Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка підсистеми автоматизації роботи відділу кадрів

Текстова частина до кваліфікаційної роботи
за спеціальністю “Інженерія програмного забезпечення”

Керівник кваліфікаційної роботи
доктор технічних наук, професор,
Глибовець А. М.

_____
(підпис)
“___” _____ 2025 р.
Виконав студент Франків Я.О.
“___” _____ 2025 р.

Київ 2025

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
доцент, кандидат наук
_____ Гороховський С.С
(підпис)
“____” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу
студенту Франківу Ярославу Олександровичу 3-го курсу факультету інформатики
ТЕМА: Розробка підсистеми автоматизації роботи відділу кадрів:

Індивідуальне завдання

Анотація

Вступ

1. Аналіз предметної області та постановка задачі
2. Проєктування підсистеми
3. Реалізація підсистеми
4. Тестування та оцінка результатів

Висновок

Список використаних джерел

Додатки

Дата видачі “___” _____ 2025 р.
Керівник _____ Завдання отримано _____

Календарний План Виконання Кваліфікаційної Роботи

Тема: Розробка підсистеми автоматизації роботи відділу кадрів

№ з/п	ПЕРЕЛІК РОБОТ	Термін Виконання	Примітка
1	Отримання теми кваліфікаційної роботи	Листопад - Грудень 2024	
2	Створення плану роботи	Кінець грудня 2024	
3	Ознайомлення з проєктом	Січень 2025	
4	Розробка практичної частини проєкту	Лютий - Березень 2025	
5	Написання першого розділу роботи	Початок квітня 2025	
6	Описання практичної частини роботи	Середина квітня 2025	
7	Перегляд змісту роботи з керівником	Кінець квітня 2025	
8	Внесення змін відповідно до зауважень наукового керівника	Кінець квітня 2025	
9	Створення презентації	Початок травня 2025	
10	Захист курсової роботи	13.05.2025	

Студент Франків Я.О. _____

Керівник Глибовець А.М. _____

“_____” _____ 2025 р.

ЗМІСТ

Анотація

У курсовій роботі розглянуто процес розробки підсистеми автоматизації роботи відділу кадрів в рамках інформаційної системи SmartUKMA. Метою підсистеми є спрощення обліку навчального навантаження викладачів, автоматизація обчислень, а також забезпечення прозорості та контролю кадрових процесів.

У ході реалізації було проаналізовано наявну ручну модель розподілу навантаження, визначено її недоліки та спроєктовано новий підхід на основі мікросервісної архітектури. Серверна частина розроблена з використанням Java, Spring Boot і Hibernate, клієнтська частина – на Angular. Автоматичні обчислення впроваджено через MapStruct-мапери, що дозволяє уникнути дублювання інформації та зберегти консистентність даних.

Також реалізовано рольову модель доступу, що гарантує розмежування прав користувачів та захищає від несанкціонованих змін. Підсистема протестована вручну, результати підтвердили коректність реалізованих функцій.

Отримане рішення демонструє значне підвищення ефективності кадрових процесів, зниження ймовірності помилок та є гнучкою основою для подальшого розширення функціональності.

ВСТУП

Розвиток інформаційних технологій докорінно змінює управлінські процеси у закладах вищої освіти. Відділ кадрів – один із ключових підрозділів університету, що забезпечує повний цикл роботи з персоналом: прийом на посаду, управління навантаженням викладачів, облік трудових відносин, формування наказів і звітності. Сьогодні більшість цих операцій здійснюється за допомогою розрізнених таблиць, текстових шаблонів і електронної пошти, що призводить до дублювання даних, помилок у розрахунках та затримок у прийнятті рішень. Автоматизація кадрової підсистеми здатна усунути ці проблеми, забезпечивши єдине джерело достовірної інформації, швидкі алгоритмічні обрахунки (аудиторне навантаження, загальну кількість викладацьких годин тощо) та прозорість усіх кадрових процедур. В умовах інтеграції з цифровою екосистемою SmartUKMA така підсистема стає невід’ємною складовою модернізації університетського менеджменту та виконання ліцензійних вимог Міністерства освіти і науки України.

Метою курсової роботи є розробка концепції та прототипу підсистеми автоматизації роботи відділу кадрів університету, що спиратиметься на сучасні web-технології й інтегруватиметься з наявними інформаційними сервісами. Для досягнення цієї мети необхідно розв’язати такі завдання:

– проаналізувати існуючі процеси обліку персоналу й визначити їх слабкі місця;

– сформулювати функціональні та нефункціональні вимоги до підсистеми;

– спроєктувати архітектуру рішення, включно з ER-діаграмою бази даних і схемою взаємодії модулів;

– реалізувати ключові модулі (облік дисциплін, автоматичний підрахунок годин, розподіл груп і студентів, поля для похідних значень);

– продемонструвати інтеграцію з системою SmartUKMA та оцінити ефективність запропонованого підходу порівняно з ручною обробкою.

Об’єктом дослідження є електронна система управління університетом SmartUKMA, а предметом – технології та методи автоматизації кадрових процесів, що забезпечують безпомилкове зберігання, обробку й актуалізацію даних персоналу.

Робота складається з вступу, п’яти розділів, висновків, списку використаних джерел та додатків. Перший розділ містить огляд поточного стану системи SmartUKMA й аналіз ручних процедур, що потребують автоматизації. У другому розділі поставлено задачі та визначено вимоги до підсистеми. Третій розділ присвячено проєктуванню архітектури й бази даних. Четвертий розділ описує реалізацію основних модулів та інтеграцію зі смарт-екосистемою. П’ятий подає методику тестування, результати експериментів і оцінку ефективності. У висновках підбиваються підсумки виконаної роботи та окреслюються перспективи подальшого розвитку системи.

Таким чином, запропонована підсистема автоматизації відділу кадрів покликана підвищити оперативність управління персоналом, знизити ризики помилок і сприяти стратегічному розвитку університету в умовах цифрової трансформації.

РОЗДІЛ 1: Аналіз предметної області та постановка задачі

1.1 Огляд існуючої системи SmartUKMA

SmartUKMA — це корпоративна інформаційна платформа, яку НаУКМА розвиває з 2019 р. для інтеграції навчального, наукового й адміністративного відділів. Вона побудована за мікросервісною моделлю: кожна бізнес-функція винесена в окремий spring-boot-сервіс, що спілкується з іншими через HTTP використовуючи REST-інтерфейси. Базові служби — account-server (єдина автентифікація та авторизація через Azure AD), info-server (довідкові дані про користувачів і підрозділи) і ui-server (шлюз для зовнішніх запитів) — забезпечують централізоване управління доступом, авторизацію за корпоративним обліковим записом і прозору маршрутизацію запитів. На цьому каркасі розгорнуті навчальні та адміністративні сервіси: user-info – сервер інформації про працівників та студентів, document-server – система документообігу університету і document-generator – сервер для генерації шаблонних документів.

Дані зберігаються багатошарово: у Postgres – структуровані дані; в ElasticSearch – повнотекстові індекси документів; на файловій підсистемі – фізичні PDF-файли й мультимедійні ресурси. Така архітектура забезпечує масштабованість, модульність і швидке розгортання оновлень завдяки CI/CD Gitlab Runner.

Незважаючи на технологічну готовність платформи, кадровий відділ залишається «цифровою прогалиною». Відділ кадрів веде Excel-реєстри, а витяги про штат та навантаження завантажує в систему із затримкою; сумарні аудиторні й викладацькі години підраховуються вручну; методисти не отримують миттєвої валідації введених даних; викладачі бачать власне навантаження лише після ручної обробки даних методистом. Поточний модуль документообігу охоплює лише загальноуніверситетські накази й розпорядження, не маючи спеціалізованих форм для кадрових документів, наприклад при

прийомі на роботу нового працівника. У результаті зростає ризик помилок, дублювання та людської похибки.

Отже, незважаючи на те, що архітектура SmartUKMA забезпечує можливість легко інтегрувати нові сервіси, відсутність цілісної кадрової підсистеми стримує подальшу цифрову трансформацію університету. Розробка спеціалізованого модуля автоматизації роботи відділу кадрів є логічним наступним кроком, що усуне перелічені обмеження й поліпшить вже створену екосистему.

1.2 Ручні процеси, які підлягають автоматизації

Нині більшість кадрово-педагогічних даних університет збирає й опрацьовує у вигляді поширених «ручних» практик: Excel-таблиць, локальних шаблонів Word і ланцюжків електронних листів. Первинне заповнення інформації про дисципліну починає методист кафедри: він вносить кількість лекційних, семінарських, практичних та лабораторних годин, зазначає обсяг самостійної роботи, кількість груп і студентів. Похідні показники – «Всього аудиторних занять» та «Всього годин дисципліни» – обраховуються у формулі Excel; однак під час подальших коригувань (наприклад, коли змінюються лекційні години чи додається нова група) часто приходиться вносити зміни вручну.

Коли дисципліна розподіляється між кількома викладачами, методист далі копіює дані в окремий аркуш, де для кожного викладача з’являються колонки «кількість груп» і «кількість студентів». Після заповнення цих двох числових полів користувач має власноруч переобчислити консультації, письмові роботи, залік чи екзамен – залежно від типу контролю. Суму годин працівник кафедри, знову ж таки, підсумовує формулами, що потребують зміни кожного разу, коли викладачі обмінюються підгрупами або кількість студентів уточнюють після завершення набору. Контроль того, чи зведені «Загальна кількість груп» і «Загальна кількість студентів» відповідають сумі індивідуальних значень,

здійснюється візуально: число, яке «зійшлося», методист підфарбовує зеленим кольором. Усі відхилення він змушений відстежувати вручну.

Аналогічно формуються індивідуальні «кваліфікаційні картки» викладачів. Кожного семестру працівник відділу кадрів переносить до карти дані про всі посади, ставки, консультації, письмові й контрольні роботи. Загальна кількість годин професора виводиться за формулою, що об’єднує дванадцять різнорідних складників (лекції, семінари, лабораторні, консультації, письмові роботи, заліки, екзамени, теза, практика, офісні години, керівництво і рецензування, робота в ДЕК). Ці обчислення виконують локально, після чого файл надсилають електронною поштою до бухгалтерії; подальші зауваження повертаються знову у вигляді виправленого файлу. Відсутність єдиної системи управління цими процесами тягне за собою типові проблеми: неузгоджені редакції, втрачені листи, дублікати записів, неможливість швидко відстежити, хто і коли змінив показник.

Створення штатного наказу відбувається поза SmartUKMA: відповідальний співробітник копіює затверджені підсумкові дані до шаблону Word, вручну заповнює прізвища, дати, номери і друкує наказ на підпис. Після підписання виникає ще одне коло копіювання: з наказу числа знову повертаються у табель обліку робочого часу. Через неоднакове форматування (десяткові коми, тисячні розділювачі, скорочені назви посад) трапляються помилки округлення, що виявляються вже під час нарахування заробітної плати.

У такий самий спосіб формуються регламентні звіти для МОН, ЄДЕБО і Державної служби статистики. Кожен новий запит надається в окремому шаблоні, і працівники у стислий строк переносять потрібні рядки, коригуючи формули під поточні вимоги. Відсутність інтеграції з ядром SmartUKMA унеможливлює оперативне формування вибірок і робить університет вразливим до людського чинника.

Підсумовуючи, процеси, що нині виконуються вручну, охоплюють: початкове внесення та багаторазове дублювання даних, ад-hoc-обчислення ключових показників, візуальну валідацію збігів, ручну підготовку наказів та звітів і постійне переміщення файлів між підрозділами. Автоматизація цих ланок через спеціалізовану кадрову підсистему дозволить використовувати єдину «точку істини», забезпечити миттєві алгоритмічні підрахунки й валідацію, мінімізувати повторне введення інформації та гарантувати цілісність і актуальність даних на всіх етапах життєвого циклу кадрової інформації.

1.3 Проблеми, що виникають при ручній роботі

Фрагментарне ведення кадрових та навчально-методичних даних у розрізнених електронних таблицях і текстових шаблонах породжує комплекс організаційних та технічних ускладнень, які істотно знижують ефективність управління ресурсами університету. Насамперед спостерігається неузгодженість даних: одна й та сама величина (наприклад, кількість аудиторних годин або студентів у групі) дублюється у кількох файлах і коригується різними особами, що неминуче призводить до розходжень між версіями. За відсутності транзакційної цілісності неможливо достеменно визначити, яка з копій є актуальною, а зіставлення показників перетворюється на трудомісткий ручний аудит.

Другою системною вадою є помилки обрахунків. Формули в Excel-аркушах часто допрацьовують «у польових умовах» — додають нові стовпці, змінюють діапазони, копіюють між файлами. Через це кожен Excel-файл зберігає надлишкову інформацію, а щоб змінити значення одного поля, наприклад кількість лекційних годин, приходиться редагувати декілька файлів, для їх узгодження. Проблему ускладнює відсутність автоматичних механізмів перевірки: методист може помітити похибку лише після формування тарифікаційного списку або, що ще гірше, на етапі нарахування заробітної плати.

Третій клас проблем стосується часових втрат і надмірних комунікацій. Переміщення файлів електронною поштою створює довгі ланцюжки “reply-all”, у яких важко простежити, хто і коли вніс зміну. Кожен уточнений наказ потребує перегенерування, повторного друку, повторного підпису й повторної розсилки, а отже затримує увесь бізнес-процес. У пікові періоди (початок семестру, тарифікаційна кампанія) відділ кадрів витрачає левову частку робочого часу саме на ручне звіряння чисел і консигнацію документів.

Окремо виокремлюються ризики невідповідності нормативним вимогам. Державна звітність (ЄДЕБО, фінансово-бухгалтерські форми, кадрова статистика) потребує точних та своєчасних даних. Будь-яка арифметична неточність або зміщення дати у шаблоні Word трактуються як формальна помилка і можуть призвести до приписів контролюючих органів, затримки фінансування чи необхідності перездавати звіти.

Велика кількість ручних процедур не забезпечує належної прозорості. Університет не має централізованого журналу змін, згідно з яким можна було б встановити, хто саме змінив параметр дисципліни чи підкоригував навантаження викладача. Крім того, відсутність аудиту ускладнює внутрішній контроль, а також підвищує ймовірність помилок.

Таким чином, ручна модель управління кадрами породжує затримки, помилки і надлишкові трудовитрати, знижує достовірність звітності й збільшує регуляторні ризики. Усунення цих проблем можливе лише шляхом переходу до інтегрованої підсистеми, яка забезпечить централізоване зберігання даних, автоматичні розрахунки, валідацію в режимі реального часу та повний історичний аудит змін.

1.4 Функціональні можливості майбутньої підсистеми

Проєктована підсистема покликана стати кадровим ядром SmartUKMA і забезпечити наскрізний цикл обробки інформації про персонал — від

первинного внесення даних до формування регламентних звітів. Її функціональні можливості умовно об’єднуються у п’ять взаємопов’язаних груп.

1. Єдине джерело даних про співробітників і навчальні дисципліни. У централізованій базі зберігатимуться особові, кваліфікаційні та викладацькі картки; історія посад і ставок; належність до структурних підрозділів; участь у навчальних програмах. Кожен запис матиме унікальний внутрішній ідентифікатор, що дозволить однозначно зв’язати викладача з дисциплінами, навантаженням і фінансовими показниками.
2. Автоматичні обрахунки та миттєва валідація. Для дисциплін система самостійно підсумовуватиме «Всього аудиторних занять» і «Всього годин дисципліни» за формулами *Аудиторні = лекції + семінари + практичні/лабораторні*; *Загальні = аудиторні + самостійна робота*. Коли методист розподіляє дисципліну між викладачами та вводить кількість груп і студентів, система негайно переобчислює консультації, письмові роботи, залік або екзамен згідно з типом контролю; похідні поля відображаються лише для читання, що виключає ручне втручання й помилки формул. Крім того, платформа порівнює введені суми з агрегованими значеннями дисципліни й підсвічує збіги зеленим індикатором, а розбіжності — червоним.
3. Модуль навантаження викладача і кадрові обліки. У кваліфікаційній картці автоматично акумулюватиметься *Загальна кількість годин викладача* як сума дванадцяти складових (лекції, семінари, лабораторні, консультації, письмові роботи тощо). Система відстежуватиме перевищення нормативів і генеруватиме попередження. На основі тих самих даних формуватимуться табелі, штатні накази та тарифікаційні списки; після затвердження інформація потраплятиме до фінансового модуля без дублювання вручну.
4. Розширений документообіг і генерація шаблонів. Для всіх типових кадрових документів (накази про прийом/звільнення, контракти, довідки)

буде реалізовано бібліотеку шаблонів із авто-заповненням реквізитів. Підписання здійснюватиметься ЕЦП у рамках існуючого документ-management-сервісу, а стан погодження відображатиметься у реальному часі. Будь-які зміни логуються й доступні в історичному журналі.

5. Ролі, безпека й інтеграція. Доступ до функцій регламентується ролями Azure AD; викладач бачить тільки власні картки, методист – дані його кафедри, а відділ кадрів – повний обсяг інформації. Через REST-API новий модуль взаємодіятиме з наявними сервісами SmartUKMA (info-server, document-generator, notifications-server), що забезпечить єдину точку входу та уніфікований UI. Система надсилатиме електронні сповіщення про критичні події: завершення дедлайнів, відхилення показників, появу нових наказів.

Завдяки зазначеним можливостям підсистема усуває ручне дублювання даних, автоматизує ключові розрахунки, скорочує час підготовки документів і звітів, підвищує прозорість кадрових процесів та формує надійну аналітичну базу для стратегічного планування університету.

1.5 Користувачі та їх ролі

Експлуатаційна модель підсистеми будується на ієрархії доступу, в межах якої кожна функція відкривається тільки тим категоріям працівників, яким вона потрібна для виконання посадових обов’язків. Найвищий рівень прав має системний адміністратор: він налаштовує інтеграційні шлюзи, реєструє та коригує ролі у службі корпоративної автентифікації, відповідає за безперервність роботи сервісів і резервне копіювання.

Співробітник відділу кадрів із розширеними повноваженнями відповідає за створення й закриття особових, кваліфікаційних і викладацьких карток, підтверджує зміни у штатному розписі, запускає масові перерахунки навантаження та формує регламентні звіти; саме на ньому лежить ведення офіційної історії посад і ставок.

Методист кафедри працює у площині дисциплін: заповнює кількість лекційних, семінарських, лабораторних годин, груп і студентів, розподіляє навантаження між викладачами, стежить за коректністю формул і балансу годин, але не змінює кадрові реквізити співробітників. Викладач у своєму кабінеті бачить підсумовані аудиторні й позааудиторні години, консультації, письмові роботи, автоматично згенеровані залікові чи екзаменаційні відомості; його можливості редагування обмежуються внесенням фактичних результатів контролю та завантаженням підтверджувальних матеріалів.

Декан або завідувач кафедри має повний читальний доступ до даних свого підрозділу й право погоджувати навантаження, ініціювати накази про сумісництво чи погодинну роботу, однак кінцеве затвердження залишається за працівником відділу кадрів. Працівники бухгалтерії отримують односпрямований канал: після затвердження тарифікацій інформація про ставки й підсумкові години автоматично передається в облікову систему, проте зворотного впливу на кадрову базу бухгалтери не мають.

Внутрішні та зовнішні ревізори (служба внутрішнього аудиту, Державна служба якості освіти, акредитаційні комісії) користуються режимом тільки-для-читання з мінімально необхідним набором персональних даних; така модель гарантує незмінність первинної інформації протягом перевірки. Інтеграційні сервіси SmartUKMA одержують окремі службові обліковки, через які запитують JSON-представлення карток; авторизація відбувається за токенами з коротким строком дії та чітко визначеними правами.

Під час автентифікації система підтягує дані про приналежність користувача до певних груп у каталозі корпоративної автентифікації й переводить їх у деталізований перелік дозволів на дії: створення картки, оновлення годин дисципліни, погодження навантаження, підпис електронним підписом тощо. Такий підхід забезпечує дотримання принципу найменших привілеїв, прозорий аудит (кожна транзакція журналюється разом із хешем

підпису й часовою міткою) та гнучке масштабування: нові функції можна вводити, просто додавши відповідні дозволи без зміни коду. У підсумку підсистема віддзеркалює реальну організаційну структуру університету, мінімізує конфлікти доступу та формує надійну основу для подальшої цифрової трансформації кадрових процесів.

1.6 Вимоги до підсистеми

Проєктована кадрова підсистема має цілком відповідати організаційним, технологічним і регуляторним рамкам НаУКМА, а також легко інтегруватися у чинну мікросервісну екосистему SmartUKMA. На рівні функціоналу система повинна забезпечити повний життєвий цикл даних про персонал і навчальне навантаження: створення та валідацію карток, автоматичні арифметичні розрахунки, маршрути погодження й підписання ЕЦП, ведення історії змін, формування штатних наказів і регламентних звітів. Кожна бізнес-операція має виконуватися транзакційно з гарантією цілісності; жодна дія користувача не допускається без збереження аудит-запису, що містить часову мітку, IP-адресу та ідентифікатор підпису. Підсистема повинна підтримувати асинхронні сповіщення (e-mail і push-нотифікації) про критичні події: наближення дедлайнів, розбіжності у сумарних годинах, появу нових або скоригованих наказів.

З точки зору інтеграції кадровий модуль зобов’язаний працювати на єдиній схемі авторизації Azure AD та приймати JWT-токени, що вже використовуються в інших сервісах SmartUKMA; усі зовнішні виклики реалізуються через REST + JSON із суворою версіонізацією URI.

Нефункціональні вимоги охоплюють високу доступність і горизонтальне масштабування через Kubernetes-кластер. Конфіденційні персональні дані мають зберігатися зашифрованими at-rest (AES-256) і передаватися лише через TLS 1.3; будь-яке масове експортування інформації можливе виключно за умови багатофакторної авторизації та детального журналювання. Система повинна

відповідати вимогам Закону України «Про захист персональних даних» і рекомендаціям НБУ щодо кіберстійкості, а також забезпечувати зберігання архівних записів протягом щонайменше 5 років із можливістю юридично значимого відтворення змісту.

З погляду експлуатації передбачено zero-downtime Deploy через GitLab CI/CD; кожна зміна коду супроводжується автотестами, що покривають всю критичну бізнес-логіку. Інтерфейс користувача реалізується у спільному концепті UI-server і підпорядковується гайдлайнам доступності WCAG 2.1 AA; локалізація українською і англійською мовами має бути підтримуватись для пошуку, сортування і роботи із сутностями.

Таким чином, підсистема має поєднувати повноту функцій, сувору безпеку, масштабовність і зручність користування, водночас залишаючись технологічно сумісною з уже розгорнутими сервісами SmartUKMA і дотримуючись галузевих та державних нормативів щодо зберігання і обробки даних.

РОЗДІЛ 2: Проєктування підсистеми

2.1. Загальна архітектура рішення

Підсистема автоматизації роботи відділу кадрів розробляється в межах платформи SmartUkma та інтегрується у її загальну мікросервісну архітектуру. Цей підхід передбачає розбиття складної системи на окремі, автономні сервіси, кожен з яких відповідає за конкретну бізнес-функцію. Завдяки цьому досягається суттєве спрощення процесу розробки, масштабування та впровадження нових функцій без суттєвого впливу на інші компоненти.

Основними сервісами екосистеми SmartUkma, з якими взаємодіє розроблюваний модуль автоматизації кадрового обліку, є:

- ui-server – відповідає за прийом і первинну обробку запитів клієнтської сторони, авторизацію користувачів через Azure AD, а також здійснює проксування цих запитів до відповідних мікросервісів.
- info-server – центральний довідковий сервіс платформи, який зберігає та надає інформацію про основні сутності університету (освітні програми, факультети, кафедри, дисципліни, навчальні плани тощо). Підсистема відділу кадрів взаємодіє з цим сервісом для отримання та оновлення інформації про дисципліни, викладачів і їхнє навчальне навантаження.
- notifications – сервіс, що відповідає за асинхронне інформування користувачів про зміни у стані сутностей. Він отримує повідомлення від інших сервісів через систему RabbitMQ та надсилає сповіщення електронною поштою або через push-інтерфейси.

Кожен із цих сервісів взаємодіє з іншими через REST API, використовуючи формат JSON поверх протоколу HTTPS. Окрім того, для асинхронної комунікації та передачі бізнес-подій між модулями використовується система черг повідомлень RabbitMQ, що дозволяє ефективно масштабувати та забезпечувати стабільність роботи навіть за високого навантаження.

Дані, необхідні для роботи сервісу, зберігаються в реляційній базі даних PostgreSQL. Цей вибір зумовлений чітко визначеною структурою та високими вимогами до цілісності й консистентності інформації. PostgreSQL надає необхідні інструменти для реалізації складних транзакційних сценаріїв та забезпечує високий рівень продуктивності при роботі з великими обсягами даних.

Для реалізації повнотекстового пошуку та швидкого отримання інформації за складними критеріями в рамках підсистеми використовується Elasticsearch. Він дозволяє ефективно індексувати навчальні плани, дисципліни та інформацію про викладачів, забезпечуючи високу швидкість виконання пошукових запитів, що є критично важливим для забезпечення зручності користувачів.

Процес розгортання всіх мікросервісів платформи SmartUkma, зокрема і сервісу автоматизації роботи відділу кадрів, виконується з використанням технологій контейнеризації. В якості основного інструменту використовується Docker, що дозволяє «упакувати» додаток з усіма його залежностями у контейнер. Для керування набором контейнерів у процесі розробки застосовується Docker Compose. Він спрощує запуск багатокомпонентних застосунків, автоматично налаштовуючи комунікацію між ними.

Для контролю версій коду та автоматизації процесів інтеграції й розгортання використовується платформа GitLab. Для кожного мікросервісу налаштовано окремий CI/CD pipeline, що автоматично здійснює збирання додатка, запуск тестів та подальше розгортання нових версій сервісу на тестові та бойові середовища. Це суттєво скорочує час доставки нового функціоналу кінцевим користувачам.

Моніторинг та аналіз роботи сервісів забезпечується за допомогою зв’язки Prometheus та Grafana. Prometheus відповідає за збір метрик із сервісів, а Grafana дозволяє візуалізувати цю інформацію у вигляді зручних та зрозумілих

дашбордів, що значно спрощує виявлення та усунення можливих проблем в роботі підсистеми.

Отже, обрана мікросервісна архітектура та сучасний технологічний стек забезпечують підсистемі автоматизації кадрових процесів необхідну гнучкість, масштабованість та стійкість до збоїв, що є ключовими вимогами для підтримки стабільної роботи університетської інформаційної системи в умовах постійного зростання обсягів даних та складності бізнес-процесів.

2.2 Серверна частина застосунку

Серверна частина підсистеми автоматизації роботи відділу кадрів реалізується на основі Java з використанням фреймворку Spring Boot. Обраний стек є одним із найпопулярніших рішень на ринку для швидкої і ефективної розробки веб-застосунків корпоративного рівня, що забезпечує високу продуктивність програмістів та стабільність роботи сервісу.

Основною перевагою Spring Boot є мінімізація кількості конфігурацій, які необхідні для початку роботи, оскільки багато стандартних налаштувань вже визначено за замовчуванням. Завдяки цьому розробник може одразу зосередитися на створенні бізнес-логіки, не витрачаючи час на попередню ручну конфігурацію залежностей та компонентів застосунку.

Підсистема використовує Hibernate як основний ORM-інструмент для взаємодії з базою даних PostgreSQL. Hibernate значно спрощує роботу з даними, автоматично перетворюючи Java-класи на таблиці бази даних та навпаки. Він забезпечує ефективне виконання складних SQL-запитів, управління транзакціями та кешування, що підвищує загальну швидкодію системи.

Крім роботи з реляційною базою, підсистема також інтегрує Elasticsearch як рішення для реалізації швидкого та повнотекстового пошуку. Elasticsearch надає високопродуктивний механізм пошуку за навчальними дисциплінами,

викладачами та навчальними планами, що є важливим функціоналом для оперативного доступу до інформації.

Взаємодія з іншими сервісами платформи SmartUkma та з клієнтською частиною застосунку здійснюється за допомогою REST API, специфікація якого описується з використанням OpenAPI. Цей підхід забезпечує прозорість та узгодженість контрактів, а також дозволяє автоматично генерувати необхідний код для сервера і клієнта. Застосування OpenAPI дозволяє уникнути багатьох типових помилок, що виникають при ручній реалізації ендпоінтів, та забезпечує швидку інтеграцію і подальший розвиток API.

Під час розробки застосунку було визначено основні контролери та їхні контракти, які описують ресурси та операції взаємодії з сервером. Кожен контролер автоматично генерується за допомогою openapi-generator-maven-plugin, після чого за необхідності може бути розширений шляхом наслідування та перевизначення конкретних методів.

Таким чином, серверна частина підсистеми автоматизації кадрової роботи поєднує в собі сучасні рішення, що дозволяють ефективно управляти великими обсягами інформації, швидко розробляти новий функціонал і легко інтегруватись з іншими компонентами екосистеми SmartUkma. Це створює надійну базу для стабільної роботи, масштабування та подальшого розвитку системи.

Візуальна модель основних класів підсистеми, та їх взаємозв’язки, подана у Додатку 1 у вигляді UML-діаграми класів. Вона відображає структуру доменної моделі, її поля, методи та напрямки залежностей між сутностями.

2.3 Збереження даних

Для збереження даних підсистеми автоматизації роботи відділу кадрів була обрана система управління базами даних PostgreSQL. Вибір саме цієї реляційної бази даних зумовлений чіткою структурою сутностей, високими

вимогами до консистентності інформації та підтримкою складних транзакційних запитів, які важливі для реалізації кадрових процесів.

Основні сутності бази даних структуровані навколо центральних об’єктів, таких як «Дисципліна» та «Викладач», які виступають ключовими бізнес-об'єктами у процесі розподілу навчального навантаження. Сутність «Дисципліна» містить атрибути, що описують навчальний курс: кількість лекційних, семінарських, практичних і лабораторних годин, години самостійної роботи та тип контролю (екзамен або залік). Дисципліни пов’язані з кафедрою, яка за них відповідає, а також мають зв’язок з навчальними програмами та планами, де визначається конкретний навчальний рік, семестр та спеціальність.

Сутність «Викладач» зберігає дані, необхідні для розрахунку навантаження, такі як кількість студентів, кількість груп та інших видів діяльності (рецензування робіт, керівництво практикою, дипломними роботами тощо). Кожен викладач пов'язаний з однією або кількома дисциплінами, а обчислення годин виконуються автоматично на основі попередньо визначених формул.

З метою мінімізації дублювання інформації та забезпечення нормалізації даних реалізовано додаткові довідники, серед яких можна виділити:

- Кафедри та факультети, які зберігають загальну інформацію про структурні підрозділи університету;
- Спеціальності та освітні програми, які допомагають чітко визначити навчальні напрями та відповідні дисципліни;
- Довідники типів контролю (залік, екзамен тощо) для використання у формулах автоматичного розрахунку годин.

Зв’язки між цими сутностями переважно мають тип «один-до-багатьох». Наприклад, кафедра має кілька дисциплін, факультет містить кілька кафедр, а освітня програма складається з багатьох дисциплін. Водночас деякі відношення,

такі як призначення викладача на дисципліну, можуть бути реалізовані за типом «багато-до-багатьох» через окрему проміжну таблицю, яка містить додаткові атрибути (кількість годин, тип залучення тощо).

Для швидкого повнотекстового пошуку та отримання необхідних даних у межах підсистеми використовується Elasticsearch. Він виступає як додаткове рішення, яке ефективно індексує такі дані, як опис дисциплін, інформацію про викладачів та навчальні плани. Основною сутністю для Elasticsearch є документ, що містить текстові поля для пошуку, а також унікальні ідентифікатори сутностей у PostgreSQL, що дозволяє швидко зв'язувати результати пошуку з відповідними записами в реляційній базі.

Описана структура збереження даних дозволяє забезпечити високий рівень цілісності та консистентності інформації, знижує ризики помилок та забезпечує легкість обслуговування й масштабування бази даних.

2.4 Опис клієнтської частини

Клієнтська частина підсистеми автоматизації роботи відділу кадрів реалізується з використанням фреймворку Angular. Дане рішення було обране завдяки можливості створювати динамічні, інтерактивні та добре структуровані користувацькі інтерфейси. Angular базується на компонентному підході, що дозволяє ефективно організовувати код, повторно використовувати компоненти та значно прискорювати процес розробки і підтримки застосунку.

Однією з ключових переваг Angular є використання мови програмування TypeScript, що забезпечує строгий контроль типів, кращу навігацію по коду, покращену автодоповнюваність та допомагає уникати типових помилок, характерних для звичайного JavaScript. В умовах розробки складних корпоративних систем, таких як університетська платформа SmartUkma, ці переваги є критично важливими, адже вони знижують кількість помилок на етапах розробки та підтримки продукту.

Ще одним важливим елементом у клієнтській частині є бібліотека RxJS, яка дозволяє ефективно працювати з асинхронними потоками даних. Завдяки цьому забезпечується швидка і реактивна взаємодія користувача з сервером та зменшується затримка під час оновлення інтерфейсу при зміні даних.

Для спрощення і пришвидшення процесу розробки, використовується інструмент Angular CLI, який автоматизує багато рутинних задач: створення нових компонентів, сервісів, маршрутизації, збирання та розгортання застосунку. Це дозволяє розробникам зосередитись саме на створенні бізнес-логіки та функціоналу, не витрачаючи зайвий час на конфігурацію проекту та ручну структуру коду.

Зв’язок між клієнтською частиною (Angular) і серверною частиною (Spring Boot) здійснюється через REST API. При цьому як клієнт, так і сервер використовують єдину специфікацію API (OpenAPI), що дозволяє автоматично генерувати код клієнтської взаємодії з бекендом, забезпечуючи цілісність та узгодженість взаємодії. Для генерації TypeScript-коду, що реалізує API-контракти на стороні клієнта, використовується npm-пакет openapi-generator-cli. Це дозволяє суттєво зменшити кількість ручного програмування і мінімізувати можливість виникнення помилок у процесі інтеграції.

Клієнтська частина взаємодіє з сервером через проксі-сервіс ui-server, реалізований на Spring Boot, який бере на себе відповідальність за автентифікацію користувачів, перевірку прав доступу та маршрутизацію запитів до відповідних мікросервісів. Такий підхід забезпечує додатковий рівень безпеки та гнучкості під час розробки та подальшої підтримки.

Для кращого розуміння структури клієнтського інтерфейсу у Додатку 2наведено скриншоти основних екранів підсистеми, включно з формами редагування дисциплін, розподілом навантаження викладачів та підрахунком годин.

Загалом, використання Angular у поєднанні з серверною частиною на Spring Boot створює сучасну, швидку, масштабовану та стабільну основу для реалізації ефективного і зручного інтерфейсу користувача, що відповідає високим вимогам корпоративного застосунку SmartUkma і дозволяє комфортно та продуктивно взаємодіяти з підсистемою автоматизації кадрових процесів.

2.5 Підсумок розділу

У третьому розділі було розглянуто особливості архітектурної побудови підсистеми автоматизації роботи відділу кадрів у рамках платформи SmartUkma. Вибрана мікросервісна архітектура забезпечує високий рівень гнучкості, можливість незалежної розробки компонентів, простоту масштабування та інтеграції з іншими модулями системи. Особливий акцент зроблено на взаємодії з основними мікросервісами платформи (ui-server, info-server, notifications), що забезпечує надійну та стабільну роботу додатку.

Реалізація серверної частини на базі технологій Java та Spring Boot гарантує ефективну розробку, знижує кількість помилок та мінімізує час, потрібний для впровадження нових функцій. Використання Hibernate як ORM-інструменту значно спрощує роботу з базою даних PostgreSQL, яка була обрана через її переваги в підтримці складних запитів, забезпеченні цілісності даних та високій швидкодії.

Для виконання завдань, пов'язаних з пошуком інформації, інтегровано Elasticsearch. Це дозволяє ефективно індексувати і швидко знаходити необхідну інформацію серед великих обсягів даних.

На клієнтській стороні застосовується сучасний фреймворк Angular, який забезпечує якісну структуру коду, швидкий і зручний інтерфейс користувача, а також ефективну інтеграцію з серверною частиною через REST API та OpenAPI-контракти.

Описаний набір сучасних технологій та інструментів дозволяє створити гнучку, стабільну та ефективну підсистему, яка здатна масштабуватись і підтримувати складні бізнес-процеси відділу кадрів у довгостроковій перспективі.

РОЗДІЛ 3: Реалізація підсистеми

3.1 Розробка основних модулів

Розробка підсистеми автоматизації роботи відділу кадрів була здійснена відповідно до визначеної архітектури та специфікації API. У процесі реалізації було створено кілька ключових модулів, кожен з яких відповідає за окрему частину бізнес-логіки та забезпечує функціональність, необхідну для роботи кадрового відділу та кафедр університету.

Одним із центральних компонентів є модуль дисциплін. У межах цього модуля реалізовано зберігання та управління інформацією про дисципліни, зокрема кількість лекційних, семінарських, практичних і лабораторних годин, години самостійної роботи та типи підсумкового контролю. Також було створено автоматичні розрахунки, які обчислюють загальну кількість аудиторних та повних годин дисципліни відповідно до формул, описаних у постановці задачі. Цей модуль тісно інтегрований з серверною частиною на базі Spring Boot, яка забезпечує REST API для взаємодії з клієнтською стороною.

Наступним важливим компонентом є модуль розподілу навантаження, що відповідає за автоматичний обрахунок та розподіл годин між викладачами. Для кожного викладача автоматично розраховуються години консультацій, екзаменів, заліків, письмових робіт, практик та інших типів навантаження залежно від типу контролю дисципліни та кількості студентів і груп. Важливим аспектом цього модуля є те, що відповідні поля розрахунків встановлюються у режимі «readonly» на клієнтському боці, що гарантує точність і достовірність інформації, а також унеможливлює помилки при ручному введенні.

Окремий модуль був створений для роботи з інформацією про викладачів, який забезпечує ведення персональних карток із даними про групи та студентів, що належать до відповідного викладача. Цей модуль відповідає за агрегування даних про кількість груп і студентів у межах дисциплін, які читає конкретний

викладач, а також забезпечує валідацію та синхронізацію цих даних із загальними показниками дисципліни, що відображаються методисту кафедри.

Для зручності користувачів був також створений модуль повідомлень та нотифікацій, що інтегрується із сервісом повідомлень платформи SmartUkma. Цей модуль автоматично інформує методистів, адміністраторів та викладачів про зміни в навантаженні, а також про необхідність узгодження чи перегляду даних. Це забезпечує прозорість процесу та оперативність у реагуванні на зміни.

Крім основних модулів, реалізовано допоміжні сервіси, такі як авторизація та автентифікація користувачів**,** що використовує Azure AD, а також модуль для взаємодії з Elasticsearch, який забезпечує швидкий і точний повнотекстовий пошук за дисциплінами, навчальними планами та інформацією про викладачів.

Кожен з перерахованих модулів побудовано відповідно до принципів мікросервісної архітектури, що дозволяє легко підтримувати, тестувати, оновлювати та масштабувати підсистему. Використання сучасних практик розробки, таких як Continuous Integration та Continuous Delivery (CI/CD), контейнеризація за допомогою Docker і автоматизація тестування, дозволили суттєво скоротити час розробки, підвищити якість коду та стабільність роботи готової системи.

3.2 Приклади реалізації

У процесі розробки підсистеми було реалізовано низку ключових функціональних елементів, що відповідають за автоматичний обрахунок навчального навантаження та формування представлень у відповіді API. Архітектура побудована таким чином, щоб обчислення базувались на бізнес-логіці, а не на збережених у БД значеннях, що забезпечує гнучкість і зменшує ризик втрати узгодженості даних.

Одним із прикладів такої реалізації є розрахунок загальної кількості аудиторних та повних годин дисципліни**.** Для цього було використано MapStruct-

конвертер, у якому визначено дві допоміжні функції mapTotalClassHours та mapTotalDisciplineHours. Обидві працюють із потоком чисел, відфільтровуючи null-значення та обчислюючи підсумок.(див. додаток 3)

Обчислені значення вбудовуються у DTO DisciplineResponse, оскільки зберігати їх у базі даних без потреби.(див. додаток 4)

Іншим прикладом є автоматичний розрахунок часткових видів навантаження викладача**:** консультацій, екзаменів, письмових робіт і заліків. Розрахунки базуються на кількості студентів, груп та типі контролю дисципліни (залік або екзамен). Кожен тип обраховується окремою функцією.(див. додаток 5)

Ці функції об’єднуються в одному мапері, який формує розгорнуту відповідь DisciplineTeacherResponse для відображення повного навантаження викладача.(див. додаток 6)

Усі ці приклади ілюструють загальну архітектурну ідею — логіка обчислень виноситься у мапери або сервіси**,** а DTO-об’єкти формуються динамічно відповідно до поточних даних. Це дозволяє не лише уникати дублювання інформації, а й забезпечити високу гнучкість та передбачуваність поведінки системи під час зміни вхідних параметрів.

Крім бекенд-частини, тісно пов’язаної з маперами, логіка автоматичних розрахунків також реалізується на клієнтському рівні для правильного відображення результатів. Усі поля, що розраховуються на сервері (наприклад, екзаменаційні години, консультації, письмові роботи), виводяться у формі лише для перегляду (readonly) та не підлягають редагуванню з боку користувача. Це запобігає помилкам і гарантує цілісність даних.

Наприклад, у файлі discipline-view.component.html реалізовано відображення підсумкових обрахованих годин.(див. додаток 7)

У файлі upsert.component.html реалізовано підсвічування поля «Розподіл по викладачам», яке залежить від коректності заповнення кількості груп(див. додаток 8)

У компоненті upsert.component.ts присутня відповідна логіка перевірки відповідності, яка використовується для візуального індикатора.(див додаток 9)

Ці приклади демонструють наскрізну реалізацію логіки — обрахунки виконуються на бекенді, а на фронтенді забезпечується їх правильне представлення та перевірка узгодженості. Такий підхід дозволяє мінімізувати помилки ручного введення, забезпечити прозорість розрахунків і спростити взаємодію з інтерфейсом для методистів та викладачів.

3.3 Інтеграція з системою SmartUkma

Розроблена підсистема автоматизації роботи відділу кадрів є частиною мікросервісу user-info в загальній екосистемі SmartUkma. Сервіс user-info є центральним компонентом платформи, який агрегує та управляє інформацією про користувачів (викладачів, студентів, адміністраторів) і навчальними даними, що використовуються іншими сервісами системи.

Інтеграція кадрового модуля в рамках user-info відбувалася з урахуванням існуючих стандартів взаємодії між компонентами SmartUkma. Вся комунікація між клієнтською частиною (Angular) і серверною частиною (Spring Boot) здійснюється через REST API, специфікація якого описана у форматі OpenAPI. Цей підхід дозволяє автоматично генерувати як серверні контролери, так і клієнтський код, забезпечуючи чіткий контракт взаємодії, зменшуючи ймовірність виникнення помилок і спрощуючи процес інтеграції.

Оскільки кадровий модуль є частиною сервісу user-info, для автентифікації й авторизації користувачів використовується спільна для всієї платформи

інфраструктура Azure Active Directory (Azure AD). Це дозволяє використовувати єдину рольову модель та уніфікований підхід до безпеки. Під час автентифікації сервіс отримує інформацію про ролі користувачів з Azure AD і визначає їхні права доступу до функціоналу підсистеми, що забезпечує надійний захист даних та гнучке управління доступом.

Вся інформація, пов’язана з дисциплінами, викладачами та розподілом навантаження, зберігається у спільній базі даних PostgreSQL, яка використовується сервісом user-info. Це забезпечує централізовану точку доступу до даних, а також гарантує цілісність і консистентність інформації на рівні всієї платформи. ORM Hibernate дозволяє ефективно взаємодіяти з базою даних, мінімізуючи кількість рутинного коду.

Окрім цього, для забезпечення швидкого повнотекстового пошуку використовується Elasticsearch, інтегрований у спільний механізм пошуку системи SmartUkma. Завдяки цьому підсистема кадрового обліку забезпечує швидкий та ефективний пошук інформації про дисципліни, навантаження викладачів та інші важливі дані.

Розгортання кадрового модуля здійснюється за допомогою контейнеризації Docker та оркестрації через Docker Compose. Автоматизоване тестування та розгортання забезпечується через GitLab CI/CD, що дозволяє швидко та безпечно оновлювати підсистему, інтегровану в сервіс user-info.

Таким чином, підсистема автоматизації роботи відділу кадрів органічно вбудовується у сервіс user-info, забезпечуючи стабільну інтеграцію, уніфікацію з іншими компонентами екосистеми та ефективну роботу всього комплексу SmartUkma.

РОЗДІЛ 4: Тестування та оцінка результатів

4.1 Методика тестування

Для перевірки функціональності підсистеми автоматизації роботи відділу кадрів використовувалась комплексна методика, що охоплює кілька рівнів тестування, забезпечуючи високу якість, стабільність та відповідність розробленого програмного продукту заявленим вимогам.

На початковому етапі проводилося модульне (Unit) тестування, метою якого була перевірка коректності роботи окремих компонентів системи, таких як методи автоматичних розрахунків, контролери REST API та сервіси для роботи з даними. Тестові сценарії писалися за допомогою бібліотеки JUnit та фреймворку Mockito, що дозволяло ефективно тестувати логіку системи, перевіряючи її поведінку на різних вхідних даних, включаючи граничні та некоректні значення.

Наступним рівнем було проведено інтеграційне тестування, яке мало на меті перевірку коректності взаємодії між модулями підсистеми, а також з іншими мікросервісами платформи SmartUkma. Для реалізації таких тестів використовувались інтеграційні інструменти Spring Boot Test та бібліотека Testcontainers, яка дозволяє запускати реальну базу даних PostgreSQL та Elasticsearch в ізольованому Docker-контейнері. Це забезпечує високу точність тестів та їхню близькість до реальних умов експлуатації системи.

Особлива увага була приділена також мануальному тестуванню, яке дозволило оцінити працездатність системи з погляду кінцевих користувачів. Цей тип тестування здійснювався заздалегідь складеними тестовими сценаріями, які містили чітко визначені кроки та очікувані результати. В процесі мануального тестування перевірялися такі аспекти:

- Коректність автоматичних обрахунків годин дисциплін та навантаження викладачів залежно від введених даних.

- Перевірка обмежень доступу до функціоналу залежно від ролей користувачів, отриманих через Azure AD.
- Зручність та інтуїтивність інтерфейсу клієнтської частини, перевірка readonly-полів, автоматичного заповнення полів та валідації введених значень.
- Перевірка процесу взаємодії між користувачами (методистами кафедр, адміністраторами, викладачами), а також коректність обчислень сумарних даних на рівні дисциплін.
- Робота системи повідомлень та нотифікацій, перевірка своєчасності та правильності отриманих сповіщень.

Під час мануального тестування реєструвалися всі виявлені проблеми, помилки та недоліки інтерфейсу чи логіки роботи застосунку. Отримані результати були оформлені у звіти з конкретними рекомендаціями щодо покращення функціоналу та зручності використання.

Після проходження всіх етапів тестування було створено узагальнені звіти, які містили перелік проведених тестових сценаріїв, опис знайдених помилок, рекомендації щодо їхнього виправлення та висновки щодо якості й готовності системи до впровадження у реальну експлуатацію. Такий комплексний підхід до тестування дозволив досягти високого рівня надійності та впевненості в якості розробленої підсистеми автоматизації роботи відділу кадрів.

4.2 Ефективність автоматизованої системи

Важливим етапом розробки підсистеми автоматизації роботи відділу кадрів було проведення аналізу її ефективності та оцінка практичного значення впровадженого рішення. У процесі експериментальної експлуатації системи на реальних даних кафедри було встановлено, що використання автоматизованих обрахунків значно скорочує час, необхідний для розрахунку та перевірки навчального навантаження викладачів.

Так, порівняно з ручним процесом, використання підсистеми скоротило час формування та погодження навчального навантаження. Якщо раніше методист кафедри витрачав до кількох годин на розрахунки годин по кожній дисципліні, перевірку відповідності даних, то завдяки автоматичним розрахункам цей процес тепер триває не більше 10-15 хвилин. Це дозволяє методистам приділяти більше уваги контролю якості навчальних планів і дисциплін, а також швидко реагувати на зміни у навчальному процесі.

Суттєвим фактором ефективності підсистеми є точність автоматичних обрахунків, що мінімізує ймовірність людських помилок. Під час тестування було виявлено, що використання автоматизованих формул практично усуває помилки, які могли виникати через ручне введення та перерахунок даних. Завдяки тому, що важливі поля встановлені у режимі «readonly», вдалося досягти високого рівня достовірності інформації.

Ще однією перевагою автоматизації стала зручність доступу до інформації про навантаження з боку викладачів і керівництва кафедри. Впроваджена підсистема дозволяє оперативно відслідковувати навантаження, зміни в кількості студентів і груп, а також здійснювати швидкий пошук інформації через інтегрований Elasticsearch. Це підвищує якість управлінських рішень, забезпечує прозорість процесів та суттєво спрощує звітність.

Крім того, інтеграція підсистеми в загальний мікросервіс user-info значно полегшує адміністрування, оновлення і масштабування сервісу, що є важливим фактором для довгострокового функціонування всієї інформаційної системи SmartUkma.

Таким чином, проведений аналіз ефективності показав, що розроблена підсистема суттєво покращує якість і швидкість роботи кадрових процесів в університеті, зменшує навантаження на працівників, знижує ризик виникнення помилок, забезпечуючи високий рівень точності й оперативності кадрового обліку.

ВИСНОВОК

У ході виконання курсової роботи було створено підсистему автоматизації роботи відділу кадрів як інтегральну частину мікросервісу user-info платформи SmartUkma. Основна мета проєкту—зменшення витрат часу на розрахунок навчального навантаження викладачів, забезпечення точності обчислень, мінімізація помилок ручного введення, а також підвищення прозорості й ефективності кадрових процесів—була повністю досягнута.

Запропоноване рішення базується на сучасних технологіях: серверна частина розроблена на базі Java із застосуванням Spring Boot, використовується Hibernate як ORM для роботи з базою даних PostgreSQL, а Elasticsearch забезпечує високошвидкісний повнотекстовий пошук. Клієнтська частина побудована з використанням Angular та інтегрується з бекендом через REST API, специфікація якого описана OpenAPI-контрактами.

Використання автоматичних обрахунків дозволило скоротити час на розподіл навантаження викладачів, а також суттєво підвищило якість і точність отриманих результатів завдяки мінімізації людського фактору. Чітка рольова модель доступу, реалізована через Azure AD, забезпечує високий рівень безпеки й ефективно регламентує права користувачів.

Тестування, проведене на різних рівнях (модульному, інтеграційному та мануальному), підтвердило стабільну роботу системи, відповідність функціональних можливостей вимогам, а також комфортність інтерфейсу для користувачів.

Отже, розроблена підсистема автоматизації кадрових процесів ефективно виконує поставлені задачі та демонструє високий рівень технологічної готовності до впровадження й експлуатації в реальному середовищі університету.

Список використаних джерел

1. Mykhailenko O., Hlybovets A. Типи працівників НаУКМА. *Clickup*. 23.05.2024. URL: https://app.clickup.com/20593116/v/dc/kmeew-975/kmeew-2215.
2. Mykhailenko O., Hlybovets A. Бізнес процеси. *Clickup*. 22.01.2024. URL: https://app.clickup.com/20593116/v/dc/kmeew-1015/kmeew-2315.
3. Mykhailenko O., Hlybovets A. Ролі/права. *Clickup*. 23.05.2024. URL: https://app.clickup.com/20593116/v/dc/kmeew-1035/kmeew-2355.
4. Elasticsearch documentation. *Elastic*. URL: https://www.elastic.co/docs/.
5. PostgreSQL documentation. *PostgreSQL*. URL: https://www.postgresql.org/docs/.
6. Angular documentation. *Angular.dev*. URL: https://angular.dev/overview.
7. Spring Boot documentation. *Spring*. URL: https://docs.spring.io/spring-boot/documentation.html.

Додатки

Додаток 1

Додаток 2

Інформація про дисципліну

Назва: Комп`ютерна графіка

Назва (англ): Комп`ютерна графіка

Короткий опис:

Короткий опис (англ):

Семестр викладання: 8

Всього аудиторних занять: 42

Всього годин дисципліни 122

Кількість лекційних годин: 24

Кількість семінарських годин: 0

Кількість годин практичних і лабораторних: 18

Кількість годин самостійної роботи: 80

Кількість кредитів ЄКТС: 4

Факультет: Факультет інформатики

Тип контролю: Екзамен

Навчальні процеси: Суспільна політика і врядування

Мова: англійська

Тип дисципліни: Екзамен

Опис:
Опис (англ):
Викладачі

Викладачі

П'ятий Тестовий

Пошта: dms5@ukma.edu.ua

Роль: Лектор

Кількість студентів: 240

Кількість груп: 15

Загальна кількість годин викладача:

Лекції: 10

Семінари: 12

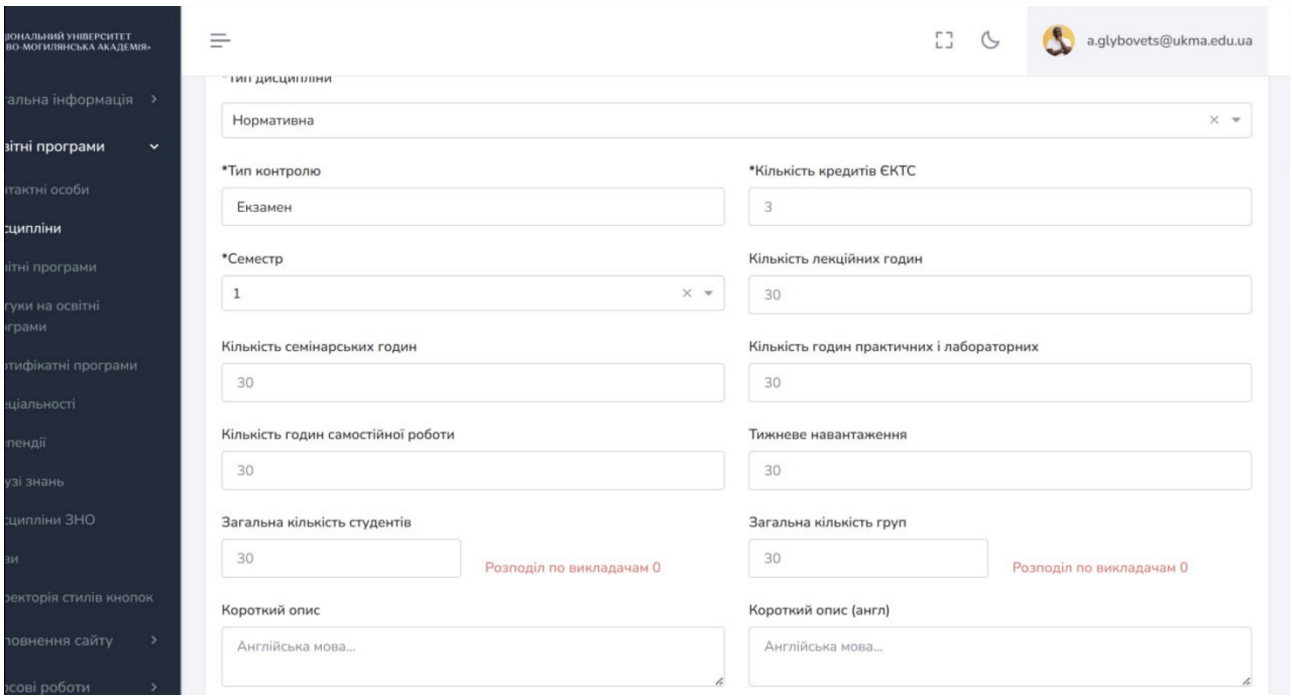
Практичні та лабораторні: 22

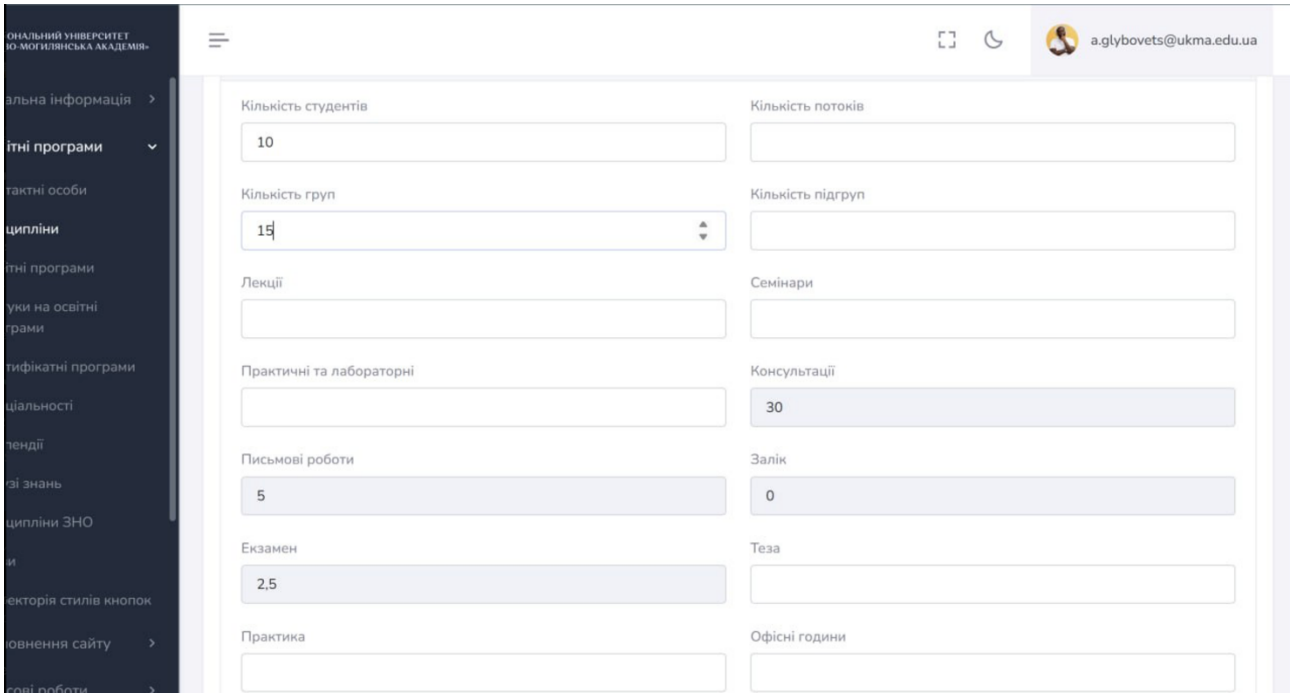
Консультації: 30

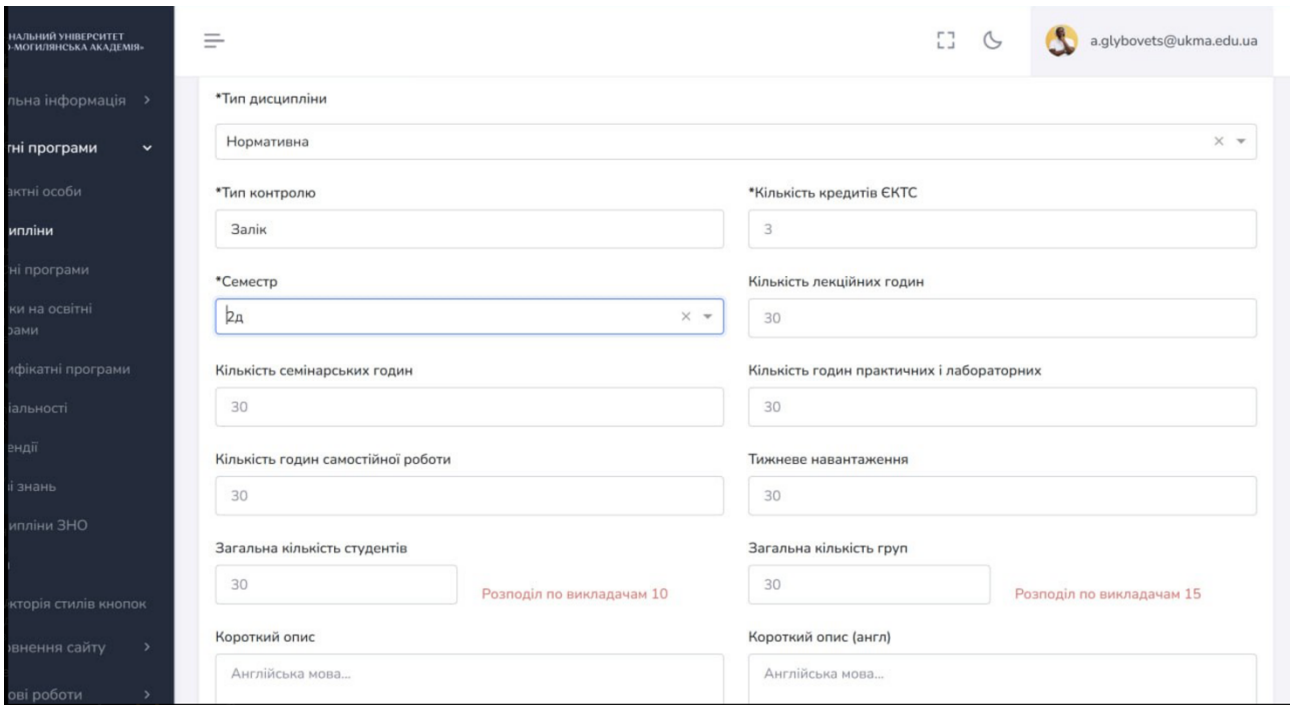
Письмові роботи: 120

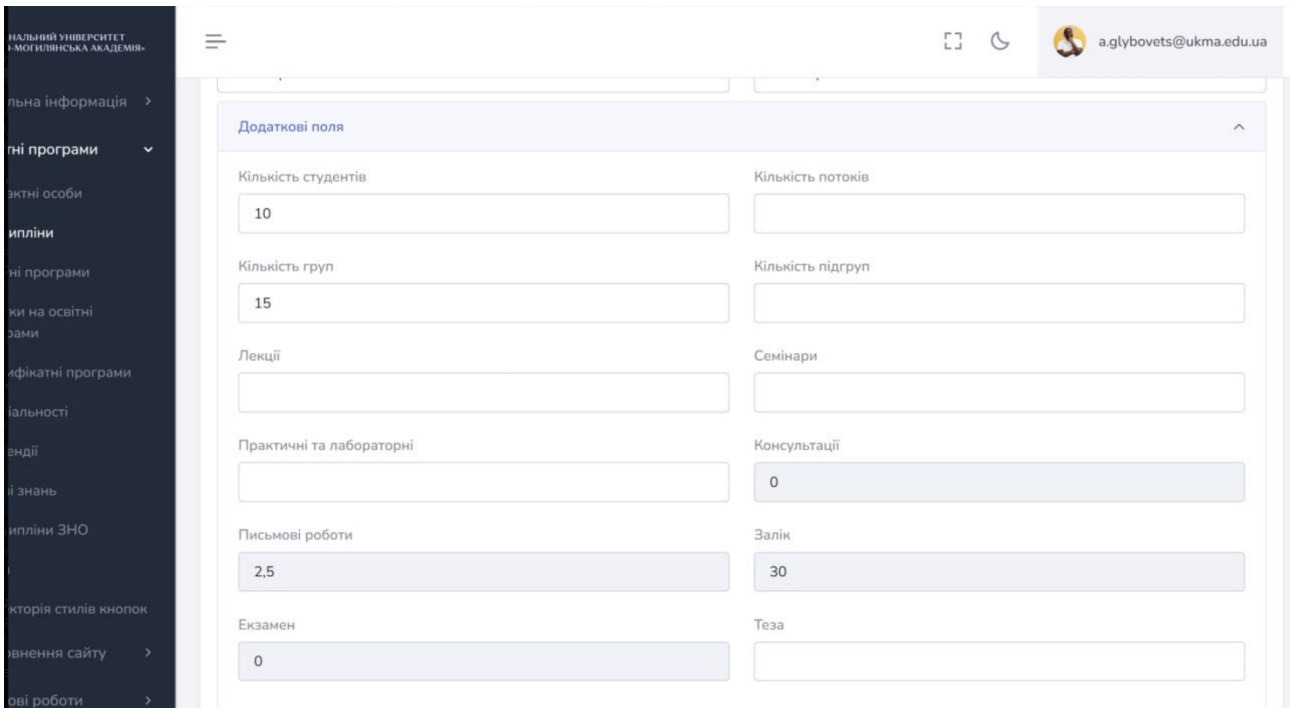
Залік: 0

Екзамен: 60


a.glybovets@ukma.edu.ua
Нормативна
*Тип контролю
Екзамен
*Кількість кредитів ЄКТС
3
*Семестр
1
Кількість лекційних годин
30
Кількість семінарських годин
30
Кількість годин практичних і лабораторних
30
Кількість годин самостійної роботи
30
Тижневе навантаження
30
Загальна кількість студентів
30
Розподіл по викладачам 0
Загальна кількість груп
30
Розподіл по викладачам 0
Короткий опис
Англійська мова...
Короткий опис (англ)
Англійська мова...


a.glybovets@ukma.edu.ua
Кількість студентів
10
Кількість потоків
Кількість груп
15
Кількість підгруп
Лекції
Семінари
Практичні та лабораторні
Консультації
30
Письмові роботи
5
Залік
0
Екзамен
2,5
Теза
Практика
Офісні години


НАЛЬНИЙ УНІВЕРСИТЕТ
-МОГИЛЯНСЬКА АКАДЕМІЯ»
льна інформація
гні програми
жтні особи
ипліни
ні програми
ки на освітні
рами
фікатні програми
іальності
ендії
і знань
ипліни ЗНО
кторія стилів кнопок
внення сайту
ові роботи
a.glybovets@ukma.edu.ua
*Тип дисципліни
Нормативна
*Тип контролю
Залік
*Кількість кредитів ЄКТС
3
*Семестр
2д
Кількість лекційних годин
30
Кількість семінарських годин
30
Кількість годин практичних і лабораторних
30
Кількість годин самостійної роботи
30
Тижневе навантаження
30
Загальна кількість студентів
30
Розподіл по викладачам 10
Загальна кількість груп
30
Розподіл по викладачам 15
Короткий опис
Англійська мова...
Короткий опис (англ)
Англійська мова...


НАЛЬНИЙ УНІВЕРСИТЕТ
-МОГИЛЯНСЬКА АКАДЕМІЯ»
льна інформація
гні програми
жтні особи
ипліни
ні програми
ки на освітні
рами
фікатні програми
іальності
ендії
і знань
ипліни ЗНО
кторія стилів кнопок
внення сайту
ові роботи
a.glybovets@ukma.edu.ua
Додаткові поля
Кількість студентів
10
Кількість потоків
Кількість груп
15
Кількість підгруп
Лекції
Семінари
Практичні та лабораторні
Консультації
0
Письмові роботи
2,5
Залік
30
Екзамен
0
Теза

Додаток 3

```
@Named(TOTAL_CLASS_HOURS_MAPPER)  1 usage  Yaroslav Frankiv
default Integer mapTotalClassHours(DisciplineEntity d) {
    return Stream.of(
                    d.getLecturesHours(),
                    d.getSeminarsHours(),
                    d.getPracticalAndLaboratoryHours()
            )
            .filter(Objects::nonNull)
            .reduce( identity: 0, Integer::sum);
}

@Named(TOTAL_DISCIPLINE_HOURS_MAPPER)  1 usage  Yaroslav Frankiv
default Integer mapTotalDisciplineHours(DisciplineEntity d) {
    return Stream.of(
                    d.getLecturesHours(),
                    d.getSeminarsHours(),
                    d.getPracticalAndLaboratoryHours(),
                    d.getIndividualWorkHours()
            )
            .filter(Objects::nonNull)
            .reduce( identity: 0, Integer::sum);
}
```

Додаток 4

```
@Mapping(target = "totalClassHours",  1 usage  1 implementation  Yaroslav Frankiv
        source = "entity",
        qualifiedByName = TOTAL_CLASS_HOURS_MAPPER)
@Mapping(target = "totalDisciplineHours",
        source = "entity",
        qualifiedByName = TOTAL_DISCIPLINE_HOURS_MAPPER)
DisciplineResponse mapDiscipline(DisciplineEntity entity);
```

Додаток 5

```
@Named(EXAMS_MAPPER)  2 usages  Yaroslav Frankiv
default Float computeExams(DisciplineTeacherEntity e) {
    if (e.getDiscipline().getControlType() != ControlType.EXAM) {
        return 0f;
    }
    int students = e.getNumberOfStudents() != null ? e.getNumberOfStudents() : 0;
    return students * EXAM_RATIO;
}

@Named(CONSULTATIONS_MAPPER)  2 usages  Yaroslav Frankiv
default Float computeConsultations(DisciplineTeacherEntity e) {
    if (e.getDiscipline().getControlType() != ControlType.EXAM) {
        return 0f;
    }
    int groups = e.getNumberOfGroups() != null ? e.getNumberOfGroups() : 0;
    return groups * GROUP_CONSULTATION_MULTIPLIER;
}
```

Додаток 6

```
@Mapping(target = "writtenWorksHours", source = "entity", qualifiedByName = WRITTEN_WORKS_MAPPER)
@Mapping(target = "examsHours", source = "entity", qualifiedByName = EXAMS_MAPPER)
@Mapping(target = "consultationsHours", source = "entity", qualifiedByName = CONSULTATIONS_MAPPER)
@Mapping(target = "zalikHours", source = "entity", qualifiedByName = ZALIK_MAPPER)


DisciplineTeacherResponse map(DisciplineTeacherEntity entity);
```

Додаток 7

```
<p><strong>Всього аудиторних занять: </strong>{{ discipline.totalClassHours }}</p>
<p><strong>Всього годин дисципліни </strong>{{ discipline.totalDisciplineHours }}</p>
<p><strong>Кількість лекційних годин: </strong>{{ discipline.lecturesHours }}</p>
<p><strong>Кількість семінарських годин: </strong>{{ discipline.seminarsHours }}</p>
```

Додаток 8

```
readonly $checkGroupsDistributionByGroups : Signal<boolean> = computed(() : boolean => {
    const maxGroups : number = this.$maxNumberOfGroups()
    const totalGroupsAmountByTeacher : number = this.$countTotalGroupsAmountByTeacher()
    return maxGroups == totalGroupsAmountByTeacher
})
```

Додаток 9

```
<label class="col-lg-6 d-flex align-items-end m-0"
       [class.text-danger]="!$checkGroupsDistributionByGroups()"
       [class.text-success]="$checkGroupsDistributionByGroups()">
  Розподіл по викладачам {{ $countTotalGroupsAmountByTeacher() }}
</label>
```