище Node.js. Але оскільки мова програмування та сервер є дуже пов'язаними компонентами, то варто обирати їх разом, а не окремо. Отож, варіантів кілька: PHP, Node.js, Python, Java. Методом аналізу можливостей та недоліків всіх технологій було одразу відкинуто варіанти з PHP та Java. В PHP є певні проблеми з швидкодією при великих навантаженнях, а з Java можуть виникнути проблеми при налаштуванні середовища на сервері.

Вибір між Python та Node.js видався дуже складним. Перший приваблює своєю простотом та різноманіттям вбудованих можливостей, зокрема і для роботи з базою даних. Node.js – це швидкий та потужний інструмент для розробки REST API [13]. Але найбільшим плюсом Node.js є його гнучкість. Розробнику надаються лише базові методи та концепції, все інше може бути реалізоване на власний розсуд. Саме тому в кінцевому результаті було вирішено будувати систему за допомогою Node.js.

Допоміжними компонентами при роботі з Node.js стали веб-фреймворк Express, бібліотека для роботи з базою даних Sequelize, модуль для логування

Bunyan, бібліотека для попередження вразливостей Helmet, а також модуль для аутентифікації JWT JsonWebToken.

Найбільш популярною системою управління базами даних для Node.js є документ-орієнтована MongoDB. З реляційних баз даних можна вибирати між MySQL та PostgreSQL. MongoDB була відкинута через специфіку роботи з нею. Документ-орієнтована база даних буде складною в плані підтримки в майбутньому, оскільки кількість спеціалістів з нереляційних баз даних на ринку доволі невелика [14].

MySQL є, можливо, найпопулярнішою, системою управління базами даних. MySQL проста у встановленні та використанні, вирізняється високою швидкістю виконання команд, а також наявністю ефективної системи безпеки. Втім, і у неї є свої недоліки. Зокрема мова запитів SQL, яка використовується в ній, не повністю відповідає міжнародному стандарту [15]. PostgreSQL, на відміну від попередньої, є ближчою до стандарту і її буде легше підтримувати у подальшому. Крім того, дана система управління базами даних має безліч додаткових можливостей, які можуть бути корисними при масштабуванні системи. Саме тому кінцевим вибором стала PostgreSQL.

Система буде складатися з багатьох компонентів, пов’язаних між собою. Архітектура системи зазначена на рисунку 2.3. Загальний принцип виглядає наступним чином. Клієнт надсилає HTTP-запит на веб-сервер з зазначенням потрібного порту. Якщо порт закріплений за процесом Node.js, то керування передається йому.

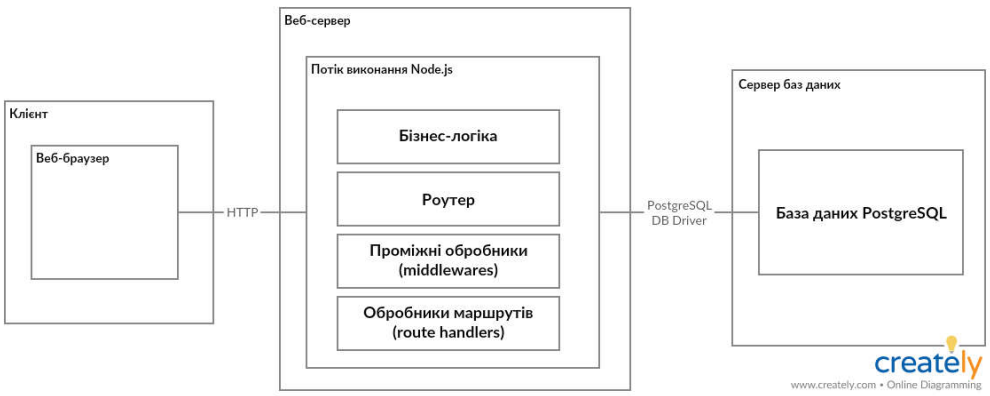

Рисунок 2.3 – Загальна архітектура системи

Першочергово дані запиту надходять на роутер – компонент, який відповідає за аналіз вхідних даних і делегування обробки запиту іншим компонентам. Компоненти системи, які відповідають за кінцеву обробку запиту називаються обробниками маршрутів. Вони перевіряють правильність даних від клієнта, вибирають потрібну інформацію з бази даних або ж формують інформацію про помилку. На проміжному рівні – між роутером та обробником – працюють проміжні обробники (англ. middleware). Дані компоненти виконують певні типові операції над даними запиту, зокрема перевірки авторизації, валідності даних.

Окремо зберігається бізнес-логіка системи. До бізнес-логіки відносяться компоненти, які забезпечують зв’язок з драйвером баз даних, роботу з автентифікацією користувачів, логуванням системних помилок тощо.

Для звернень до бази даних використовується бібліотека Sequelize, яка базується на драйвері pg для PostgreSQL.

2.3 Проектування структури баз даних

При проектуванні бази даних треба брати до уваги специфіку вимог до системи. У проектованій системі необхідно відобразити наступні сутності:

- користувач (User);
- факультет (Faculty);
- кафедра (Department);
- викладач (Lecturer);
- коментар (Comment);
- вчене звання (Academic Status);
- науковий ступінь (Degree).

Для кожної з перелічених сутностей необхідно створити модель з відповідними атрибутами. Моделі, атрибути та зв'язки між ними зображені на рисунку 2.3.

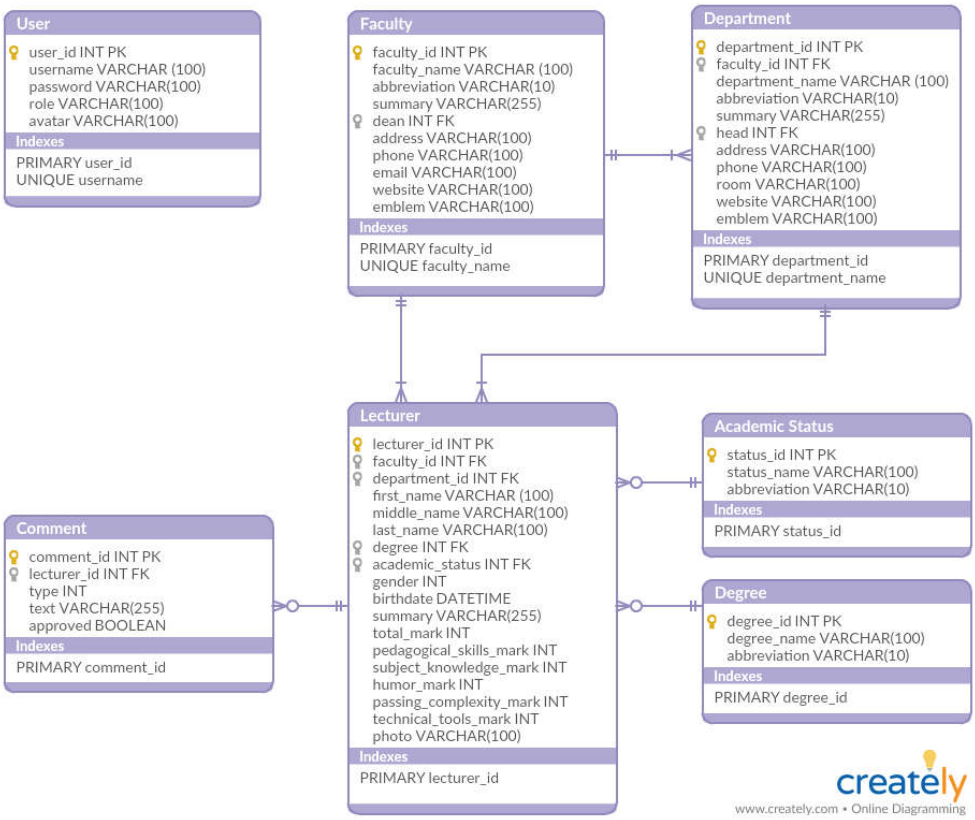

Рисунок 2.4 – Структура баз даних

2.4 Інтеграція з клієнтською частиною

Вибір інтерфейсу для взаємодії між серверним та клієнтським модулями системи не був дуже складним. Варіантів було всього два: REST, SOAP, RPC.

SOAP – це інтерфейс, підвид протоколу RPC, базується на передачі повідомлень у вигляді XML. На даний момент він є застарілим і рідко використовується.

REST (англ. Representational State Transfer) – підхід до архітектури мережевих інформаційних систем. Був описаний і популяризований у 2000 році Роєм Філдінгом, одним з засновників протоколу HTTP.

Отже, система анонімного оцінювання викладачів буде побудована на принципах REST API.

REST – це в першу чергу архітектурний стиль, і тому в нього є певні архітектурні обмеження. Оскільки він базується на клієнт-серверній архітектурі, то і успадковує її обмеження. Зокрема, клієнт-серверна архітектура вимагає розділення відповідальності між модулями, які займаються зберіганням та обробкою даних та презентацією цих даних [16].

Ще одним обмеженням є те, що стан взаємодії між сервером та клієнтом не зберігається. Тобто сервер не отримує даних про поточний стан клієнта. Даний принцип передбачає зберігання даних сесії на клієнті.

Системи в стилі REST повинні підтримувати кешування. Кешування дозволяє збільшити продуктивність системи, уникаючи зайвих запитів. Недоліком такого підходу є те, що дані в кеші можуть бути застарілими та неактуальними.

Як архітектурний стиль, REST містить певні архітектурні елементи. До них належать ресурс, ідентифікатор ресурсу, представлення та контрольні дані.

Ресурсом називають певний об’єкт, який може запитувати клієнт у сервера. Прикладом ресурсу може бути документ, деяке динамічне значення тощо. Ідентифікатор ресурсу – це певна адреса ресурсу, яка дозволяє посилатися на нього. Найпопулярнішими прикладами ідентифікаторів ресурсу є URL та URN [17].

Представлення – це метадані представлення ресурсу, тобто вигляд, в якому його отримає клієнт. Контрольні дані служать для опису даних запиту, опису необхідних операцій, які треба провести над ресурсом, та значення відповіді сервера на запит.

В стандартному розумінні, ресурс може бути будь-чим, однак дії та операції, які можна над ним проводити, описуються стандартним протоколом, в більшості випадків – це протокол HTTP.

Станадарт HTTP визначає 8 типів повідомлень:

- GET – отримати представлення ресурсу;
- POST – стоврити новий ресурс;
- PUT – замінити стан поточному ресурсу;
- DELETE – видалити ресурс;
- HEAD – отримати заголовки без представлення;
- OPTIONS – отримати список методів, актуальних для цього ресурсу;
- CONNECT – для використання з проксі-серверами, які можуть динамічно переключатиися в тунельний режим SSL;
- TRACE – отримати відповідь таким чином, що клієнт побачить, що проміжні сервери змінюють у запиті.

Існує ще один тип, який описується не в самому стандарті, а в додатку RFC 5789 – PATCH. Він дозволяє змінити лише частину ресурсу, а не повністю. Це зменшує кількість даних які необхідно передати.

Методи GET, PUT, DELETE є ідемподентними, тобто незалежно від того скільки разів виконається операція – результат завжди буде однаковий. Ідемподентність є дуже важливою у випадках, коли ви не знаєте чи досягнув запит успіху, і не отримавши відповіді, посилаєте його ще раз.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВЕРНОГО МОДУЛЯ

3.1 Основи процесу розробки програмного забезпечення

Існує дві основні моделі розробки програмного забезпечення: водоспадна, або послідовна, та ітеративна.

Водоспадна модель життєвого циклу передбачає послідовне виконання всіх етапів проекту у строго фіксованому порядку. Перехід на наступний етап означає повне завершення робіт на попередньому етапі. Ця модель виходить з того, що всі помилки будуть зосереджені на реалізації, а тому їх усунення відбувається рівномірно під час тестування системи [18]. Однак на практиці такий підхід важко втілити в життя і він може бути ефективним тільки для створення невеликих систем [19].

Дане програмне забезпечення розроблялося на основі ітеративної моделі розробки. Ітеративний підхід передбачає створення програмного продукту не за один довгий період часу, а за короткі ітерації. Основними перевагами ітеративного підходу є:

- зниження вірогідності впливу серйозних ризиків на ранніх стадіях проекту, що веде до мінімізації витрат на їх усунення;
- організація ефективного зворотнього зв’язку проектної команди з кінцевим користувачем або замовником, і створення продукту, який дійсно відповідає їхнім очікуванням;
- акцент зусиль на найбільш важливі та критичні напрями проекту;
- неперервне ітеративне тестування, яке дозволяє оцінити успішність проекту в цілому;
- раннє виявлення конфліктів між вимогами, моделями і реалізацією проекту;
- більш рівномірне навантаження учасників проекту;
- ефективне використання накопиченого досвіду;

- реальна оцінка поточного стану проекту і, як наслідок, більша впевненість замовників і в його успішному завершенні;

- витрати розподіляються по всьому проекту, а не групуються в його кінці.

Таким чином, програмне забезпечення розроблялося ітеративно, тобто частинами. Схема процесу розробки зображена на рисунку 3.1.

Рисунок 3.1 – Схема процесу розробки програмного забезпечення

3.2 Розробка компонентів системи

При розробці серверного модуля необхідно враховувати специфіку роботи з Node.js та фрейворком Express.

Express - це мінімалістичний та гнучкий фреймворк для веб-додатків, побудованих на Node.js, що надає широкий набір функціональності. Маючи в своєму розпорядженні безліч допоміжних HTTP-методів та проміжних обробників, створювати надійні API можна легко і швидко. Express забезпечує

тонкий прошарок базової функціональності для веб-застосунків, що не спотворює звичну та зручну функціональність Node.js [20].

Express та всі інші потрібні бібліотеки поставляються як окремі модулі менеджера пакетів npm. Тому насамперед необхідно налаштувати менеджер пакетів. Повна конфігурація npm разом з усіма скриптами міститься в Додатку А.

Наступним кроком буде задання структури папок і файлів. Отже, в кореневому каталозі буде міститись файл package.json, який задає налаштування npm, службовий файл системи контролю версій .gitignore, файл конфігурації бібліотеки Sequelize .sequelizerc та службові файли eslint .eslintignore, .eslintrc. Також кореневий каталог містить кілька папок, зокрема папку app, в якій знаходиться основний код для роботи системи, папки migrations та seeders, які відповідають за зберігання міграцій і початкових даних для бази даних, і папку test, в яку помістимо інтеграційні тести.

Детальніше варто зупинитися на папці app. В ній є всього лиш два файли app.js і run.js. Вони відповідають за початковий запуск сервера і обробку критичних помилок. Окрім файлів тут також є такі папки: assets, config, database, forms, libs, middlewares, models і routes.

Перш за все треба винести загальні конфігурації системи в окремі файли. Вони будуть знаходитися в папці app/config. Конфігурації будуть різними залежно від середовища, в якому відбувається робота. Середовище задається в аргументах скрипта при старті сервера. Середовищ у системі буде три:

- development – система запускається на локальній машині розробника;
- production – система запускається на віддаленому сервері і готова для експлуатації кінцевими користувачами;
- test – система запускається на локальному або віддаленому сервері для запуску тестів.

Файли конфігурацій мають розширення .json і зберігають в собі інформацію про термін життя авторизаційних токенів, порт сервера, кореневі

папки для логування та завантаження файлів, а також дані для з'єднання з базою даних PostgreSQL. Файли мають імена відповідно до середовища, для якого застосовуються. Додатково використовується модуль nconf, який відповідає за експорт потрібного конфігураційного файлу в інші модулі системи відповідно до середовища.

Важливим моментом є логування всіх операцій, які відбуваються в системі. Це дозволяє швидко знайти причину помилок при їх виникненні, а також при аналізі методів рефакторингу й оптимізації роботи системи. Для логування використовуємо зовнішню бібліотеку bunyan. Загальні конфігурацій модуля для логування знаходяться у файлі app/libs/logger.js.

Додатково у папці libs потрібно створити кілька власних бібліотек та службових файлів. Файли з закінченням .enum.js слугують для зручності роботи з типами перерахування (enumeration). Бібліотека response-messages.js відповідає за збереження типів помилок та загального формату відповідей про помилки. Останні два файли form.js та validator.js є службовими модулями для обробки форм та валідації вхідних даних.

Наступним кроком є написання модуля роботи з базою даних. Допоміжною бібліотекою для роботи з PostgreSQL було обрано Sequelize. Перевагою бібліотеки Sequelize є автоматичне створення моделей та міграцій, а також можливість керування ними за допомогою скриптів npm. Початкове налаштування цієї бібліотеки записується у файл .sequelizerc у кореневому каталозі. У папці database основним буде файл postgres.js, в який помістимо функцію для з'єднання з базою даних. Для оптимізації взаємодії з сервером бази даних буде використовуватися пул з'єднань (connection pool) – своєрідний кеш для запитів, таким чином з'єднання можуть перевикористовуватися для майбутніх запитів. Пул з'єднань дозволяє значно збільшити продуктивність виконання команд в базі даних, оскільки у високонавантажених системах такі операції можуть бути часо- та ресурсозатратними.

Після першочергових і обов’язкових модулів можна переходити до написання модуля запуску сервера. Цим займається файл app/app.js, який містить функції для безпосереднього запуску сервера, перехоплення фатальних помилок та підключення інших основних модулів, зокрема модулів баз даних, генерації ключів для шифрування токенів аутентифікації, проміжних обробників та обробників маршрутів. Оскільки операції підключення до бази даних та запуску сервера можуть займати відносно багато часу та ресурсів, то вони виконуються асинхронно. Асинхронність реалізована за допомогою вбудованих в JavaScript об’єктів Promise.

У файлі app/run.js записуються функції для розподілу навантаження в багатоядерних процесорах серверів. Детальніше про це у підрозділі 3.3.

Основні компоненти серверного модуля готові і його запуск вже можливий. Тепер необхідно реалізувати функціональність системи.

Почати варто з моделей. Моделі являють собою правила для представлення даних структури у базі даних. Моделі є одних з базових понять концепції ORM (Object-Relational Mapping) – об'єктно-реляційної проекції. Дана концепція передбачає поєднання двох парадигм – об’єктно-орієнтованого програмування та реляційних баз даних. Модель представляється у вигляді об’єкту з властивостями, які позначають атрибути таблиці бази даних. В системі використовуватимуться вже раніше спроектовані моделі. Для генерації файлів моделей у Sequelize існують вбудовані скрипти.

На основі моделей будують форми. Форми являють собою загальні вимоги і критерії для вхідних даних. При надходженні запиту з корисним навантаженням (payload), яке може містити дані авторизації або ідентифікатори ресурсів, сервер повинен перевірити правильність формату цих даних, і в разі невідповідності до зразка надіслати повідомлення про помилку. Зразком тут виступає саме форма. Форми зберігатимуться у папці app/forms.

Важливу роль при обробці вхідного запиту відіграють проміжні обробники (middleware). Концепція проміжних обробників передбачає використання окремих компонентів для зв'язку між компонентами, які отримують запит, і кінцевими обробниками маршрутів. Тобто вони вбудовуються в запит, виконують з ним певні операції і відправляють модифікований запит кінцевому обробнику. В системі будуть реалізовані одразу кілька проміжних обробників:

- auth (перевіряє чи користувач аутентифікований);
- error (виконує логування помилок під час виконання);
- logger (виконує логування всіх вхідних запитів);
- response-extender (розширює можливості вбудованого об'єкта відповіді сервера);
- roles (перевіряє роль та дозволи користувача);
- validator (перевіряє вхідні дані у запитах з корисним навантаженням);
- version (додає до відповіді заголовок Content-Version з актуальною версією системи).

Коли всі моделі, бібліотеки, форми та проміжні обробники реалізовані можна переходити до реалізації обробників маршрутів (route handlers). Якщо провести аналогію з концепцією Model-View-Controller, то обробник маршруту виконує роль вигляду і частково контролера. Кінцевий обробник формує дані перед відправкою клієнту. У фреймворку Express є вбудований клас Router, за допомогою якого можна з легкістю керувати всіма обробниками. Вбудовані механізми дозволяють також використовувати вкладені маршрутизатори, тобто призначати на маршрут не один обробник, а весь маршрутизатор, який зробить можливим обробку всіх похідних маршрутів. Під час побудови схеми маршрутів варто користуватися логікою і загальними практиками побудови REST API.

На найвищому рівні буде реалізовано два маршрутизатора – один для обробки маршрутів, які потребують авторизації, інший – для обробки всіх інших маршрутів. В перший буде включено проміжний обробник auth. На нижчих рівнях працюватимуть маршрутизатори для обробки окремих запитів. В маршрути включаються відповідні проміжні обробники за необхідності. В системі використовуються кілька типів кінцевих точок (endpoint) маршрутів:

- запит на читання списку ресурсів (GET);
- запит на читання окремого ресурсу (GET);
- запит на створення ресурсу (POST);
- запит на модифікацію ресурсу (PUT);
- запит на часткову модифікацію ресурсу (PATCH);
- запит на видалення ресурсу (DELETE) [21].

Алгоритм роботи кінцевого обробника залежить від контексту його використання. В загальному випадку кінцевий розробник витягує дані з запиту і виконує асинхронні операції з моделлю.

3.3 Балансування навантаження

Веб-сервіси можуть приймати одночасно кілька сотень, а то й тисяч запитів від різних клієнтів. Таким чином веб-сервіси – це високонавантажені системи, що від них вимагає максимальної надійності. Для забезпечення такого рівня надійності варто оптимізувати операції, які є найбільш довготривалими та ресурсозатратними. Цього можна досягти за допомогою принципів паралелізму. Паралелізмом називають властивість систем, коли декілька процесів обчислення відбуваються водночас. В мові JavaScript існує зручний механізм асинхронних операцій за допомогою об’єкту Promise. В реалізованій системі цей інструмент використовується повсюди, зокрема:

- операції читання/запису в файл;
- операції звернення до бази даних;
- операції звернення до зовнішніх сервісів.

Node.js позиціонується як платформа, що забезпечує достатню швидкодію серверних додатків, навіть незважаючи на те, що вона є однопоточною. Це відбувається завдяки потужному вбудованому механізму циклу подій (event loop). Згадані вище ресурсозатратні операції в даному циклі є блокуючими операціями, тобто вони блокують потік і є неможливим виконання інших операцій до їх завершення. Інструменти асинхронності дозволяють обійти це обмеження.

Існує ще один ефективний метод балансування навантаження на рівні, близькому до апаратного. Завдяки окремому модулю cluster можна реалізувати систему в якості кластера. За замовчуванням додатки на платформі Node.js виконуються лише на одному логічному ядрі процесора. Модуль cluster дозволяє запустити по додатку для кожного ядра процесора. Це забезпечує додатковий приріст в швидкодії та продуктивності. Це досягається за допомогою досить простого алгоритму. Створюється батьківський процес (master process) і від нього породжуються дочірні процеси (worker). Кількість дочірніх процесів дорівнює кількості логічних ядер процесора. При кластеризації зберігається можливість вивантажити всі складні обчислення у фонові процеси і забезпечити комунікацію між ними через сервер черги повідомлень. Загальна схема зображена на рисунку 3.2.

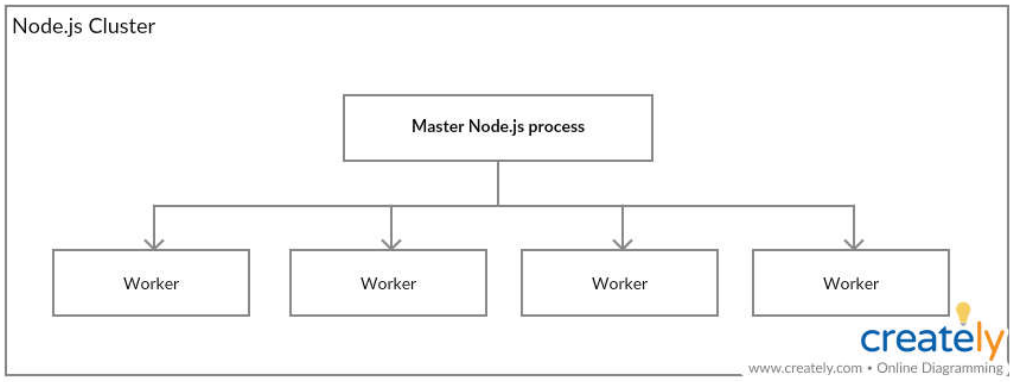

Рисунок 3.2 – Ілюстрація кластера у Node.js

У випадку, якщо визначеного приросту продуктивності буде недостатньо, можна використати окремий сервер у якості балансувальника навантаження. Принцип такого балансування ілюстрований на рисунку 3.3. Одним із найкращих варіантів для такої ролі може стати сервер nginx [22]. Він буде проміжним сервером між клієнтом і серверами Node.js та вирішуватиме на який сервер відправляти запит.

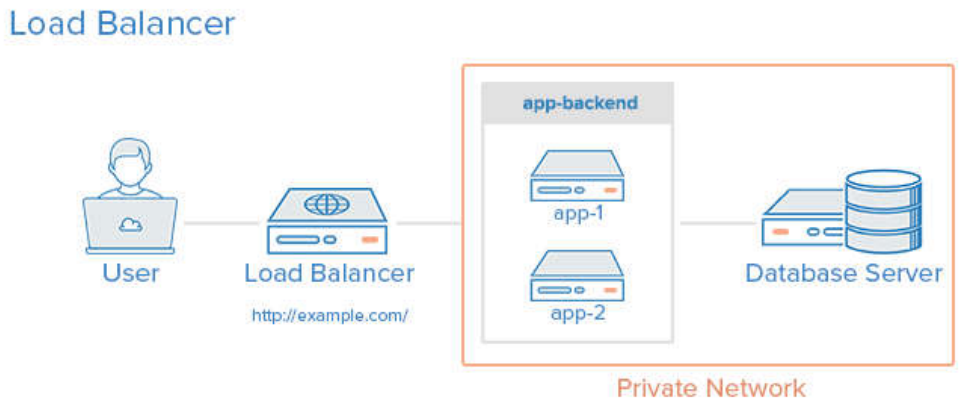

Рисунок 3.3 – Балансування навантаження за допомогою окремого сервера

3.4 Масштабованість системи

Масштабованість – одна з фундаментальних властивостей інформаційної системи. На рівні архітектури програмного забезпечення, масштабованість – це здатність підтримувати великі кількості архітектурних компонентів та з'єданань між ними [23].

Масштабованість у розробці програмного забезпечення тісно переплітається з поняттям «розширюваність». Розширюваність вказує на можливість розширення функціональності системи в разі необхідності [24].

З самого початку система проектувалася таким чином, щоб її можна було масштабувати та розширювати. Виходячи з особливостей архітектури Node.js та фреймворка Express проектовану систему можна розширювати додатковими компонентами та модулями без падіння продуктивності. Ієрархія та структура компонентів спроектована в такий спосіб, щоб додавання нових було максимально простим і швидким. Всі однотипні модулі реалізовувалися за подібним принципом та поміщалися в одну папку. Як приклад, для моделей і форм існують загальні класи, які повинні перевикористовуватися при створенні нових сутностей. Це гарантує уніфікацію коду в межах розробленої системи.

3.5 Тестування розробленого програмного забезпечення

Метою тестування даного програмного забезпечення є виявлення та усунення помилок, допущених на етапі розробки, а також запобігти появі нових дефектів [25]. Тестування необхідно проводити, тому що всі люди роблять помилки. Також помилки можуть виникати і через збої в роботі апаратного забезпечення або через зовнішнє середовище.

Цілями тестування даної системи є:

- знаходження дефектів;
- запобігання появі нових дефектів;
- отримання даних про рівень якості продукту;
- отримання інформації про відповідність продукту очікуванням кінцевих користувачів.

Тестування проводиться в ручному та автоматизованому режимах, як з доступом до коду та документації, так і без нього.

Щоб покрити всі можливі варіанти виникнення помилок, тестування необхідно проводити на кількох рівнях, зокрема на рівні компонентів та на рівні системи. Окремим, але не менш важливим, є інтеграційне тестування,

адже розроблена система буде інтегруватися зі стороннім програмним забезпеченням.

Компонентні та інтеграційні тести повинні бути автоматизованими та покривати як мінімум 80% функціоналу продукту.

Системне тестування проводиться в ручному режимі за допомогою спеціальних інструментів, таких як Postman та Fiddler. Ефективне тестування можливе за умови продумування всіх комбінацій вхідних даних. Для цього можна застосувати техніки, що базуються на структурі та специфікації, такі як умови, рішення, твердження, аналіз граничних значень, розбиття на класи еквівалентності.

Для забезпечення максимальної якості продукту треба проводити функціональне тестування наряду з нефункціональним [26]. З нефункціонального тестування варто звернути увагу на продуктивність, надійність та підтримуваність системи.

Тестування є одним з найважливіших етапів, оскільки воно може надавати об'єктивну, незалежну інформацію про якість програмного забезпечення, ризики відмови, як для користувачів так і для замовників [27].

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

В цьому розділі дипломного проекту проводиться економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання проектного рішення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для візуалізації метричних ультразвукових сигналів і характеризується підвищеною ефективністю виконання алгоритму, що призводить до зменшення часу візуального представлення об’єкту дослідження.

4.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

$$K = K_1, + K_2$$

де K_1 - витрати на розробку програмних засобів, *грн*.

K_2 - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп’ютері, *грн*.

Витрати на розробку програмних засобів включають:

- витрати на оплату праці розробників ($B_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($Вф$);
- витрати на покупні вироби ($Пв$);
- витрати на придбання спецобладнання для проведення експериментальних робіт ($Об$);
- накладні витрати ($Н$);
- інші витрати ($Ів$).

4.1.1 Розрахунок витрат на оплату праці

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту; студент-дипломник; консультант техніко-економічного розділу.

Таблиця 4.1 - Вихідні дані для розрахунку витрат на оплату праці

№ п/п	Посада виконавців	Місячний оклад, грн.
1	Керівник ДП, викладач	4800
2	Консультант техніко-економічного розділу, доцент	5000
3	Студент	1100

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.1)$$

де n_{ij} – чисельність розробників i-ої спеціальності j-го тарифного розряду, *осіб*; t_{ij} – затрачений час на розробку проекту співробітником i-ої спеціальності j-го тарифного розряду, *год*; C_{ij} – годинна ставка працівника i-ої спеціальності j-го тарифного розряду, *грн*.,

Середньо годинна ставка працівника може бути розрахована за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \qquad (4.2)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, *грн*.; *h* – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат); $РЧ_i$ - місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, *год.* (приймаємо *168 год.*).

Результати розрахунку записують до таблиці 4.2.

Таблиця 4.2 - Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, *год*	Погодинна заробітна плата, *грн/год.*	Витрати на розробку, *грн*
1	Керівник ДП, викладач	20,5	28,57	585,685
2	Консультант техніко-економічного розділу, доцент	2	29,76	59,52
3	Студент	144	6,55	943,2
Разом				1588,405

4.1.2 Відрахування на соціальні заходи

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5 % від суми заробітної плати:

$$B_Ф = \frac{20,5}{100} \cdot 1588,405 = 325,62 \, грн.$$

4.1.3 Розрахунок витрат на матеріали та комплектуючі

У таблиці 4.3 наведений перелік купованих виробів і розраховані витрати на них.

Таблиця 4.3- Розрахунок витрат на матеріали та комплектуючі

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна, *грн*	Кількість купованих виробів	Сума, *грн*	Транспортні витрати (10% від суми)	Загальна сума, *грн*
1	Папір (формат А4)	*уп*	45,0	2	90,00	9,0	99,0
2	Ручка кулькова	*шт*	2,0	2	4,00	0,4	4,40
3	Олівець простий	*шт*	1,50	2	3,00	0,3	3,30
4	Диски CD-R	*шт*	2,0	2	4,00	0,4	4,40
5	Зошит, 96 арк	*шт*	3,50	1	3,50	0,35	3,85
6	Тонер для принтера	*уп*	20	1	20	2,0	22,0
Разом							136,95

4.1.4 Витрати на використання комп’ютерної техніки

Витрати на використання комп’ютерної техніки включають витрати на амортизацію комп’ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп’ютером. За даними обчислювального центру ТНЕУ для комп’ютера типу IBM PC/ATX вартість години роботи становить 4,5 грн. Середній щоденний час роботи на комп’ютері – 2 години. Розрахунок витрат на використання комп’ютерної техніки приведений в таблиці 4.4.

Таблиця 4.4- Розрахунок витрат на використання комп’ютерної техніки

№ п/п	Назва етапів робіт, при виконанні яких використовується комп'ютер	Час використання комп'ютера, *год.*	Витрати на використання комп'ютера *грн.*
1	Проведення досліджень та оформлення їх результатів	60	270
2	Оформлення техніко-економічного розділу	8	36
4	Оформлення ДП	12	54
Разом		80	360

4.1.5. Накладні витрати

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати:

$$H = 1{,}5 \cdot 1588{,}405 = 2382{,}61 \ (грн.)$$

4.1.6. Інші витрати

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$I = 1588{,}405 \cdot 0{,}1 = 158{,}84 \ (грн.)$$

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_Ф + B_{ПВ} + H + I$$

$$K_1 = 1588{,}405 + 325{,}62 + 136{,}95 + 2382{,}61 + 158{,}84 = 4572{,}425 \ (грн.)$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від} \qquad (1.5)$$

де $S_{м.г.}$ - вартість однієї машино-години роботи ПК, *грн./год*.

$t_{від}$ - комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, *год*.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 2,5 грн. Тому

$$К_2 = 2,5 \cdot 60 = 150 \text{ } грн.$$

На основі отриманих даних складаємо кошторис витрат на розробку програмного забезпечення.

Таблиця 4.6- Кошторис витрат на розробку програмного забезпечення

№ п/п	Найменування витрат	Сума витрат, *грн*.
1	Витрати на оплату праці	1588,405
2	Відрахування у спеціальні державні фонди	325,62
3	Витрати на куповані вироби	136,95
4	Накладні витрати	2382,61
5	Інші витрати	158,84
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	150,0
Разом		4722,425

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}}$$

де $E_п$ - одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1п}$ - вартість підготовки даних для експлуатації ПЗ (аналогу), грн.;

$E_{2п}$ - вартість роботи комп'ютера для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{еп}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}}$$

де $N_п$ - періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{n} n_i t_i c_i \ ,$$

де *i* - категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i - кількість працівників і-ої категорії, *осіб*.;

t_i - трудомісткість роботи співробітників і-ої категорії по підготовці даних, год.;

c_i - середнього годинна ставка працівника і-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m}$$

де c_i^0 - основна місячна заробітна плата працівника і-ої категорії, *грн.*;

b - коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57;

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен

один працівник, основна місячна заробітна плата якого складає: с° = *1200 грн*. Тоді:

$$c_1 = \frac{1200(1+0{,}57)}{22*8} = 10{,}7 \; грн/год$$

Трудомісткість підготовки даних для проектного рішення складає 1 год., для аналога 1,5 год.

Таблиця 4.7- Розрахунок витрат на підготовку даних та реалізацію проектного рішення на комп'ютері

№	Час роботи співробітників, *год*.	Середньогодинна заробітна плата, *грн./год*.	Витрати , *грн*.
	Проектне рішення		
1	1	10,7	10,7
	Аналог		
1	1,5	10,7	16,05

Витрати на експлуатацію комп'ютера визначається за формулою:

$$E_{2П} = t * S_{МГ}$$

де t - витрати машинного часу для реалізації проектного рішення (аналогу), *год*.;

$S_{МГ}$ - вартість однієї години роботи комп'ютера, *грн./год*.

$E_{2п} = 1*2{,}5 = 2{,}5$ грн.; $E_{2а}=1{,}5 \cdot 2{,}5 = 3.75$ грн.

$E_п=10{,}7+2{,}5=13{,}2$ грн.; $E_а=16{,}0+3{,}75 = 19{,}75$ грн

.

$B_{еп}= 13{,}2*252=3326{,}4$ грн.; $B_{еа}=19{,}75*252=4977$ грн.

4.3 Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

$$Ц_{С(П)} = Ц_{П} + B_{(E)NPV}$$

де Цп - ціна придбання проектного рішення, *грн.*:

$$Ц_{П} = K(1 + \frac{П_{Р}}{100}) + K_{0} + K_{к}$$

де К - кошторисна вартість;

Пр - рентабельність;

$К_о$ - витрати на прив'язку та освоєння проектного рішення на конкретному об'єкті, *грн*.;

$К_к$ - витрати на доукомплектування технічних засобів на об'єкті, *грн*.;

$$Ц_{Д} = 4722{,}475 \cdot (1 + 0{,}3) = 6139{,}22 \ (грн.).$$

Вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

$$B_{епрv} = \sum_{t=0}^{T} \frac{B_{еП}}{(1+R)^{t}}$$

де $В_{еп}$ - річні експлуатаційні витрати, грн.;

Т - строк служби проектного рішення, років;

R - річна ставка проценту банку.

$$B_{епрv} = \sum_{t=1}^{5} \frac{3326{,}4}{(1+0{,}08)^{t}} = 13272{,}3 \text{ грн.}$$

$$B_{епрv} = \sum_{t=1}^{5} \frac{4977}{(1+0{,}08)^{t}} = 19858{,}2 \text{ грн.}$$

Тоді ціна споживання проектного рішення дорівнюватиме:

$Ц_{сп}$ = 6139,22+13272,3=19411,52 грн.

Аналогічно визначається ціна споживання для аналогу:

$$Ц_{са} = 3500,0+19858,2=23358,2 \text{ грн.}$$

4.4 Визначення показників економічної ефективності

Економічний ефект в сфері проектування рішення:

$$E_{ПР} = Ц_{П} - Ц_{А}$$

$$Е_{ПР} =6139,22– 3500,0 = 2639,22 \text{ грн.}$$

Річний економічний ефект в сфері експлуатації:

$$E_{КС} = B_{ЕА} - B_{ЕП}$$

$$Е_{кс}=4977-3326,4=1650,6 \text{ грн.}$$

Додатковий економічний ефект у сфері експлуатації:

$$\Delta E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t}$$

$$\Delta E_{екс} = \sum_{t=1}^{5} 1650,6(1+0,08)^{5-t} = 9656,01 \text{грн.}$$

Сумарний ефект складає:

$$E = E_{пр} + \Delta E_{екс} = 2639,22+9656,01=12295,23 \text{ грн.}$$

Таблиця 4.8 - Показники економічної ефективності проектного рішення

№	Найменування	Одиниці вимірювання	Значення показників	
			Базовий варіант	Новий варіант
1	Капітальні вкладення	*грн.*	-	3350,44
2	Ціна придбання	*грн.*	3500,0	6139,22
3	Річні експлуатаційні витрати	*грн.*	19858,2	13272,3
4	Ціна споживання	*грн.*	23358,2	19411,52

5	Економічний ефект в сфері проектування	*грн.*	-	2639,22
6	Економічний ефект в сфері експлуатації	*грн.*	-	1650,6
7	Додатковий ефект в сфері експлуатації	*грн.*	-	9656,01
8	Сумарний ефект	*грн.*	12295,23	

4.5 Висновки

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Здійснено порівняння з існуючим аналогом, і цим показано, що дане проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентноздатним. Крім того, отримано економічний ефект у розмірі 12295,23 грн. і тому розробка і впровадження цього проектного рішення є економічно доцільними.

ВИСНОВКИ

Спроектовано базу даних що містить наступні сутності:

- користувач (User);
- факультет (Faculty);
- кафедра (Department);
- викладач (Lecturer);
- коментар (Comment);
- вчене звання (Academic Status);
- науковий ступінь (Degree).

Для кожної з перелічених сутностей створено модель з відповідними атрибутами.

Сформовано основні функціональні вимоги для серверного модуля системи оцінювання викладачів: надсилання клієнту даних про викладачів, факультети та кафедри, поточні оцінки викладачів, перевірка достовірності клієнта, а також його прав на читання чи зміну існуючих даних. Відповідно до наведених вимог побудовано загальний алгоритм роботи системи.

Програмне забезпечення розроблено на основі ітеративної моделі розробки. Ітеративний підхід передбачає створення програмного продукту не за один довгий період часу, а за короткі ітерації. При розробці серверного модуля враховвано специфіку роботи з Node.js та фрейворком Express.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кононенко О.Є. Атестація педагогічних працівників: нормативи, методичні рекомендації, документи / О.Є. Кононенко. – Харків: Основа, 2008.
2. Концепція національного виховання // Освіта. - 1994. — 26 жовтня.
3. Кремень В. Г. Філософія освіти XXI століття /В.Г. Кремень // Освіта України. – 2002. – № 103. – С. 6–7.
4. Концептуальні засади демократизації та реформування освіти в Україні: Педагогічні концепції. - К.: «Школяр», 1997. - 148 с.
5. Андрій Гуляйницький. Системна криза вищої освіти України: формування периферійної моделі // Спільне. — 27.10.2017.
6. Бабінець. С. До питання моніторингу як засобу прогнозування педагогічної діяльності / С. Бабінець // Освіта і управління : Науково-практичний журнал. – 2003. – Том 6, № 3.
7. Казаріцька Т. Компетентність вчителя: інструментарій оцінки та самооцінки / Т. Казаріцька // Директор школи. – 2004. – № 6.
8. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. — 3. — MIT Press, 2009. — 1312 с.
9. Albert Endres, Dieter Rombach. A Handbook of Software and Systems Engineering. — Addison Wesley, 2003.
10. Микола Глибовець. Основи комп'ютерних алгоритмів. — Видавничий дім «Києво-Могилянська Академія», 2003. — 452 с.
11. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. — Пер. с англ. — М.: ДМК, 2000. — 432 с.
12. Крег Ларман. - Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. — 3-е вид. — М.: Вільямс, 2006. — 736 с.

13. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с.

14. Кайл Бэнкер. MongoDB в действии. — ДМК Пресс, 2014. — 394 с.

15. В. Васвани. MySQL: использование и администрирование. — М.: «Питер», 2011. — 368 с.

16. Richardson, Leonard; Amundsen, Mike; Ruby, Sam (2013). RESTful Web APIs

17. A Short History of "Resource" in web architecture. [Електронний ресурс] / Tim Berners-Lee – 2018. – Режим доступу: https://www.w3.org/DesignIssues/TermResource.html. – Назва з екрану.

18. Брукс Ф. Міфічний людино-місяць, або як створюються програмні системи: пер. з англ. / Ф. Брукс. — Санкт-Петербург : Символ-Плюс, 1999. — 304 с

19. Мірошниченко Е. А. Технології програмування: навчальний посібник / Е. А. Мірошниченко. — 2-е вид. — Томськ: Вид-во Томського політехнічного університету, 2008. — 128 с.

20. Express – фреймворк для веб-застосунків, побудованих на Node.js [Електронний ресурс] / Node.js Foundation – 2018. – Режим доступу: http://expressjs.com/uk. – Назва з екрану.

21. Mark Masse, REST API Design Rulebook / Mark Masse – O'Reilly Media, Inc., 2011 – 26 с.

22. Choosing an NGINX Plus Load-Balancing Technique [Електронний ресурс] / Tony Mauro. – Режим доступу: https://www.nginx.com/blog/choosing-nginx-plus-load-balancing-techniques/ .

23. André B. Bondi, 'Characteristics of scalability and their impact on performance', Proceedings of the 2nd international workshop on Software and performance, Ottawa, Ontario, Canada, 2000, с. 195-203.

24. Fielding Roy. Architectural Styles and the Design of Network-based Software Architectures. — University of California, Irvine, 2000.

25. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев: ДиаСофт, 2001. — 544 с.

26. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. — СПб.: Питер, 2004. — 320 с.

27. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc (1999). Testing Computer Software, 2nd Ed. New York, et al: John Wiley and Sons, Inc. c. 48