«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу

Кафедра системного проектування

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Вадим МУХІН
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою

“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”

зі спеціальності 122 "Комп'ютерні науки"

на тему: «Автоматизація документообігу на основі мікросервісної архітектури»

Виконав :
студент IV курсу, групи ДА-81
Науменко Євгеній Олегович _____

Науковий керівник:
Доцент, к.т.н
Безносик Олександр Юрійович _____

Консультант з економіки:
Доцент, к.е.н.
Рощина Надія Василівна _____

Рецензент:
зав. каф. АПУПС ТЕФ, д.т.н. проф.
Аушева Наталія Миколаївна _____

Засвідчую, що у цій бакалаврській роботі немає запозичень з праць інших авторів без відповідних посилань.
Науменко Євгеній Олегович _____,

Київ – 2022

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра системного проектування

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інтелектуальні сервіс-орієнтовані розподілені обчислювання»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В. Є. Мухін

«___»_____2022 р.

ЗАВДАННЯ

на дипломну роботу студенту

Науменко Євгеній Олегович

1. Тема роботи «Автоматизація документообігу на основі мікросервісної архітектури», керівник роботи Безносик Олександр Юрійович, доцент, к.т.н., затверджені наказом по університету від «06» червня 2022 р. № 906-с
2. Термін подання студентом роботи 17.06.2022
3. Вихідні дані до роботи

Мова програмування – JVM мова (Kotlin/Java), середовище розробки – Intellij IDEA, фреймворк для розробки – Spring Boot v2.6.x.

4. Зміст роботи:
    1. Аналіз переваг мікросервісної архітектури;
    2. Порівняння існуючих рішень автоматизації документообігу;
    3. Аналіз технологій, які будуть використовуватися для виконання дипломного проекту;
    4. Реалізація сервісів та комунікації між ними;
    5. Результати та тестування роботи програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний	Рощина Н.В.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	11.02.2022	
2	Збір інформації	11.03.2022	
3	Ознайомлення з літературою і підготовка теоретичної частини роботи	28.04.2022	
4	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	15.04.2022	
6	Розробка програмної реалізації	16.05.2022	
7	Тестування розробленої системи	28.05.2022	
8	Розробка економічної частини дипломного проекту	05.06.2022	
9	Оформлення дипломної роботи	15.06.2022	
10	Отримання допуску до захисту та подача роботи в ДЕК	17.06.2022	

Студент Євгеній НАУМЕНКО

Керівник Олександр БЕЗНОСИК

АНОТАЦІЯ

Дипломна робота: 141 с., 39 рис., 7 табл., 7 дод., 29 джерел.

АВТОМАТИЗАЦІЯ ДОКУМЕНТООБІГУ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Предмет дослідження — документообіг адміністративних одиниць.

Об’єкт дослідження — автоматизоване підписання документів

Мета роботи — автоматизація документообігу шляхом розробки веб-додатку на основі мікросервісної архітектури

Методи дослідження — використання технологій для полегшеної розробки мікросервісів та комунікації між ними.

Проведено огляд монолітної та мікросервісної архітектури.

Досліджено готові програми для вирішення поставленої проблеми.

Розроблено веб-додаток для дипломної роботи.

Оцінено програмний продукт

ABSTRACT

Thesis: 141 pages, 39 figures, 7 tables, 7 appendices, 29 sources.

DOCUMENT WORKFLOW AUTOMATION BASED ON MICROSERVICE ARCHITECTURE

Document workflow automation based on microservice architecture

The subject of research is the document flow of administrative units.

The object of research is the automated signing of documents

The purpose of the work - automation of document management by developing a web application based on microservice architecture

Research methods - the use of technologies to facilitate the development of micro-services and communication between them.

A review of monolithic and microservice architecture was conducted.

Ready-made programs for solving the problem are studied.

A web application for the thesis has been developed.

The software product is evaluated

ЗМІСТ

ВСТУП

Автоматизація документообігу є важливою частиною кожного великого підприємства або закладів, де безперервно виконується паперова робота. Адже завдяки цьому можна пришвидшити роботу всього персоналу або окремих відділів, від чого збільшиться продуктивність як всієї організації, так і декількох компаній, що комунікують між собою.

Також такою системою є можливість налагодити бізнес-процеси так, щоб не порушувати законодавство, надати гарантії того, що документ не буде скомпрометовано та він буде збережений надійно у базі даних задля зведення загублення до мінімуму.

Слід зазначити економічну вигоду наявності програмної автоматизації документообігу. По-перше, скоротяться витрати на папір. За опитуванням міської адміністрації міста Середина-Буда Сумської області закупівля становить приблизно 50 упаковок паперу А4 з кількістю у 500 листів на місяць за середньою ціною, яка становить 220 гривень. Тобто щомісячна витрата становить 11 000 гривень та 132 000 гривень на рік. Враховуючи те, що розмір адміністрації цього міста не є великим, то витрати можуть становити в два або в три рази більше. По-друге, якщо документообіг між компаніями, які розташовані у різних містах, то пересилати документи краще в електронному вигляді, так як не буде затрат на кур’єра та пересилку та доставка буде миттєвою. І, звісно, не буде потреби витрачати гроші на оренду архівів, бо кількість матеріалу з кожним роком буде збільшуватись, а розмір архіву просто так не збільшити.

Метою даної бакалаврської роботи є автоматизація документообігу шляхом створення веб-додатку на основі мікросервісної архітектури. Слід забезпечити захищеність даних під час переправлення їх між сервісами.

Цілі бакалаврського дослідження:

1. Проаналізувати переваги мікросервісної архітектури;
2. Провести порівняння вже існуючих програмних засобів автоматизації документообігу;
3. Розробка та тестування програми.

1 ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ

Ми маємо справу в розробці повноцінного веб-додатку для спрощення роботи з документами в адміністративній одиниці, який має сервер у вигляді N-ої кількості сервісів, які працюють окремо та слабо пов’язані між собою. Таку архітектуру з великою кількістю сервісів називають мікросервісним.

1.1 Опис мікросервісної архітектури

Мікросервісна архітектура - метод організації архітектури, заснований на ряді незалежних служб. Ці служби мають власну бізнес-логіку та базу даних з конкретною метою. Оновлення, тестування, розгортання та масштабування виконуються всередині кожної служби. Мікросервіси розбивають великі завдання, притаманні конкретному бізнесу, на кілька незалежних баз коду. Вони не знижують складність, але роблять будь-яку складність видимою і більш керованою, поділяючи завдання на дрібніші процеси, які функціонують незалежно один від одного і роблять внесок у загальне ціле.

1.2 Переваги мікросервісів

Гнучкість. Є можливість впроваджувати гнучкі методи роботи серед невеликих команд, які регулярно розгортаються.

Гнучке масштабування. Коли мікросервіс досягає граничного навантаження, можна швидко виконати розгортання нових екземплярів цієї служби у супутньому кластері та знизити навантаження. Тепер можна працювати з декількома власниками та без збереження стану, а клієнти розподілені за різними примірниками. З таким підходом можемо підтримувати екземпляри значно більшого розміру.

Безперервне розгортання. Оскільки кожний сервіс є одною незалежною змінною, тепер є можливість на регулярні та прискорені цикли

релізу. Тобто через те, що розгортання значно швидше, ніж у монолітного проекту, є можливість частіше віддавати якусь готову частину коду клієнту.

Легкість обслуговування та тестування. Команди можуть експериментувати з новими функціями та повертатися до попередньої версії, якщо щось не працює. Це спрощує оновлення коду та прискорює випуск нових функцій на ринок. Крім того, в окремих службах легко знаходити та виправляти помилки та баги.

Незалежне розгортання. Мікросервіси є окремими модулями, тому з ними можна легко і швидко виконувати незалежне розгортання окремих функцій.

Гнучкість технологій. При використанні архітектури мікросервісів команди можуть вибирати інструменти з урахуванням переваг.

Висока надійність. Розгортаючи зміни для конкретної служби, можна не боятися, що програма вийде з ладу повністю.

За статтею Ncube[1] можна добре побачити різницю між цими двома архітектурами (між монолітною – розробка всього коду в одному кодовому середовищі, як минулий стандарт проектів та мікросервісами) (рис. 1.6).

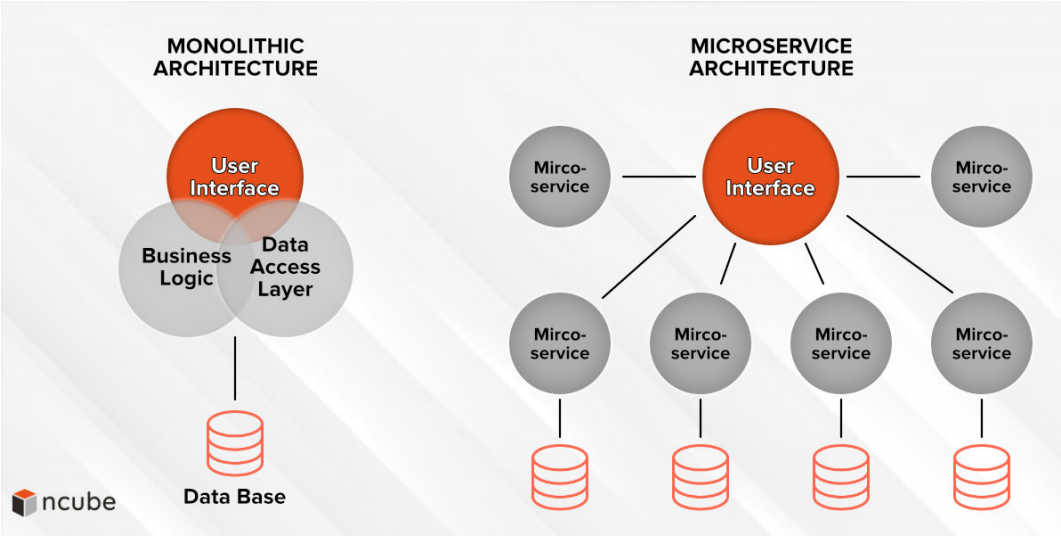

Рисунок 1.1 - Різниця між архітектурами [1]

1.3 Огляд та порівняння існуючих програм для вирішення задачі

1.3.1 Актуальність проблеми

Автоматизація документообігу – нажаль, майже повсякденна проблема. Є багато організацій, котрі зберігають документи у фізичному вигляді, тобто використовують папір. По-перше, це шкодить навколишньому середовищу. По-друге, очікування підпису та ухвалення документу дуже велике. І щоб якийсь документ пройшов перевірку, він має побути у декількох людей на огляді, після чого його можуть повернути на виправлення, що потребує ще більше ресурсів, або відправляють далі іншій людині, більш вищої посади зазвичай, на повторне переглядання. Це все втрата часу.

Вже є сучасні рішення цієї проблеми – програмні забезпечення, які вирішують усі ці рутинні проблеми. Можна навести декілька прикладів:

1. BAS документообіг КОРП [2];
2. JSolution [3];
3. FossDoc[4];
4. HotDocs [5].

Кожна з них по-своєму вирішує кожну з вище зазначених проблем. Але будь-яка програма має свої переваги, недоліки так унікальності у використанні. Також додаток може не підходити до якоїсь задачі або для організації. Слід уважно ставитися до вибору, бо від нього буде залежність успішність всій корпорації.

1.3.2 BAS документообіг КОРП

Рисунок 1.2 - BAS-застосунок [2]

«BAS Document Management Corp» — сучасне, популярне та високопродуктивне рішення для управління бізнес-процесами й командною роботою. Перевірені методи та практики, які допоможуть організувати електронний документообіг, налаштувати процеси, забезпечити контроль за виконанням завдань, регламентувати управлінську діяльність та підвищити її ефективність.

Компанія дає короткий та зрозумілий опис свого продукту за документацією [6]

Сфера застосування: підприємства будь-якої галузі із великим документообігом

Завдяки програмі можна зберігати документи, скоротити час на реєстрацію документів та їх пошук, забезпечує зручність роботи з єдиною базою даних документації.

Рішення BAS Документообіг КОРП дає змогу автоматизувати документообіг підприємства з нуля та швидко приступити до роботи.

Серед унікальностей можна зазначити:

1. Має шаблони (близько 62-ух) створення документів;
2. Сповіщення працівників про дедлайни підписання;

3. Готові форми для звітностей;

4. Фільтрація документів: наказ, договір тощо.

Вартість ліцензій BAS відповідає цінам, встановленим компанією-розробником, та однакова у всіх учасників "Спілки автоматизаторів бізнесу" [8].Нажаль безкоштовної програми немає навіть для тестування її. Тобто, не купивши додатку, не дізнаєшся зручності інтерфейсу, швидкодії та всього функціоналу. Також потрібно зазначити, що ціна не є низькою. З прайсу, розташованого на офіційному сайті, можна побачити, що найнижча ціна на фінансову базову частину «Малий бізнес» [7] становить 2240 грн (рис. 1.2). Для організації, яка тільки починає свій шлях не є кращим рішенням – придбання подібного.

Программные продукты "BAS"

Название продукта	Стоимость, грн	Бонусы при покупке
BAS ERP	180000	18000
BAS ERP. Ліцензія для дочірніх підприємств і філій	54000	5400
BAS Управління холдингом	600000	60000
BAS Документообіг. КОРП	62700	6270
BAS Управління торгівлею	8400	840
BAS Роздрібна торгівля	6690	670
BAS Бухгалтерія. Базова	2240	224
BAS Бухгалтерія ПРОФ	6690	670
BAS Бухгалтерія. КОРП	18000	1800
BAS Бухгалтерія. Комплект для 5 користувачів	12900	1290
BAS Модуль обліку акцизного палива (для програм "BAS Бухгалтерія ПРОФ" і "BAS Бухгалтерія КОРП")	4800	480
BAS Малий бізнес	8400	840
BAS Малий бізнес. Базова	2240	224

Рисунок 1.3 - Прайс-лист від BAS КОРП[3]

1.3.3 JSolution

Програма для електронного документообігу дозволяє підприємству, компанії, фірмі або підрозділу спростити роботу з документами та зменшити обсяги витраченого часу на їх упорядкування. За допомогою системи ви зможете систематизувати інформацію з різних типів

документів, скоротити час на пошук потрібних даних завдяки зручній функції відбору за заданими параметрами. Ведіть документацію в єдиній базі, розмежуйте доступ до неї за допомогою налаштування відповідних прав доступу для працівників. Комп'ютеризація роботи з документами не тільки робить роботу швидше, а й підвищує ефективність бізнес-процесів компанії.

Також компанія надає хмарне середовище для будь-яких сфер діяльності.

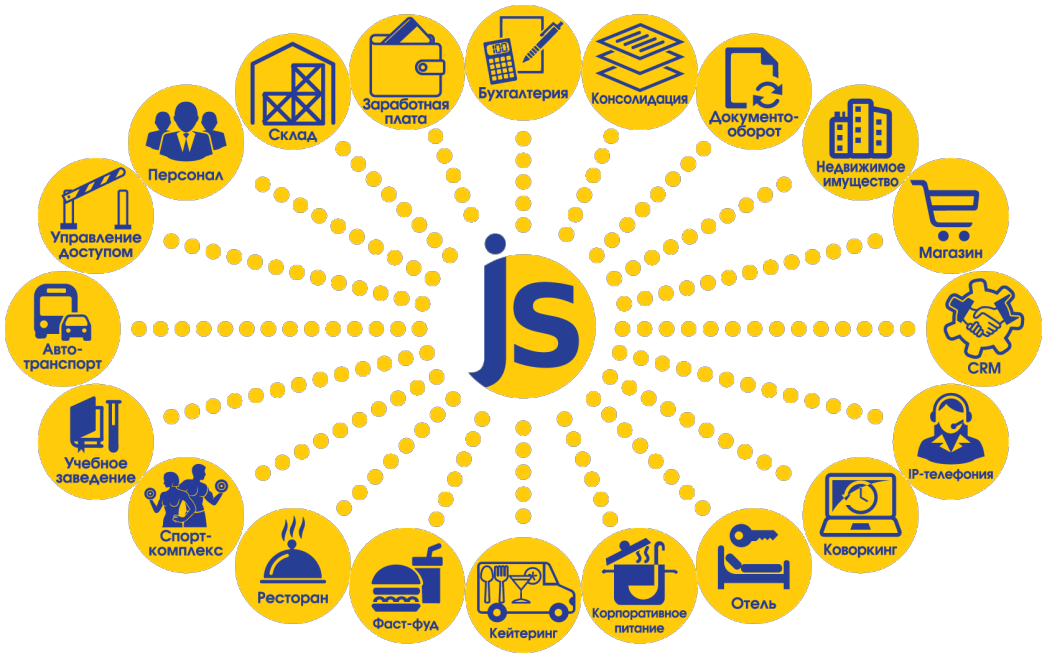

Рисунок 1.4 - Різноманіття JSolutions [3]

У системі є можливість створення та обліку наказів на підприємстві. Друк наказів здійснюється за заздалегідь налаштованими шаблонами, як стандартним (прийом на роботу, звільнення, створення нової посади, переклад, відпустку тощо), так і індивідуально розробленим під завдання конкретно вашої фірми. Усі вхідні (поштові, електронні) звернення фіксуються в системі, з можливістю відстеження листування по обігу, фіксації даних відправника та рішення щодо кожного окремого звернення

Модуль jSolutions для документообігу працює з різним програмним забезпеченням для друку. Використовуючи безкоштовні текстові та табличні редактори Open office [9], Libre office [10], клієнт мінімізує витрати на використання інформаційних технологій. Це є великою перевагою, адже такі сервіси у вигляді програми не часто є з підтримкою для декількох ОС.

Автоматизація документообігу це відмінний інструмент керування, адже всі внутрішні документи, що входять/виходять, завжди будуть в повному порядку. Ви завжди можете сформувати звіти про виконання, кількість отриманих документів за період, кількість документів, які потрібно списати до архіву та інших звітів, що налаштовуються під вимоги клієнта.

1.3.4 FossDoc

«Система електронного документообігу FossDoc — рішення на платформі FossLook[13], потрібна для створення електронного архіву документів, організації корпоративного документообігу та автоматизації бізнес-процесів на підприємствах, установах та організаціях будь-якого роду діяльності. Програма дозволяє вирішити велику кількість завдань, реалізація яких покладена відповідні модулі. Система може бути легко перенастроєна з урахуванням специфіки роботи кожного конкретного підприємства.» [4]

Сама програма побудоване на основі класичної клієнт-серверної архітектури, на самому сайті виробників є схема, яка ілюструє роботу їх додатку (рис. 1.4).

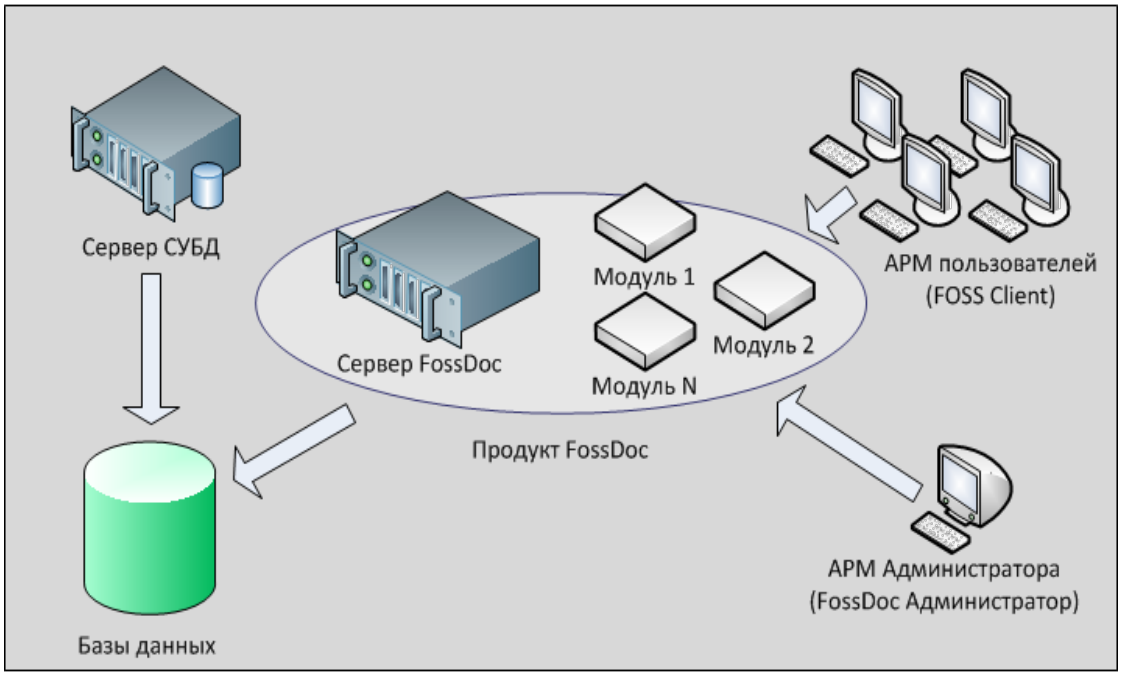

Рисунок 1.5 - Схема продукту FossDoc [4]

Сервер FossDoc — сервер для взаємодії клієнтських програм із сервером СУБД. Модулі, які позначають ролі системи, підключаються до сервера додатків. Тобто продукти будуть мати відмінності, що містять різні набори модулів.

«Web-сервер — дозволяє працювати з сервером програм за допомогою звичайного веб-браузера. Реалізований у вигляді зовнішнього модуля, що підключається до сервера додатків.» [4]

База даних – у даному випадку маємо змогу керувати за допомогою сервера СУБД (Oracle [11] або Microsoft SQL Server [12]). Зберігає документи, довідники, інформацію про користувачів, налаштування тощо.

FossDoc Client - клієнтський додаток-програма, яка, нажаль, є тільки для Windows ОС, автоматизований робочий (АРМ) користувача. Основна мета програми, як будь-якого клієнту – підключення звичайного користувача до сервера додатків (з локальної мережі або інтернету) і вирішує свої службові завдання, як співробітник підприємства (організації). У той же час АРМ користувача є клієнтом-поштою і може приймати-надсилати листи із зовнішніх поштових серверів.

FossDoc Web Client - веб-інтерфейс (веб-клієнт), який реалізує ті ж функції робочого місця користувача, що й звичайний FossDoc Client, за допомогою звичайного веб-браузера. Тобто компанія надала змогу використовувати їх клієнт-продукт з будь-якої операційної системи. Роботу веб-клієнта забезпечує веб-сервер системи.

FossDoc Адміністратор – подібна до FossDoc Client, клієнтська програма, майстер адміністрування. За допомогою цього ПЗ можна здійснювати адміністрування сервера програм (відтворення нових типів документів, розподіляти права доступу, підключати додаткові функції, вводити користувачів, проектувати маршрути тощо)

1.3.5 HotDocs

Рисунок 1.6 - HotDocs[5]

HotDocs — це набір відзначених нагородами програмних додатків, які значно скорочують час, який потрібно витрачати на створення документів для клієнта або клієнта (таких як контракти, пропозиції продажу, державні та судові форми, заявки на позику та медичні форми).

HotDocs перетворює документи та графічні (PDF) форми в шаблони для створення документів і розгортає ці шаблони в різних серверних середовищах.

Моделювання документів у HotDocs може варіюватися від вставок змінних до формування та вставок складних обчислених змінних. Бізнес-логіка, що складається з операторів IF/THEN і циклів REPEAT, може бути вбудована в шаблон для контролю включення або виключення мовних блоків. HotDocs включає в себе ряд інших інструкцій сценаріїв і наборів попередньо запакованих функцій з використанням булевої логіки. HotDocs також дозволяє системним архітекторам створювати власні функції.

Під час використання шаблон HotDocs запитує у користувача інформацію, необхідну для створення документа (або набору документів), і зберігає інформацію у файлі відповідей. Потім програма використовує збережену інформацію для створення спеціальної версії документа, вставляючи та форматуючи змінну інформацію, вставляючи правильні речення на основі умов транзакції та вставляючи правильні займенники та дієслова.

Стек технологій HotDocs включає логічне ядро, набір інструментів розробки, платформи для розгортання інтелектуальних шаблонів у будь-якому середовищі та широкий спектр технологій рівня користувача (веб-додатки для використання шаблонів HotDocs).

Специфічна, але теж свого роду автоматизування. Хоч такий вибір не повністю, а лише частково вирішує нашу поставлену задачу, але програма, як доповнення, може дуже пришвидшити час створення та заповнення документу.

1.4 Постановка задачі

Задано організацію з декількома відділами та працівниками з різними ролями в кожному відділу. Потрібно розробити програму на основі мікросервісної архітектури, яка дасть змогу завантажувати документи певного розміру та можна буде підтвердити правильність документу або погодитись на його умови, після чого програма перевірить наявність всіх погоджень та автоматично розробить підпис до документу від лиця відділу, якщо будуть виконані певні умови. Тобто документу буде надано підпис від обличчя відділу автоматично, якщо, наприклад, голова відділу підтвердить правильність документу. Також програма повинна мати функціонал перевірки підпису.

Також додаток повинен бути таким, щоб у майбутньому його можна було легко оновлювати та додавати новий функціонал дуже просто, тому що в розробці буде використана мікросервісна архітектура.

1.5 Висновки до розділу 1

Архітектура мікросервісів значно спростить масштабування та додавання нових можливостей до вашої програми. Тож якщо ви плануєте розробити велику програму з кількома модулями та мандрівками користувача, найкращим способом впоратися з цим буде шаблон мікросервісу.

Для користувача не важливо, як ви реалізуєте продукт, йому важливий результат та швидкодія.

Усі програми схожі між собою, але залишається невідомим шлях розробки та список технологій, які використовують вони. Є ще безліч аналогів, але відміні між ними присутні, так як кожний продукт повинен бути унікальним.

Але можна зазначити, що завдяки кожній з них можна отримати оптимізацію бізнес-процесів на підприємстві та такі переваги:

1. Електронний архів усіх документів, а також можливість працювати лише з актуальними версіями документів (тих, до яких було внесені останні зміни);
2. Шаблони файлів, просту та зрозумілу структуру та прискорення у декілька разів процесу узгодження;
3. Ще одна особливість – змога прив'язати до документів будь-яких аналітиків (фінансових, наприклад), щоб встановити взаємозв'язок між бюджетуванням та електронним документообігом;
4. У компаніях з декількома філіями прискорюється та спрощується процес узгодження та передачі файлів;
5. Систему можна налаштувати таким чином, щоб певні зміни в документах запускали супутні процеси, наприклад після оплати починалося виконання замовлення;
6. Покращений досвід роботи з клієнтами. Інтелектуальна автоматизація документів дозволяє командам отримувати

інформацію з низки підключених джерел і миттєво створювати контент, який розповідає історію за допомогою персоналізованих точок даних;

7. Незважаючи на першу програму BAS КОРП, велика економія грошей. Хоч і потрібно у деяких програмних забезпеченнях оформлювати платну підписку, все одно це менші витрати, ніж купляти папір та техніку з фарбою для друку.

Слід уважно ставитися до вибору технологій та програм, якими буде користуватися вся компанія.

2 ОПИС СТЕКУ ТЕХНОЛОГІЙ

2.1 Spring Boot

Java Spring Framework (Spring Framework) — це популярна платформа корпоративного рівня з відкритим кодом для створення автономних додатків виробничого рівня, які працюють на віртуальній машині Java (JVM).

Java Spring Boot (Spring Boot) [14] — це інструмент, який робить розробку веб-додатків і мікросервісів за допомогою Spring Framework швидшою та простішою за допомогою трьох основних можливостей:

1. Автоналаштування
2. Упевнений підхід до налаштування
3. Можливість створювати окремі програми

Ці функції працюють разом, щоб надати вам інструмент, який дозволяє налаштувати програму на основі Spring з мінімальною конфігурацією та налаштуванням.

Серед переваг можна зазначити такі:

1. Він забезпечує гнучкий спосіб налаштування Java Beans, конфігурацій XML і транзакцій бази даних;
2. Він забезпечує потужну пакетну обробку та керує кінцевими точками REST;
3. У Spring Boot все налаштовується автоматично; не потрібні ручні налаштування;
4. Він пропонує додаток на основі анотацій (тобто потрібно над функцією або змінною поставити кодове слово для використання певного коду);
5. Полегшує управління залежностями;
6. Він включає в себе вбудований контейнер сервлетів.

2.2 Gradle

Gradle [15] — це система керування збірками загального призначення. Він підтримує автоматичне завантаження та налаштування залежностей або інших бібліотек. Він підтримує репозиторії Maven та Ivy для отримання їх залежностей.

Переваги:

1. Широко настроюваний — Gradle моделюється таким чином, що його можна настроювати та розширювати найбільш фундаментальними способами;
2. Швидкість — Gradle швидко виконує завдання, повторно використовуючи вихідні дані з попередніх виконання, обробляючи лише вхідні дані, які змінилися, і виконуючи завдання паралельно;
3. Потужний — Gradle є офіційним інструментом для створення Android, який підтримує багато популярних мов і технологій;
4. Gradle підтримує створення кількох проектів і артефактів.

2.3 Spring Cloud Api Gateway

Шлюз [16] - це окрема Spring Boot програма, через яку проходять всі запити, реалізація шаблону Reverse Proxy. Тобто мікросервіси не знають один про одного, а звертаються до проксі. Зовнішній користувач також відомий тільки проксі. Проксі, у свою чергу, аналізує запит, перенаправляє його до потрібного мікросервісу та повертає відповідь назад.

Шлюз API може допомогти спростити зв’язок між клієнтом і службою, чи то між веб-браузером користувача і сервером, віддаленим за милі, або між інтерфейсним додатком і серверними службами, на які він покладається.

Шлюз API дозволяє реалізувати функціонал окремо від клієнта, переміщуючи цю відповідальність з боку користувача на сторону сервера. Все, що клієнт повинен знати, це як спілкуватися зі шлюзом. Не має

значення, чи переміщуються серверні служби, чи переходять у автономний режим чи стають нестабільними, якщо шлюз знає, як впоратися з такими ситуаціями.

Під капотом шлюз має не дуже важку схему (рис. 2.1).

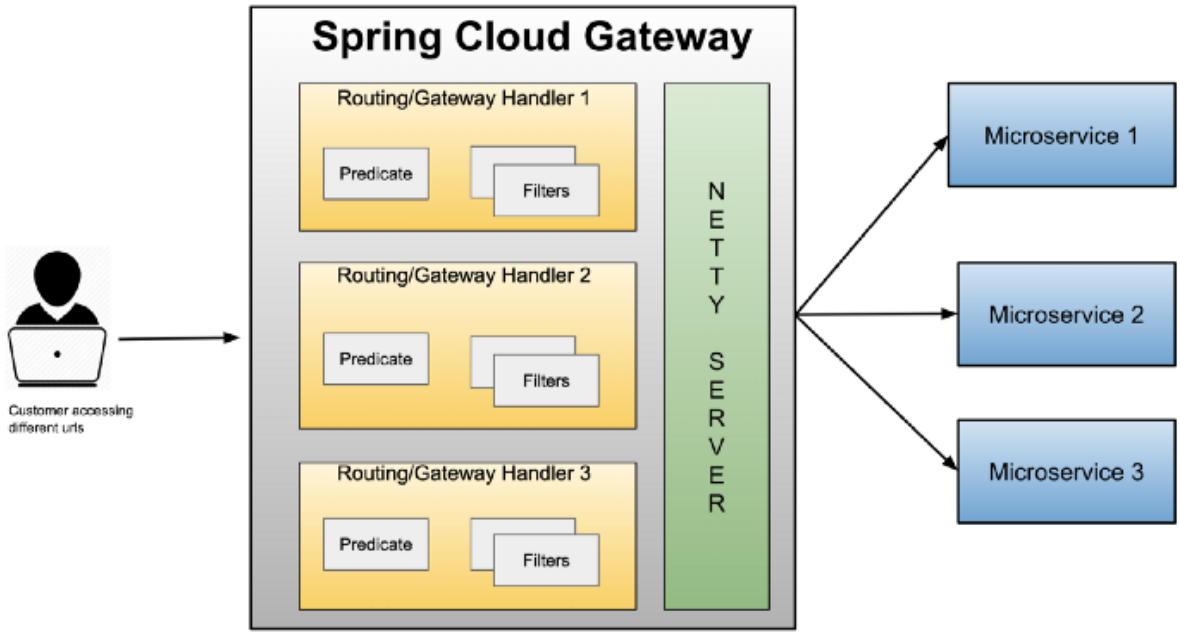

Рисунок 2.1 - Елементи Spring Cloud Api Gateway

Щоб зіставити вхідний url (що йде в Spring Cloud Gateway) вихідному url (що йде до мікросервісу), потрібно задати три пункти:

Predicate: умова, коли запит перенаправляється (наприклад, якщо url відповідає такому-то шаблону).

URI: містить шлях-силку куди перенаправляємо запит (який мікросервіс).

Фільтр (необов'язково): як модифікувати запит (на шляху туди чи назад).

Ці три пункти (ще ідентифікатор Route) об'єднані в Route - основний будівельний блок. З кількох таких блоків і складається налаштування

2.4 Spring Cloud Config Server

Це клієнт-серверний підхід Spring для зберігання та обслуговування розподілених конфігурацій у кількох додатках та середовищах. [17]

Це сховище конфігурації має означення версіями під керуванням Git version control і може бути змінено під час виконання програми. Хоча він дуже добре підходить для додатків Spring, що використовують всі підтримувані формати файлів конфігурації разом з конструкціями, такими як Environment, PropertySource або @Value, він може використовуватися в будь-якому середовищі, що працює будь-якою мовою програмування. Приклад наведено на рисунку 2.2.

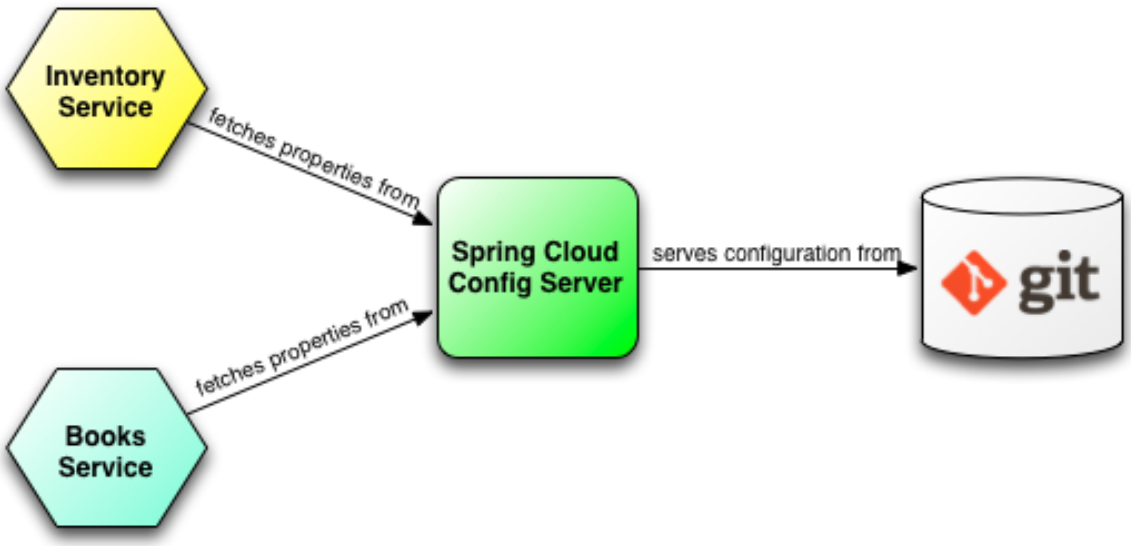

Рисунок 2.2 - Схема Cloud Server

Перед кожним запуском сервісу, він спочатку дивиться на сховище з конфігураціями і використовує їх перед основною автоконфігурацією проекту. Одна з особливостей – зміна конфігурації тоді, коли проект вже запущений. Тобто проект час від час питає у сховища чи були зміни, і якщо такі були, заміняє їх майже не помітно для клієнтів.

2.5 Netflix Eureka Server

Відкриття служби на стороні клієнта дозволяє службам знаходити та спілкуватися один з одним без жорсткого кодування імені хоста та порту. Єдиною «фіксованою точкою» в такій архітектурі є реєстр послуг, у якому кожна служба має зареєструватися.

Одним з недоліків є те, що всі клієнти повинні реалізувати певну логіку для взаємодії з цією фіксованою точкою. Це передбачає додаткову мережу туди й назад перед фактичним запитом.

З Netflix Eureka [18] кожен клієнт може одночасно діяти як сервер, щоб реплікувати свій статус підключеному однорангові. Іншими словами, клієнт отримує список усіх підключених однорангових партнерів у реєстрі служб і робить усі подальші запити до інших служб за допомогою алгоритму балансування навантаження.

Щоб отримати інформацію про присутність клієнта, вони повинні надіслати сигнал «серцевого ритму» в реєстр.

Кожен мікросервіс реєструється на сервері Eureka, і Eureka знає всі клієнтські програми, що працюють на кожному порту та IP-адресі.

Якщо простими словами, то це сервер імен або реєстр сервісів. Обов'язки:

1. Надавати імена кожному мікросервісу.
2. Реєструє мікросервіси та віддає їх IP іншим частинам програми. Таким чином, кожен сервіс реєструється в Eureka і відправляє запит серверу, щоб повідомити, що він активний.

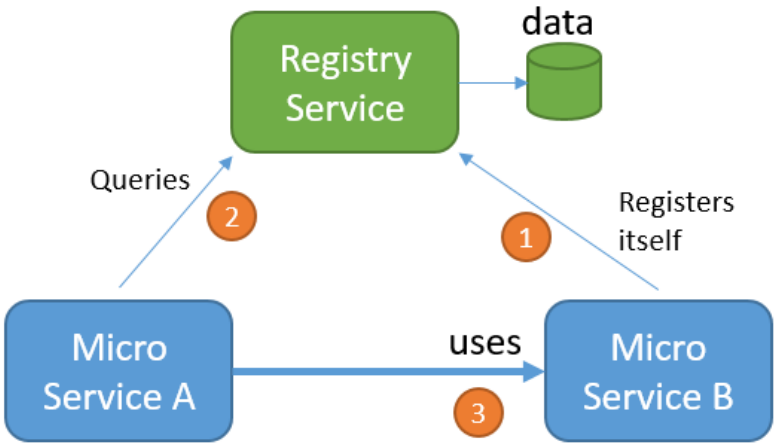

Рисунок 2.3 - Схема роботи Eureka Server

2.6 MongoDb

Документоорієнтована система управління базами даних, яка потребує опису схеми таблиць. Вважається одним із класичних прикладів NoSQL-систем, використовує JSON-подібні документи та схему бази даних [20]. JSON[19] або JavaScript Object Notation — це мінімальний, читабельний формат для структурування даних. Він використовується в основному для передачі даних між сервером і веб-додатком, як альтернатива XML.

Як і інші бази даних NoSQL, MongoDB не вимагає попередньо визначених схем. Він зберігає будь-які типи даних. Це дає користувачам можливість створювати будь-яку кількість полів у документі, полегшуючи масштабування баз даних MongoDB порівняно з реляційними базами даних.

Основною функцією MongoDB є його горизонтальна масштабованість, що робить його корисною базою даних для компаній, що працюють із додатками для великих даних. Крім того, шардінг дозволяє базі даних розподіляти дані між кластером машин. Нові версії MongoDB також підтримують створення зон даних на основі ключа шарда.

MongoDB підтримує ряд механізмів зберігання даних і надає API механізмів зберігання, які дозволяють третім сторонам розробляти власні

механізми зберігання. Саме у Spring Boot не виключення і в ньому буде використовуватися монго апі для зберігання та оновлення даних.

2.7 MongoCompass

MongoDB Compass [21] є офіційним графічним інтерфейсом для MongoDB, який підтримується офіційним розробником бази даних. MongoDB Compass допомагає користувачам приймати розумні рішення щодо структури даних, запитів, індексування та багатьох інших дій, які ви можете виконувати з базою даних.

MongoDB Compass є набагато кращою альтернативою для оболонки Mongo. Compass може виконувати всі операції, які можна виконувати через командну строку або через термінал (Mongo Shell), і багато іншого, зокрема:

1. Візуалізувати та досліджувати дані, що зберігаються у базі даних;
2. Створювати бази даних та вставляти, оновлювати та видаляти дані у своїй базі даних;
3. Отримувати негайну статистику сервера в реальному часі;
4. Відстежувати та розуміти проблеми продуктивності за допомогою візуального пояснення планів;
5. Керувати індексами;
6. Перевіряти свої дані за допомогою правил перевірки схеми JSON;
7. Розширюється за допомогою плагінів.

2.8 Redis

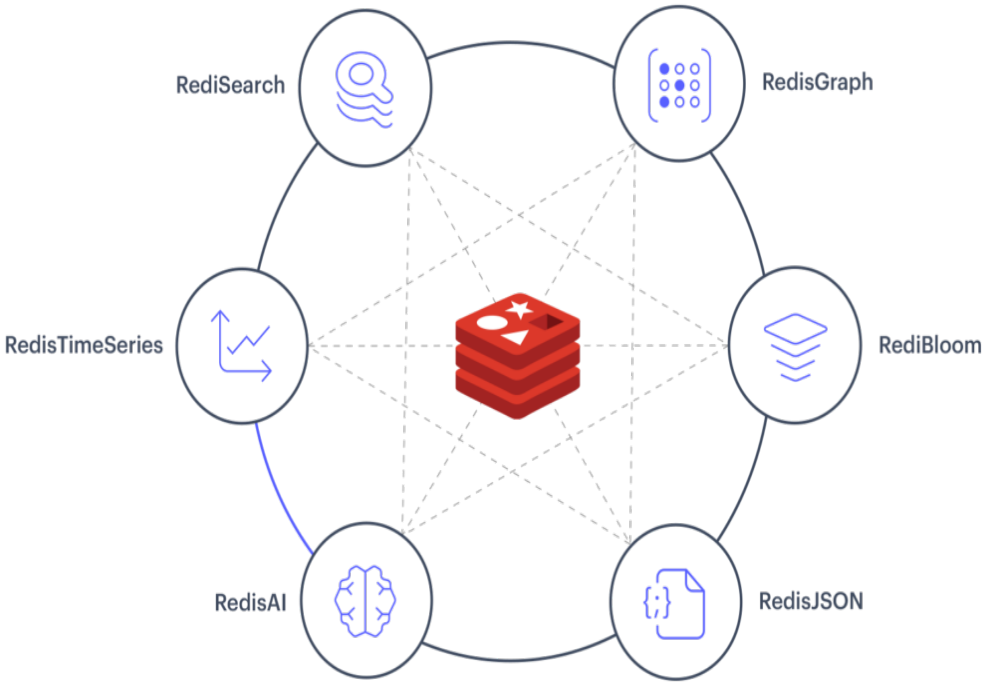

Рисунок 2.4 - Redis база даних

Redis [22] — це сховище з відкритим вихідним кодом, сховище в оперативній пам’яті, яке використовується як база даних, хеш-пам’ять, посередник повідомлень і механізм потокової передачі. Redis надає такі структури даних, як листи, хеші, списки, набори, відсортовані набори із запитами діапазону, растрові зображення, гіперлогові журнали, геопросторові індекси та потоки. Redis має вбудовану реплікацію, сценарії Lua, транзакції та різні рівні збереження на диску.

Є можливість виконувати атомарні операції з цими типами, наприклад, додавати до рядка; збільшення значення в хеші, переміщення елемента до списку, обчислювальний набір перетину, об'єднання та різниці; або отримання члена з найвищим рейтингом у відсортованій групі, тобто ця база даних поєднує в собі всю булеву алгебру як і будь-яка інша.

Щоб досягти найвищої продуктивності, Redis працює з набором даних у пам’яті. Залежно від вашого випадку використання, Redis може зберігати ваші дані, періодично виписуючи набір даних на диск, або додаючи кожну

команду до журналу на диску. Також є така функція як вимкнути збереження, якщо вам просто потрібен багатофункціональний, мережевий кеш у пам’яті.

Redis підтримує асинхронну реплікацію, швидку неблокуючу синхронізацію та автоматичне повторне підключення з частковою ресинхронізацією при розділенні мережі.

Використовується як для баз даних, так і для реалізації кешу, брокерів повідомлень. Вона є не дуже безпечною для збереження даних, тому її краще використовувати для кешу. У власній роботі буде використано для збереження jwt токену, який буде використовуватися для авторизації користувача

2.9 JWT

JWT [23]- це запропонований Інтернет-стандарт для створення даних із додатковим підписом та/або необов’язковим шифруванням.

«Це відкритий стандарт (RFC 7519) для створення токенів доступу, що базується на форматі JSON. Як правило, використовується для передачі даних для автентифікації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує цей токен для підтвердження своєї особи».

Токен складається з трьох частин: корисного навантаження (англ. payload), заголовка (англ. header) та підпису або даних шифрування. Перші два елементи – це JSON об'єкти певної структури. Третій елемент заложить від алгоритму для створення за яким, з використанням даних перших компонентів (у разі використання непідписаного JWT, може бути опущений)

Коротше кажучи, JWT використовуються як безпечний спосіб аутентифікації користувачів і обміну інформацією.

Як правило, приватний ключ або секрет використовується емітентом для підписання JWT. Одержувач JWT перевірить підпис, щоб переконатися, що токен не був змінений після того, як його підписав емітент.

Проте не всі алгоритми підписання створені однаковими. Наприклад, деякі алгоритми підпису використовують секретне значення, яке спільно використовують емітент і сторона, яка перевіряє JWT. Інші алгоритми використовують відкритий і закритий ключ. Закритий ключ відомий лише емітенту, тоді як відкритий ключ може бути широко поширений. Для перевірки підпису можна використовувати відкритий ключ, але для створення підпису можна використовувати лише закритий ключ. Це більш безпечно, ніж загальний секрет, оскільки приватний ключ повинен існувати лише в одному місці.

Через це серверу не потрібно зберігати базу даних з інформацією, необхідною для ідентифікації користувача. Для розробників це чудова новина — сервер, який видає JWT, і сервер, який його перевіряє, не повинні бути однаковими. Схема роботи зображена на рисунку 2.5.

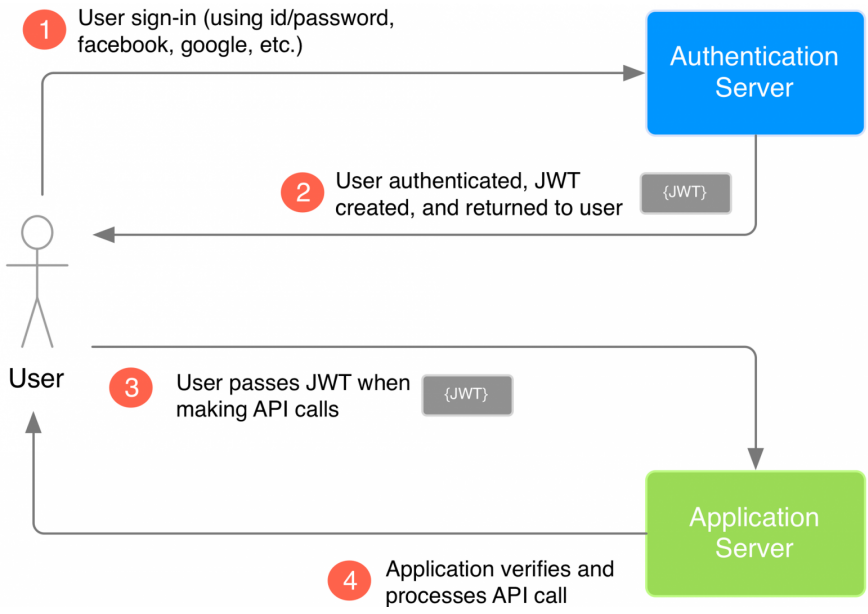

Рисунок 2.5 - Схема роботи JWT

2.10 Висновок до розділу 3

Автоматизація документообігу – об’ємна робота и її можно розбити не декілька частин, тобто мікросервісів. Слід підібрати такий інструментарій, щоб було зручніше розробляти додаток самотужки, бо самому працювати на декількох сервісах – складна робота. Саме завдання було виконано з використанням Spring Boot фреймворку на мові Kotlin[24] у середовищі Intellij IDEA[25].

Для більш зручної розробки було використано такий стек технологій від Spring Boot та деякі інші:

1. Spring Cloud Api Gateway
2. Spring Cloud Config Server
3. Netflix Eureka Server
4. MongoDb
5. Redis
6. Jwt principle

Кожна частина дуже важлива в розробці. Ознайомлення з вже існуючими рішеннями тієї чи іншої задачі завжди дає пришвидшення в розробці, бо навіщо кожного разу створювати новий велосипед.

Також маємо архітектуру, в котрій немає циклічних зав’язків між сервісами . Також можна побачити, що в нас є спілкування лише між двома сервісами всередині блоку всіх інших (user-service -> department-service), тобто поки що наша схема виглядає так, що ми не маємо плутанини між спілкуванням сервісів, все виглядає прозоро та зрозуміло.

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис архітектури

За поставленою задачею потрібно розробити такі сервіси:

1. Сервіс автентифікації – забезпечує розпізнавання користувача за його електронною поштою та паролем, які зберігаються у базі даних;
2. Сервіс користувача – основний функціонал для роботи з даними користувача (наприклад, створення робітника);
3. Сервіс відділу – має подібні функції, як у сервісу з пункту 2, але стосовно відділу адміністрації;
4. Сервіс документів – реалізує збереження до серверу та скачування документів. Основна задача цього сервісу – підтвердження документів та здійснення підпису, якщо будуть виконані певні умови;

Через описаний вище стек ми можемо представити серверну частину за схемою, наданою розробниками Spring Boot (рис. 3.1)

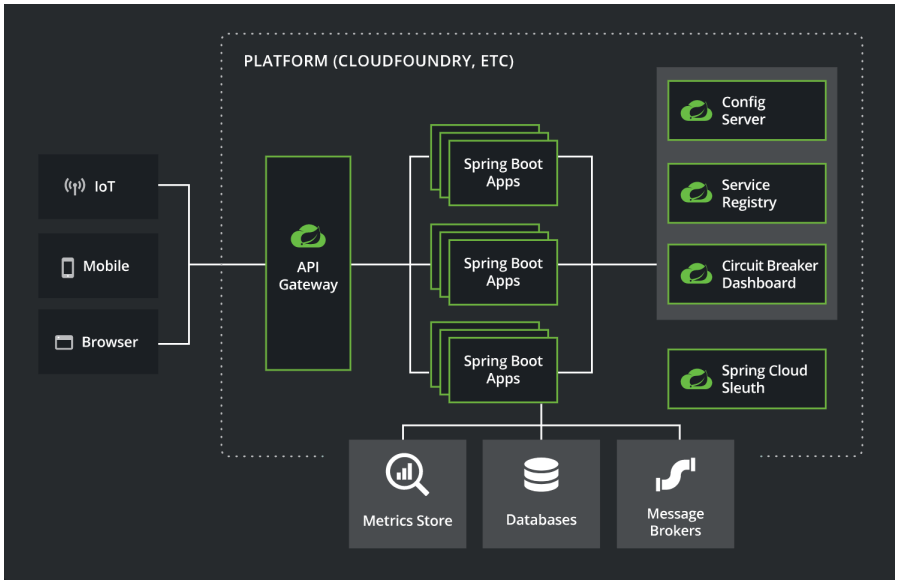

Рисунок 3.1 - Вигляд мікросервісної архітектури від Spring

Отже маємо, що користувач зможе використовувати наш додаток завдяки браузеру. А з браузера йдуть запити до серверу, який має доступ до баз даних (рис.3.2).

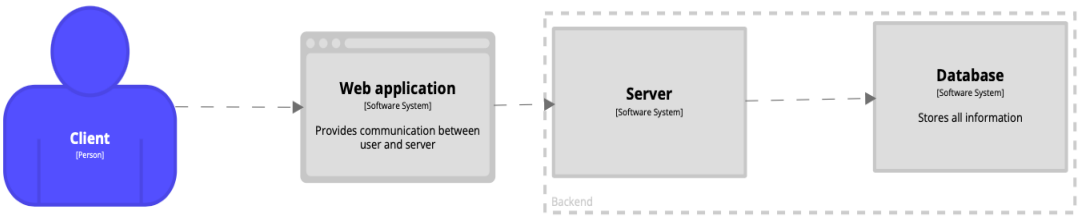

Рисунок 3.2 - Загальний опис нашої архітектури

На рисунку 3.3 зображено схему як виглядає архітектура проекту в середині блоку “Server”.

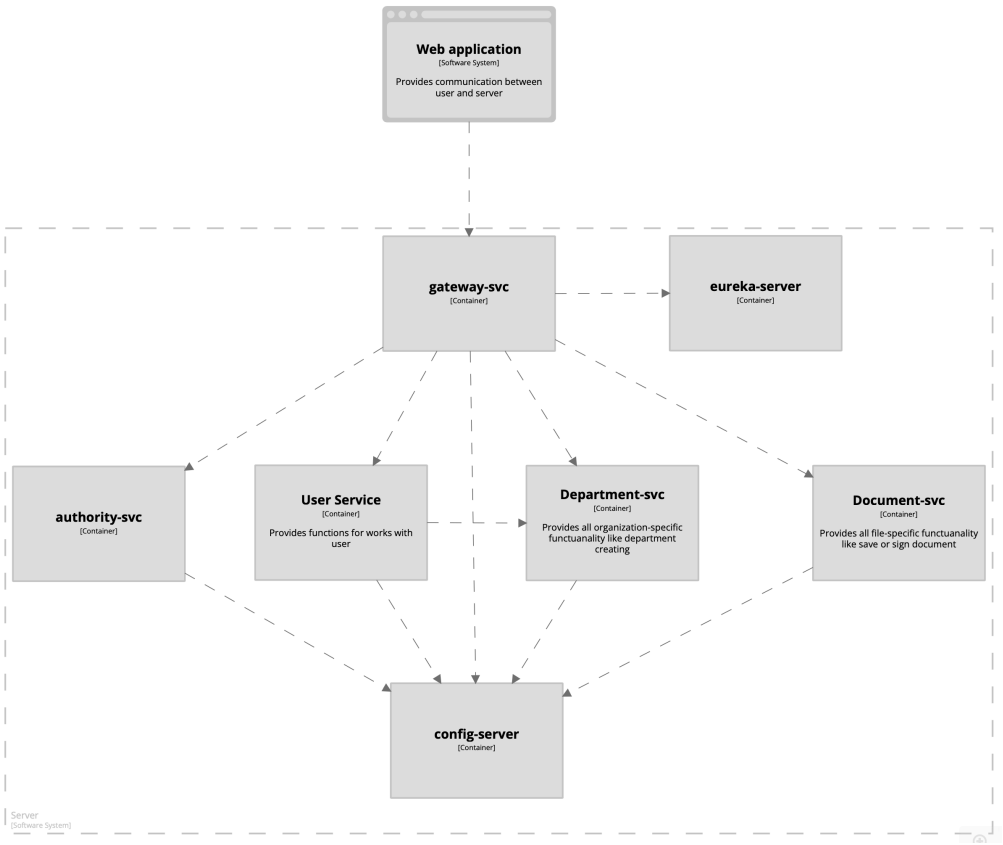

Рисунок 3.3 - Схема сервісів блоку "Server"

3.2 Комунікація між сервісами. Комунікація клієнт-сервер

За схемою, коли ми запускаємо наш сервер, кожний з сервісів реєструє себе та свій ідентифікатор у Eureka-server. Для доступу до будь-якого с сервісу клієнт надсилає запит до шлюзу, який сам вирішує, якому мікросервісу потрібен запит від користувача. Шлюз не має знати адреси кожного з сервісу (вони можуть бути динамічними), це не його робота, йому слід просто звернутися до Eureka-server за певним ідентифікатором, а той вже надає інформацію, за якою адресою можна надіслати запит.

Кожний з сервісів звертаються до сервісу-конфігурації. Це значить, що нам не потрібно зберігати конфігурацію на кожному с сервісів. А коли настройки знаходяться в одному місці, їх набагато легше узгоджувати між собою.

Spring boot дає чудову можливість повертати клієнту дані в залежності від запиту. Спочатку згідно схемі дані йдуть до gateway, він перевіряє за допомогою authority-svc, чи авторизовано користувача та перевіряє сам запит, чи може його обробити хоч один з сервісів.

Сам же сервіс-шлюз визначає, куди відправляти за допомогою серверу Eureka, на якому при запуску сервісу проходить реєстрація. Також бібліотека Eureka дає візуалізацію, приклад на рисунку 3.4, вже зареєстрованих сервісів, які можемо подивитися за певним шляхом-посиланням.

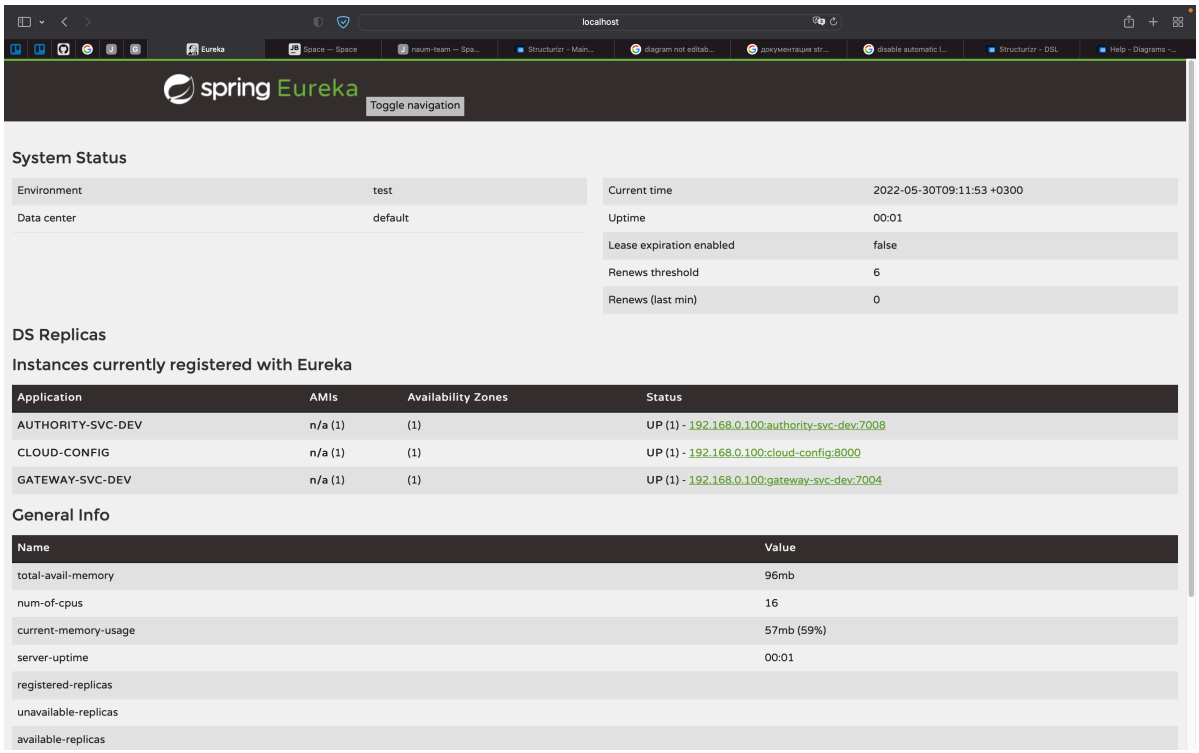

Рисунок 3.4 - Приклад зареєстрованих сервісів

Для спілкування між самими сервісами використовуються файли Proto[27] та OpenFeign[28] бібліотека.

OpenFeign – це декларативний клієнт веб-служби. Його зовнішній вигляд спрощує розробку клієнта веб-служби. Щоб використовувати Feign, вам потрібно тільки створити інтерфейс і додати відповідні інструкції, такі як інструкції FeignClient. Feign має анотації, що підключаються, включаючи анотації Feign і анотації JAX-RS. Feign також підтримує кодувальники та декодери. Spring Cloud Open Feign покращує Feign для підтримки анотацій Spring MVC і може використовувати HttpMessageConverters, такі як Spring Web

3.3 Сервіс автентифікації

На схемі 3.5 зображений приклад запиту авторизації користувача:

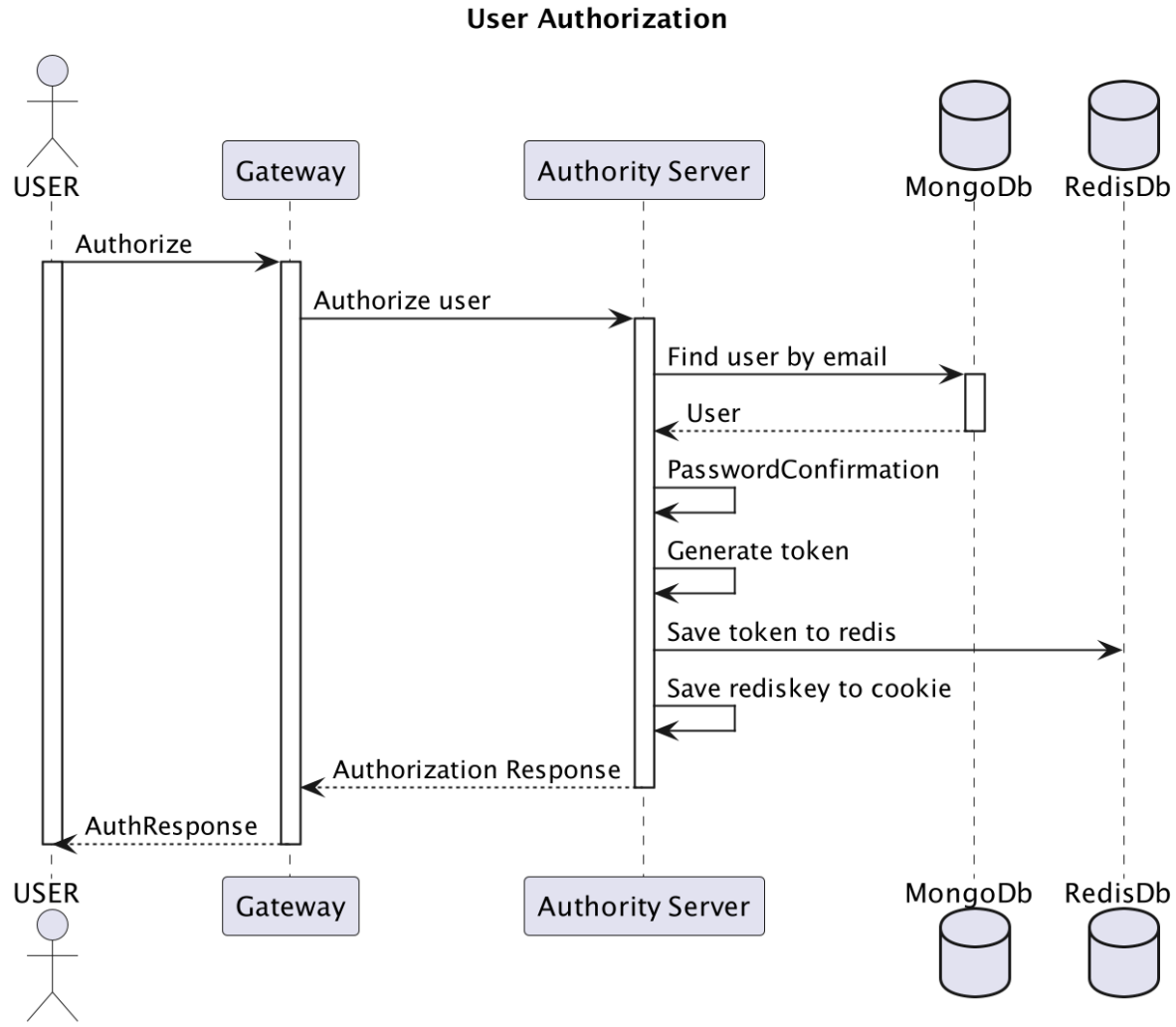

Рисунок 3.5 - Схема автентифікації користувача

Для авторизації проходимо такі етапи:

1. Користувач робить запит авторизації, надіславши пошту та пароль;
2. Шлюз. Визначає, що подібний запит потрібно надіслати до сервісу автентифікації і відправляє туди дані, введені користувачем;
3. Сервіс надсилає запит для знаходження клієнту в базі даних за поштою і має два випадки: якщо не було знайдено, повертається

помилка, що не було знайдено, якщо знайдено виконується наступний крок;

4. У знайденого користувача з бази даних порівнюється хеш паролю з бази даних з хешем надісланого паролю. При нерівності відправляється помилку клієнту;
5. Генерація jwt з використанням даних користувача, раніше знайденого у базі даних, як його ідентифікатор, ПІБ, хеш паролю, роль у відділі, ідентифікатор відділу, до якого він належить;
6. Створюється ключ з хешованої пошти та паролю користувача за яким зберігається JWT.
7. Зберігається до redis-сховища та кукі браузера;
8. Надсилається відповідь користувачу.

Надалі шлюз при виконанні будь-якого запиту буде спочатку перевіряти, чи авторизований користувач, дістаючи з кукі браузера ключ для редісу (які були збережені у хешованому вигляді), і валідує його завдяки існуючій бібліотеці java.

Також на цьому сервісі є функція виходу з системи. У цьому випадку все значно легше:

1. Користувач надсилає запит;
2. Сервіс видаляє ключ з редісу-сховища.

І тепер шлюз не зможе знайти потрібний ключ і буде видавати помилку про те, що користувач не автентифікований при будь-якому запиті.

3.4 Сервіс користувача

Основною метою даного сервісу буде реєстрація клієнтів, яку зможе виконувати лише хтось з співробітників. Так зроблено, бо потрібно, щоб хтось з компанії вирішував, до якого відділу буде відправлений працівник. Приблизна схема буде виглядати так, як на рисунку 3.6.

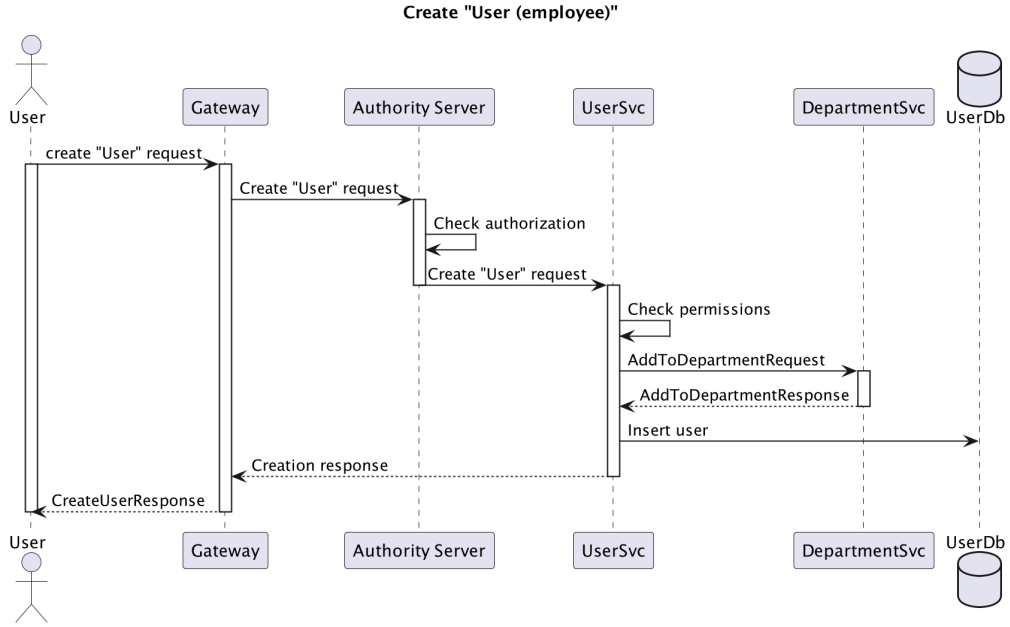

Рисунок 3.6 - Схема створення користувача

Маємо таку історію виконання даної схеми:

1. Користувач надсилає запит;
2. Шлюз перевіряє автентифікацію клієнту;
3. Відправляє запит до сервісу;
4. Проходить перевірка, чи може співробітник створювати користувачів;
5. Відправлення запиту далі до сервісу відділу;
6. Перевірка, чи можна додати користувача до відділу;
7. Відправлення відповіді до сервісу користувача;
8. При успішному результаті зберігається користувач до бази даних;
9. Відправлення відповіді клієнту.

Для безпечності даних користувача пароль у чистому вигляді не зберігається, а обчислюється хеш а вже потім надсилається до бази даних. На рисунку 3.7 зображено сутність користувача, з якою працює сервер:

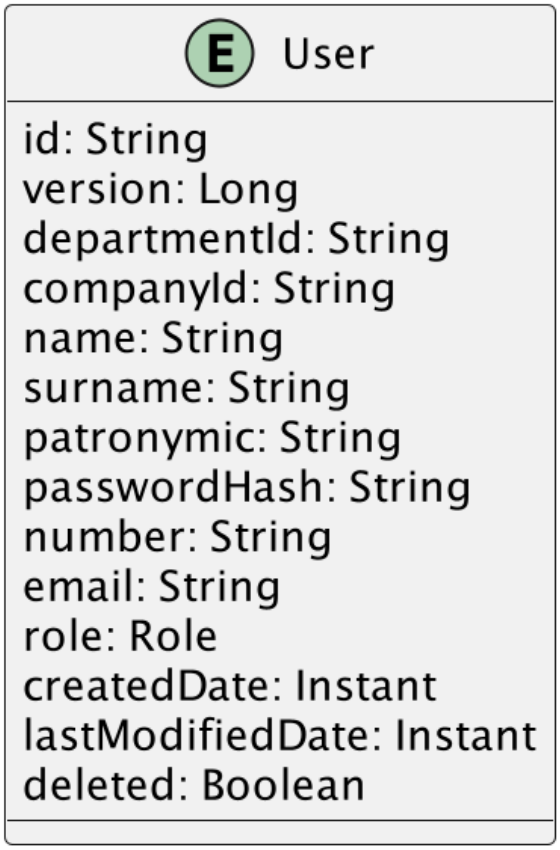

Рисунок 3.7 - Сутність користувача

А так виглядає його дані у базі даних MongoDB (рис. 3.8).

Рисунок 3.8 - Сутність користувача у базі даних

3.5 Сервіс відділу

Даний сервіс має таку ж саму мету, як і сервіс користувача, з однією відмінністю, він буде направляти запит до сервісу адмінстрації (рис. 3.9).

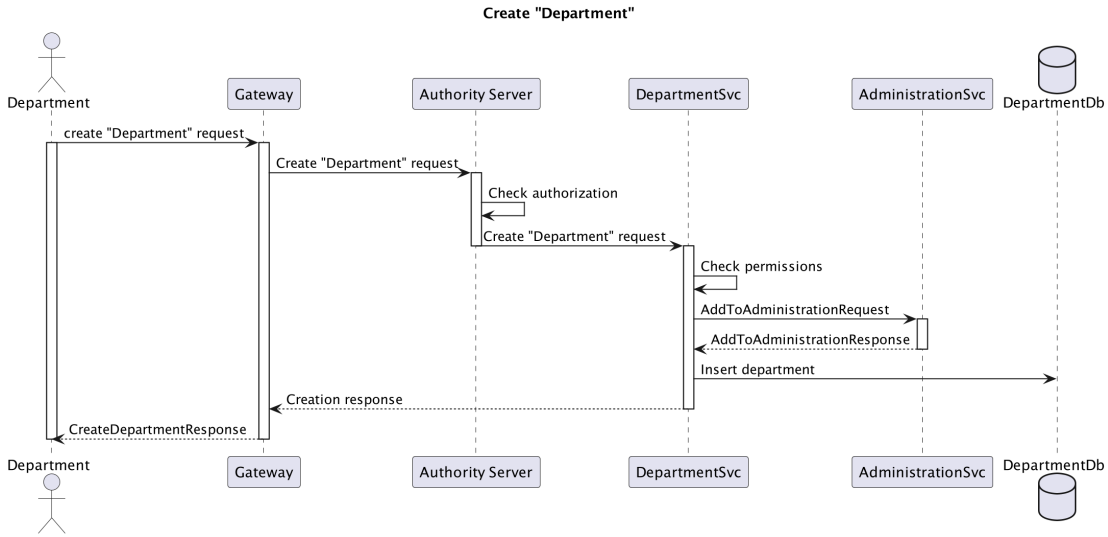

Рисунок 3.9 -Схема створення відділу

На рисунку 3.9 та 3.10 зображена сутності відділу та його типу, з якими працює сервер:

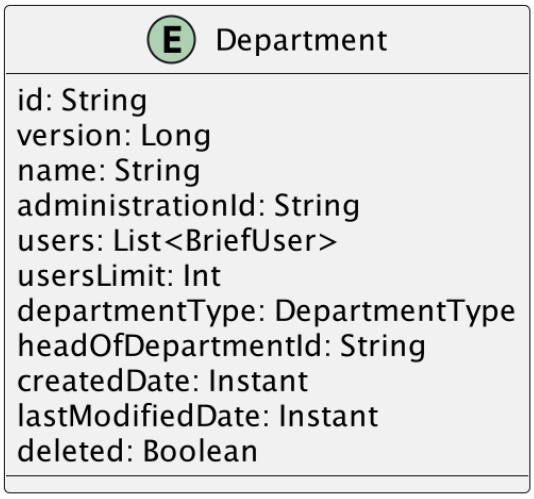

Рисунок 3.9 - Сутність відділу

Примітка. Ми не розглядаємо сервіс адміністрації, тому що вона не має великої цінності для розробки диплому, а розгортка додаткового сервісу займає купу часу. Тому під капотом проекту тимчасово буде емуляція адміністрації, яку можна буде легко створити у майбутньому

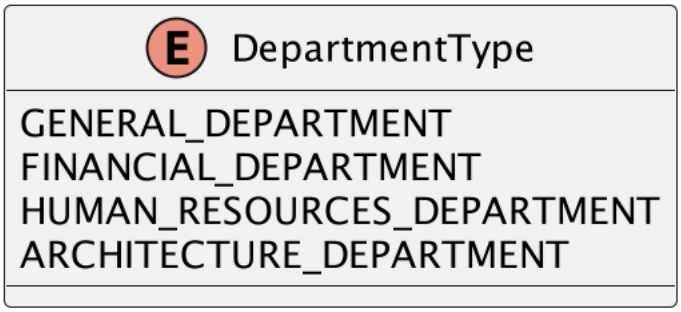

Рисунок 3.10 - Види відділів

На рисунку 3.11 вигляд того, як його дані відображуються у базі даних MongoDB


```
_id: ObjectId('62950cffc47ca8267f8a73a4')
administrationId: "62950cffc47ca8267f8a73a3"
name: "qFGAKzsXUk"
headOfDepartmentId: "FKNKcPtITYDnfHGsfVm"
› users: Array
usersLimit: 10
departmentType: "GENERAL_DEPARTMENT"
createdDate: 2022-05-30T18:29:19.674+00:00
lastModifiedDate: 2022-06-15T16:01:59.985+00:00
version: 2
deleted: false
_class: "learning.diplom.department.svc.model.Department"
```


Рисунок 3.11 - Сутність відділу у базі даних

3.6 Сервіс документів

Даний сервіс може виконує всі функції для роботи з файлами. Клієнт може отримати список документів своїх документів, скачати документ (або архів, якщо підписано), підтвердити документ, та підписувати його за певних умов.

З загрузкою файлу до серверу, або скачування з бази даних зрозуміло, це звичайні запити до БД. Після загрузки файлу, до нього в базі даних MongoDB вписується метадата з інформацією, хто загрузив файл, з якого відділу і створюються поля для підтвердження файлу, для підпису та публічного ключа, які скачуються разом з файлом, якщо підписаний.

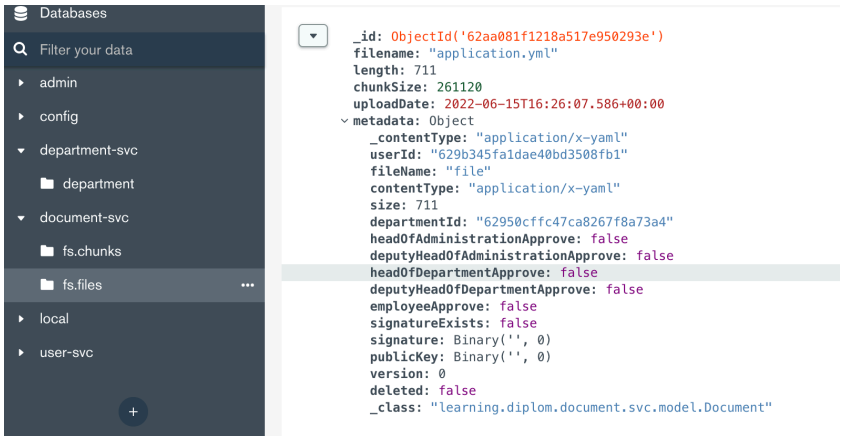

Рисунок 3.12 - Приклад доданого файлу у БД

Трохи складніше з підтвердженням документу, тому зроблено схему 3.13 для кращого розуміння та легшої розробки.

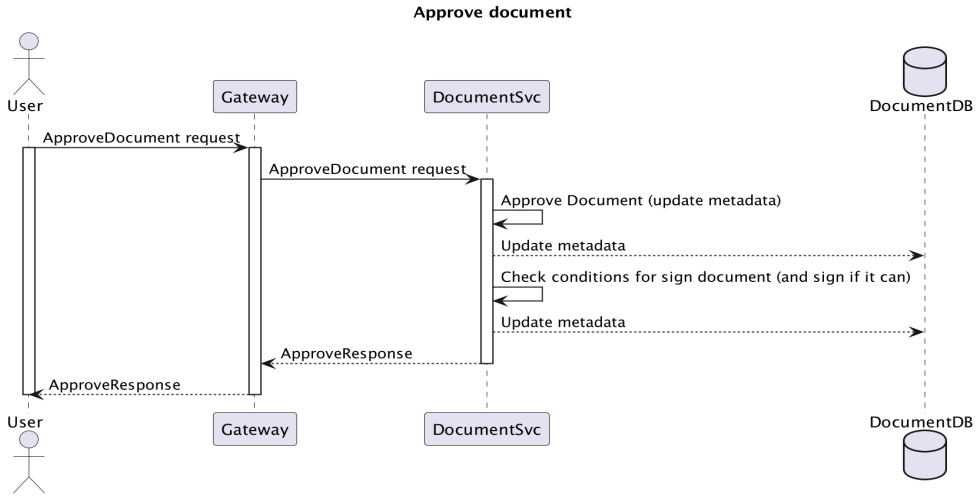

Рисунок 3.13 - Схема підтвердження документу

Опис схеми:

1. Як завжди запит від користувача до шлюзу;
2. Шлюз перенаправляє запит до сервісу документу;
3. Сервіс перевіряє роль користувача, який зробив запит та надсилає до бази даних оновлення метадати файлу, що певна роль підтвердила файл;
4. Одразу після підтвердження здійснюється перевірка умов для підпису файлу, наразі є дві умови: підтвердження головою відділу або підтвердження заступником голови відділу та звичайним робітником. Якщо умова не здійснена, то нічого не відбувається і надсилається відповідь клієнту.
5. Якщо були виконані умови для підпису, то сервер визначає хеш всього файлу та на його основі разом приватним ключем відділу створює підпис;
6. Надсилає відповідь до користувача.

Для підпису використовуються пара DSA ключів (приватний та публічний), які створюються разом з відділу та зберігаються до бази даних. На рисунку 3.14 відображено, як виглядають ключі у БД.

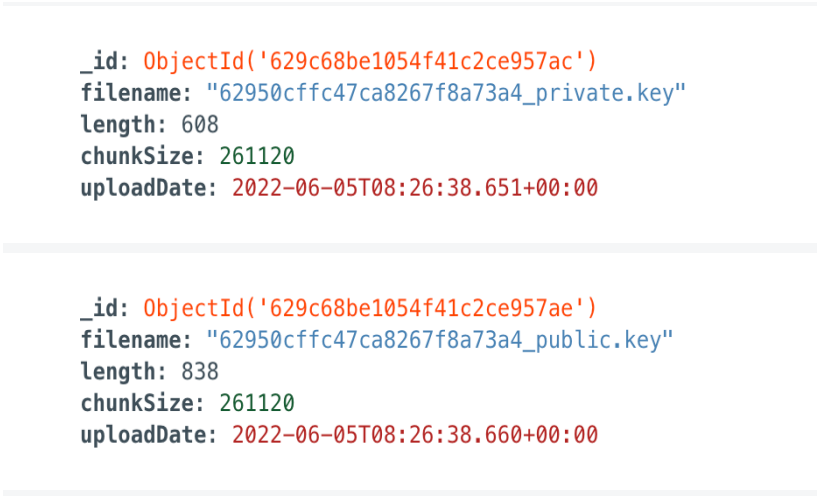

Рисунок 3.14 - Збережені ключі

Підпис здійснюється від обличчя відділу за таким алгоритмом:

1. Після підтвердження здійснюється перевірка умов для підпису;
2. Сервіс шукає приватний, публічний ключ відділу, та сам файл, який хочемо підписати;
3. Завдяки бібліотекам Java ми створюємо підпис за інформацією з приватного ключа та байтам самого файлу;
4. Додаємо дані до метадати файлу.

Також даний сервіс має змогу верифікувати підпис документу. Для цього слід скачати архів з файлом, публічним ключем відділу та з самим підписом. На наступному рисунку 3.15 вигляд скачаного архіву.

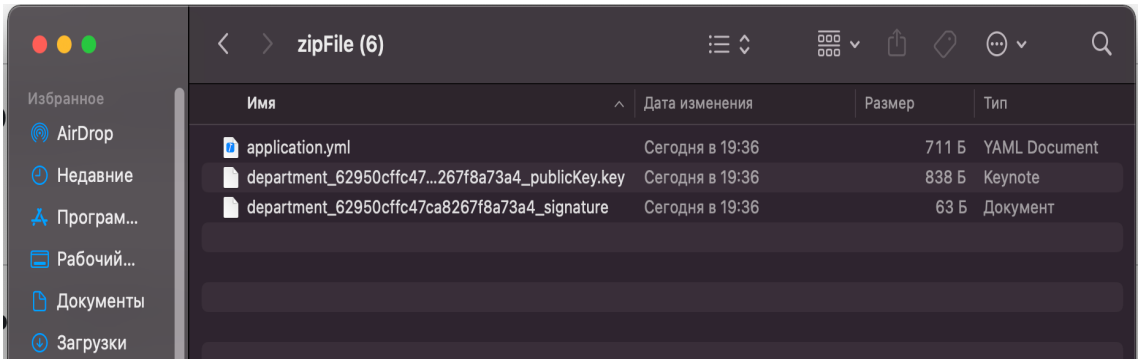

Рисунок 3.15 - Результат скачування підписаного файлу

Верифікація підписаного файлу також здійснюється за допомогою бібліотек Java, ми просто завантажуємо всі три файли та отримаємо результат верифікації.

3.7 Висновок до розділу 3

Було описано сервіси, які потрібно розробити для вирішення поставленої задачі.

Як результат розділу:

1. Побудована архітектура проекту;
2. Побудовані схеми для більш легкої розробки;
3. Розроблені всі потрібні сервіси для виконання дипломної задачі;
4. Описані етапи виконання основних функцій проекту

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Автентифікація користувача

Авторизація виконується за допомогою authority-svc.

Було створено вікно авторизації (рис. 4.1). При успішній автентифікацію, буде перекинуто на сторінку користувача. При помилці вона буде відображена з інформацією. Окрема сторінка не була розроблена для помилок.

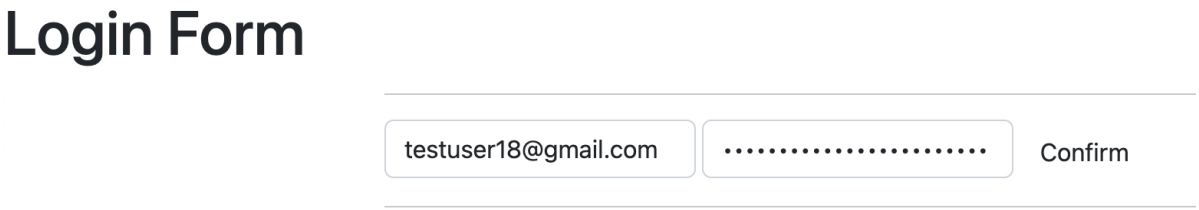

Рисунок 4.1 - Вікно авторизації

Рисунок 4.2 - Вікно профілю після авторизації

Як і було зазначено вище у схемі, після автентифікації сервер зберігає новий кукі файл до хедеру браузеру (рис. 4.3) з ключем для редіс сховища (рис. 4.4) за яким отримуються дані про користувача як з сесії та будемо перевіряти кожний раз, чи авторизований користувач.

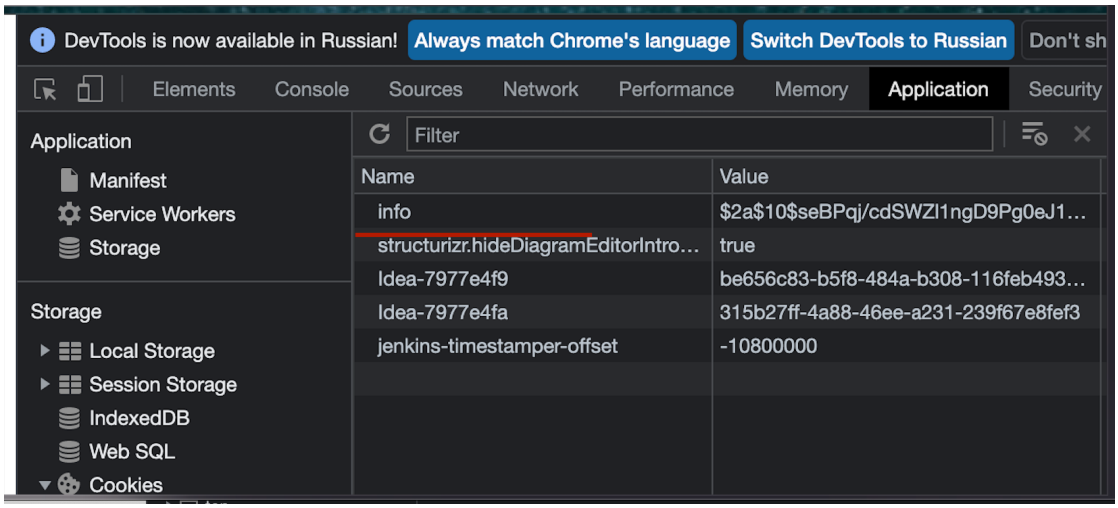

Рисунок 4.3- Cookie файл

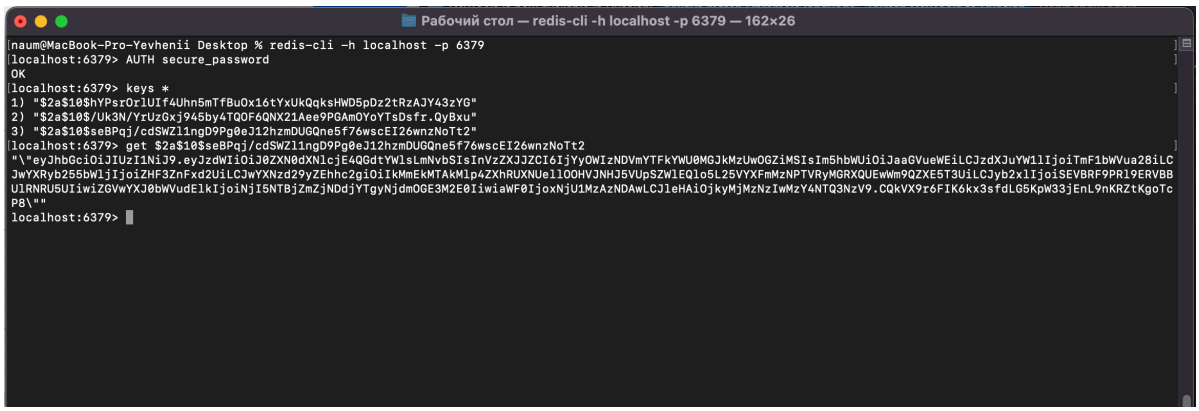

Рисунок 4.2 - Збережений jwt токен у редісі

Офіційний сайт JWT[23] допоможе візуалізувати дані для перевірки, які були збережені до токену (рис. 4.4)

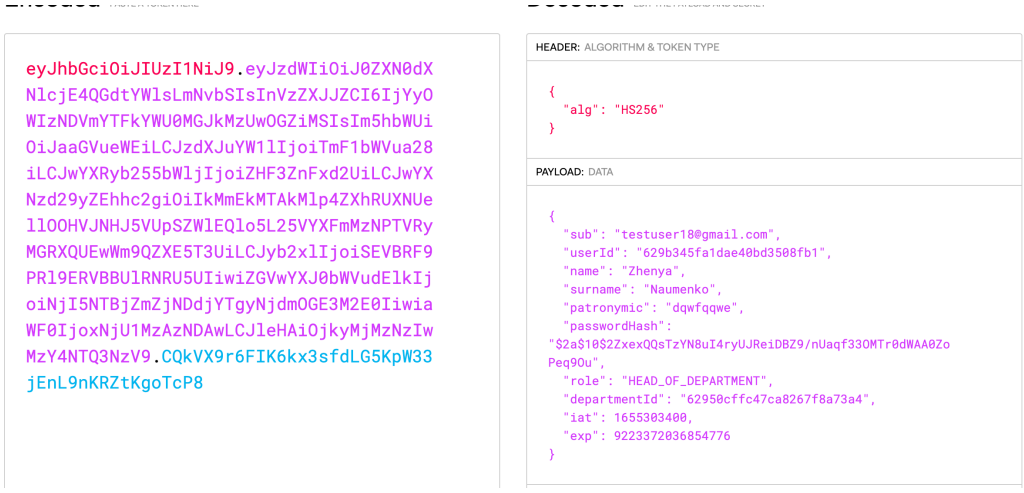

Рисунок 4.4 - Перевірка збереженого токену

Якщо сталася помилка під час авторизації, то нам буде показано цю помилку з додатковою інформацією. Для прикладу виконано авторизацію з даними користувача, якого не існує і як результат отримали помилко, що даного клієнта не знайдено, та отримаємо деталі помилки з введеними даними у форму автентифікації (рис. 4.5).

```
USER not found. Detail: email = 124124, password = fqw
```

Рисунок 4.5 - Помилка при хибній автентифікації

Під час будь-якого запиту Cloud Api Gateway на своїй стороні перевіряє те, щоб клієнт був автентифікован. У протилежному випадку буде помилка і скористатися сервісом не буде нагоди (рис. 4.6).

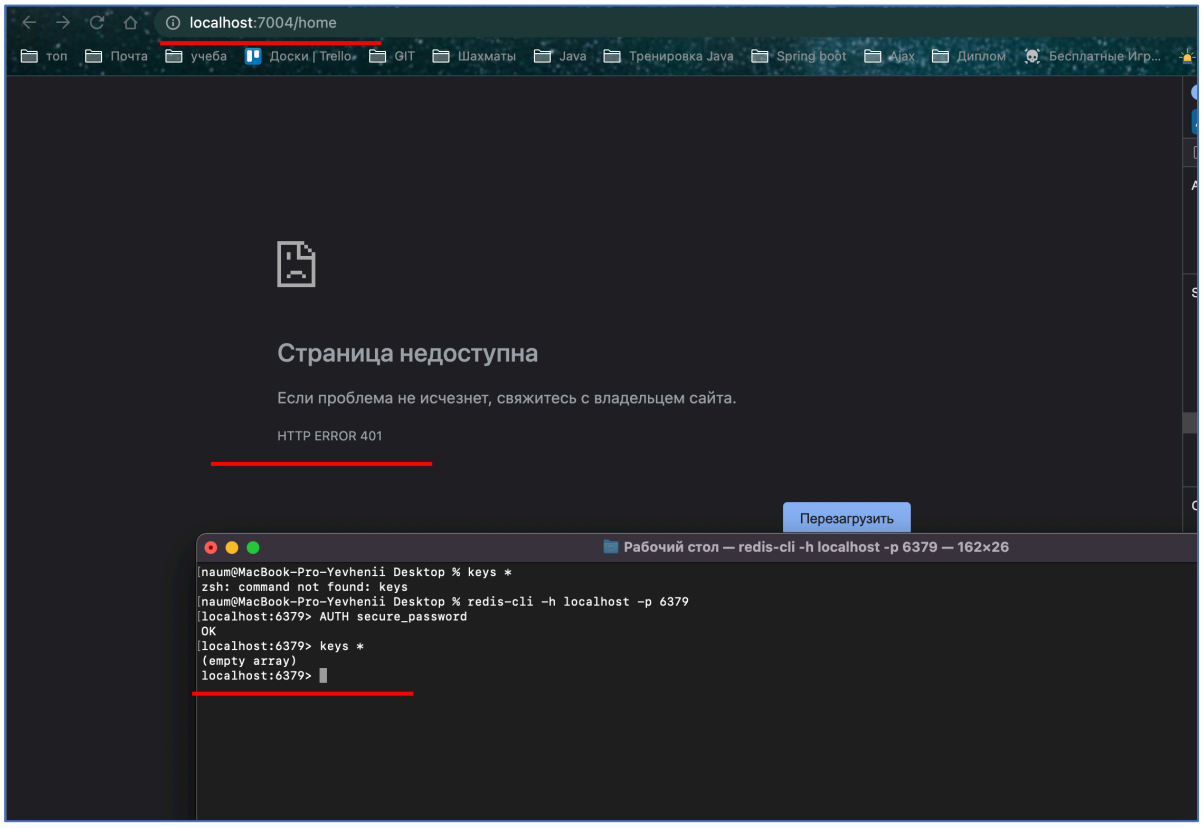

Рисунок 4.6 - Помилка при запиті без автентифікації

4.2 Сервіс користувача та відділу

Створення відділу та реєстрація клієнтів мають практично однакову реалізацію у нашому проекті. Реєстрацію клієнтів зможе виконувати лише хтось з співробітників. Так зроблено, бо потрібно, щоб хтось з компанії вирішував, до якого відділу буде відправлений працівник.

Окреме вікно не було розроблено цих двох запитів, тому як можливість для створення можна використати програму Postman [26]

Якщо ввести ідентифікатор відділу (або адміністрації при створенні відділу), якого не було створено, то отримаємо помилку, що такого не знайдено як на рисунку 4.7.

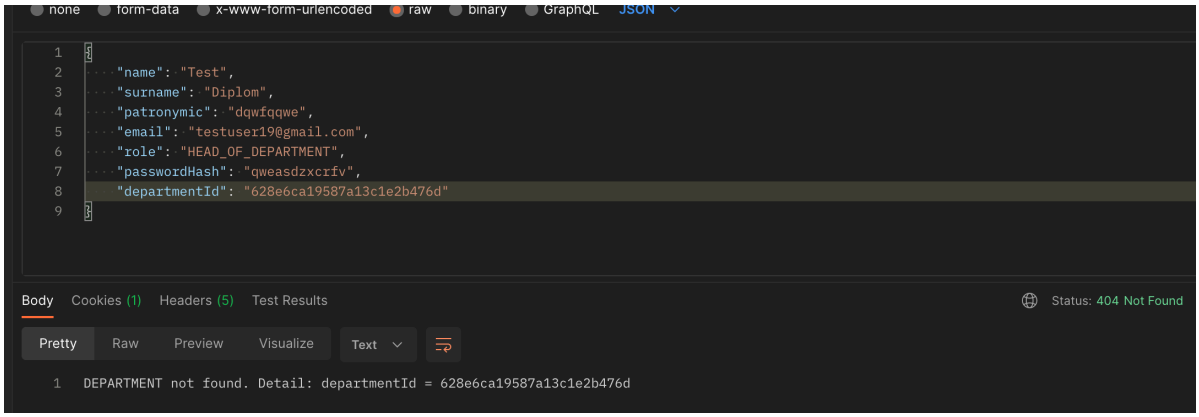

Рисунок 4.7 - Помилка при створенні користувача

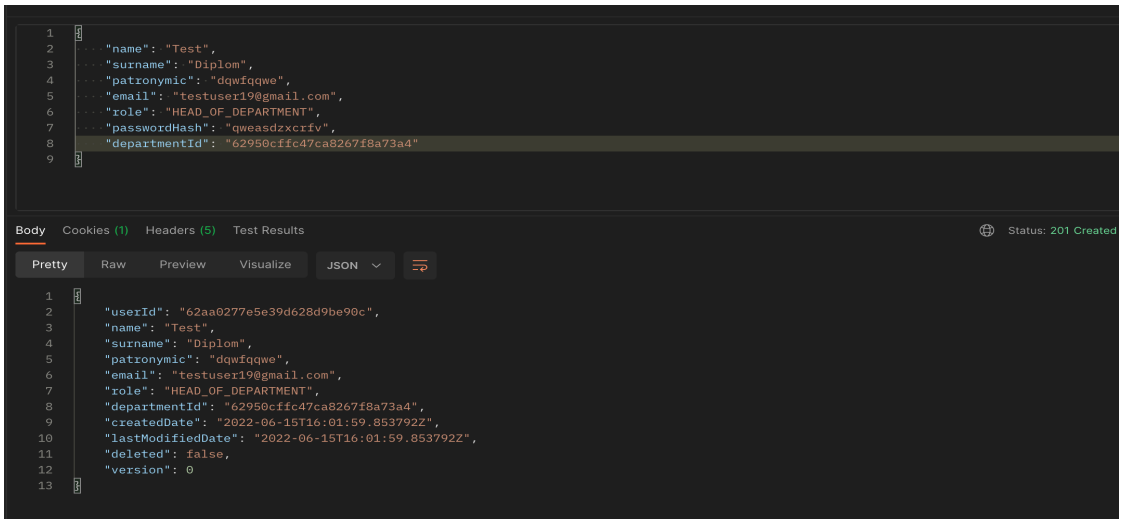

Рисунок 4.8 - Успішне створення користувача

4.3 Сервіс документів

Даний сервіс може виконує всі функції для роботи з файлами. Клієнт може отримати список документів своїх документів (рис. 4.9).

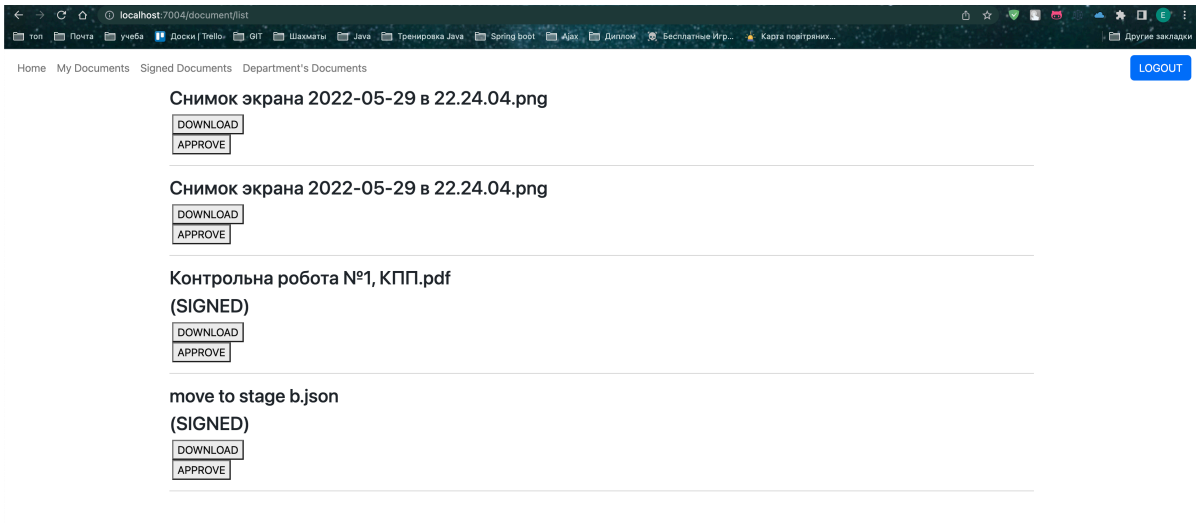

Рисунок 4.9 - Список власних документів

Для перегляду документу є можливість скачати документ. Якщо документ вже підписаний, до буде скачуватися архів с документом, публічним ключем, та самим підписом, які можна буде підтвердити (рис. 4.10).

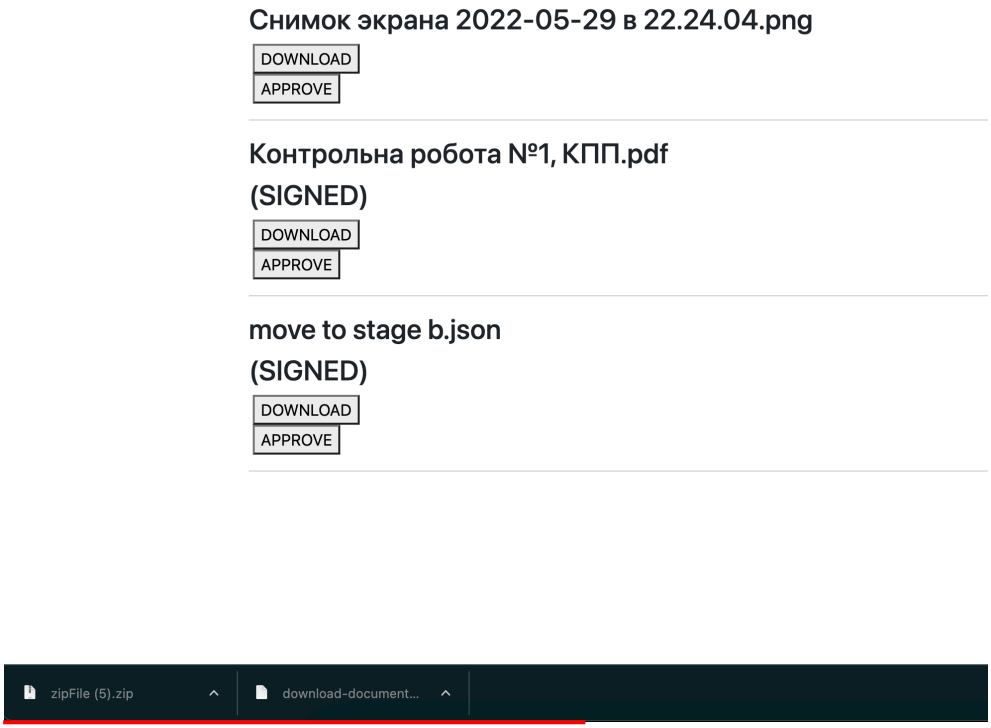

Рисунок 4.10 - Скачування документів

Була реалізована загрузка файлів до свого аккаунту (рис. 4.11 та 4.12).

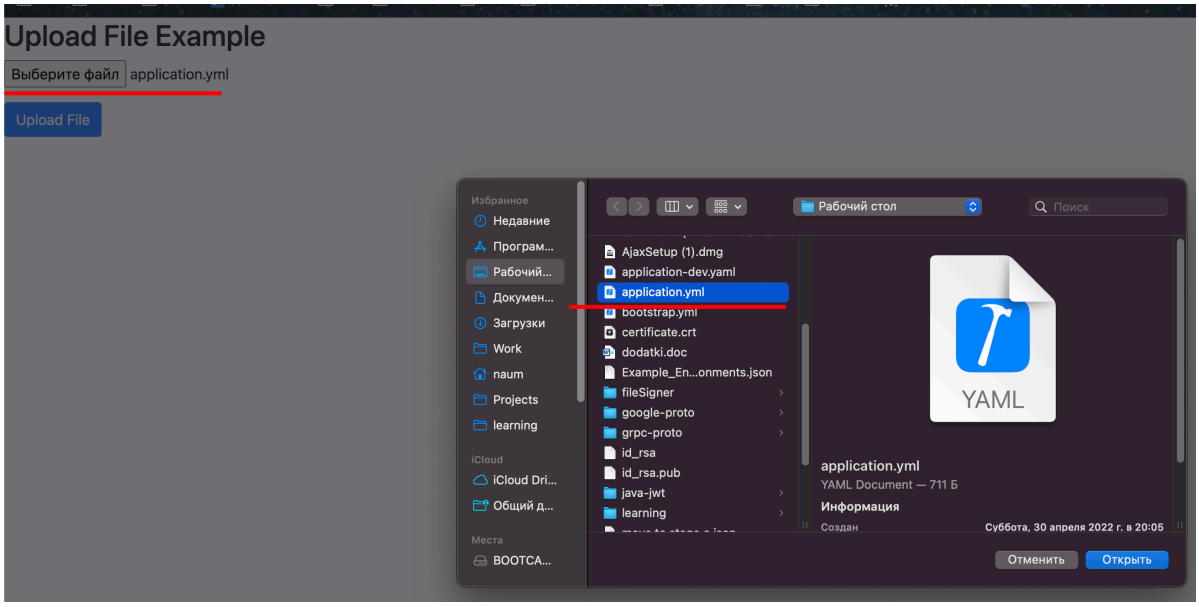

Рисунок 4.11 - Загрузка файлів

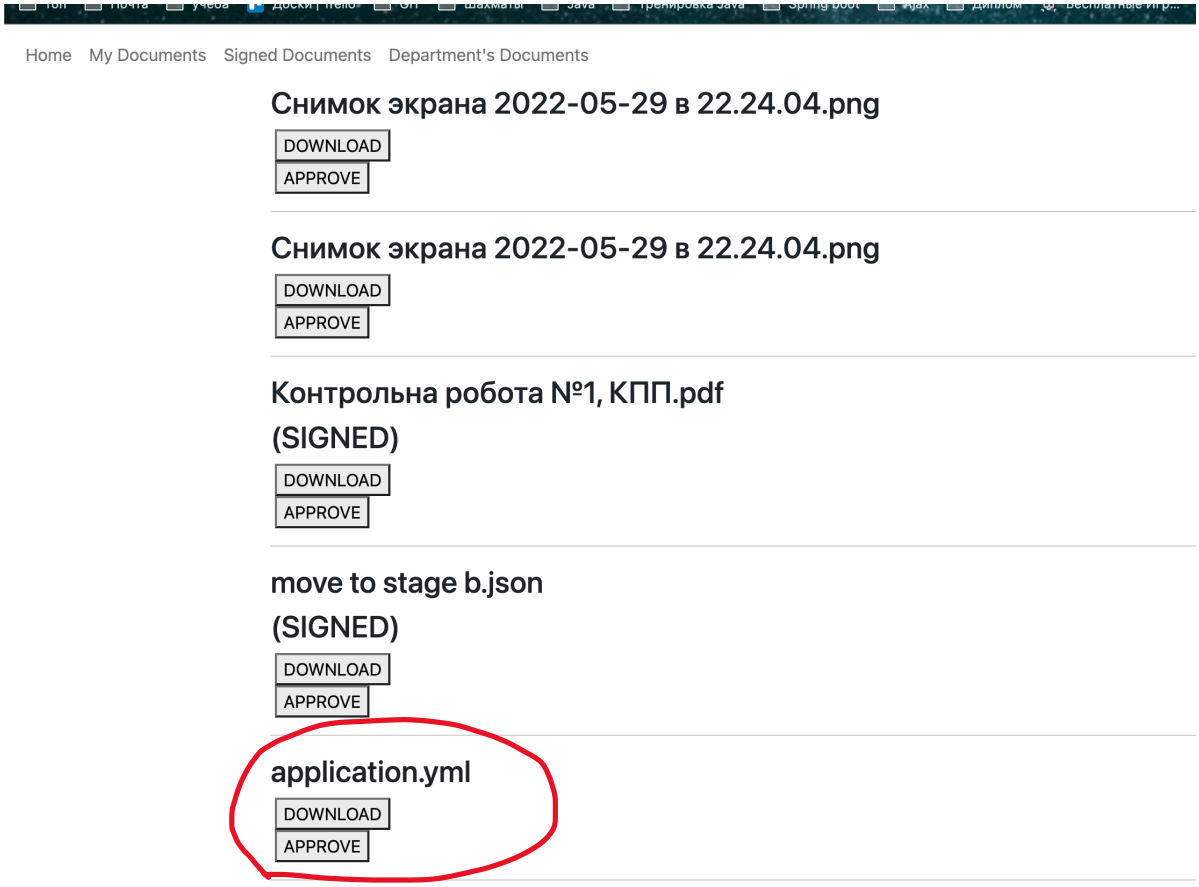

Рисунок 4.12 - Результат загрузки файлу

Одна з особливостей, яка була поставлена в задачі диплому – автопідпис файлу за певних умов. Одна з таких умов – підтвердження від голови відділу. Так як зараз ми знаходимося на акаунті голови відділу, то при підтверджені він одразу підпишеться.

На рисунку 4.13 відображено успішність операції, а на 4.14 зміни на сторінці за файлами (додано підпис «signed»).

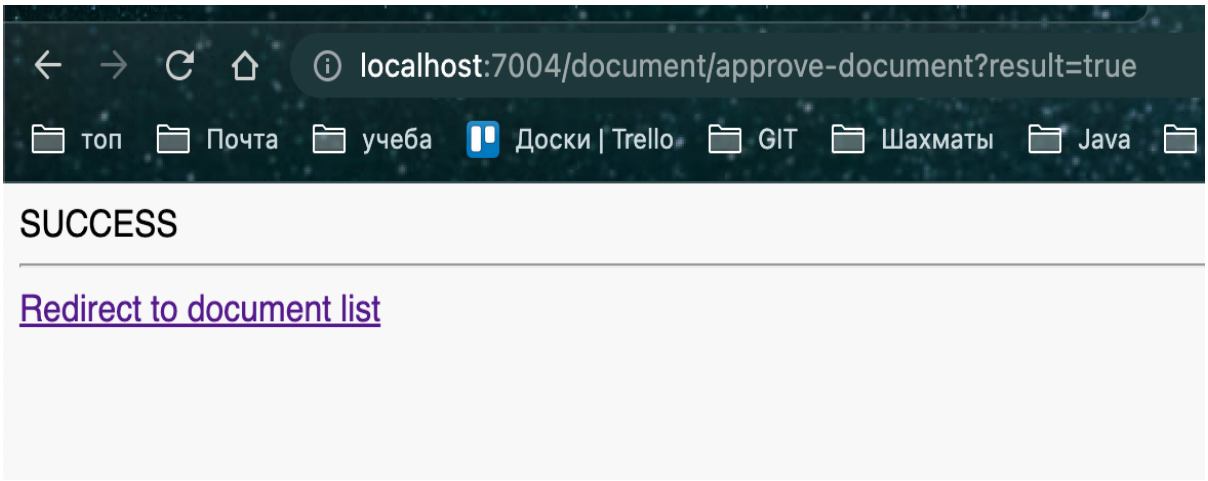

Рисунок 4.13 - Результат підтвердження 1

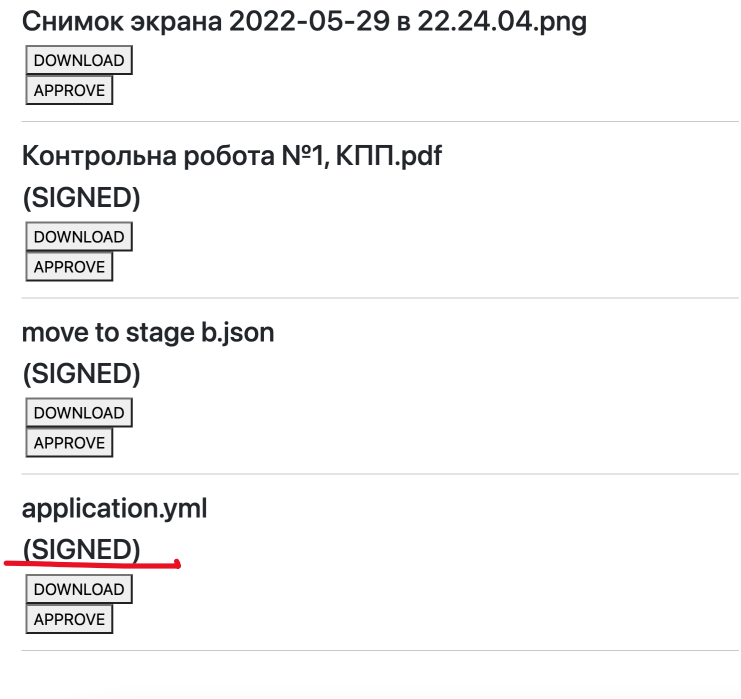

Рисунок 4.14 - Результат підтвердження 2

Як і зазначалося вище, при скачуванні підписаного файлу, скачується архів з файлом, підписом та публічним ключем (рис. 4.15).

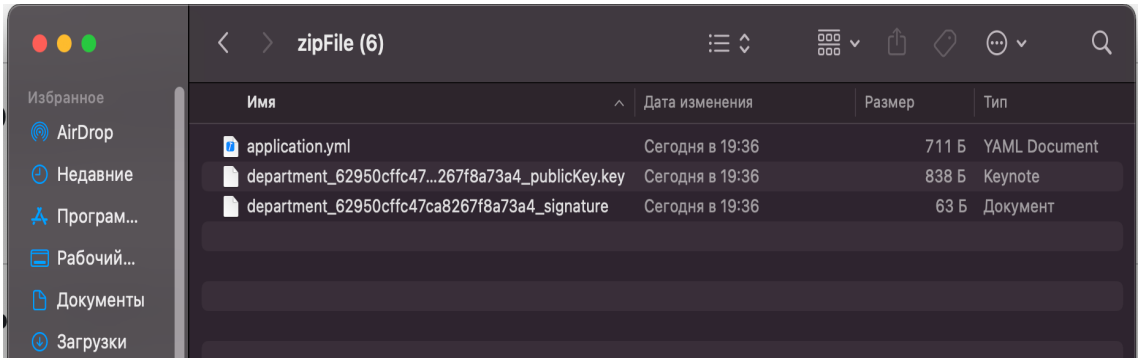

Рисунок 4.15 - Результат підпису 3

Якщо клієнт буде намагатися верифікувати файли, з ключем або сигнатурою, яка є невалідною то отримаю помилку (рис. 4.17), тобто файл підтвердження не пройшло. На рисунку 4.16 ми завантажуємо документ з підписом, а замість публічного ключа файл, файл, який не підходить.

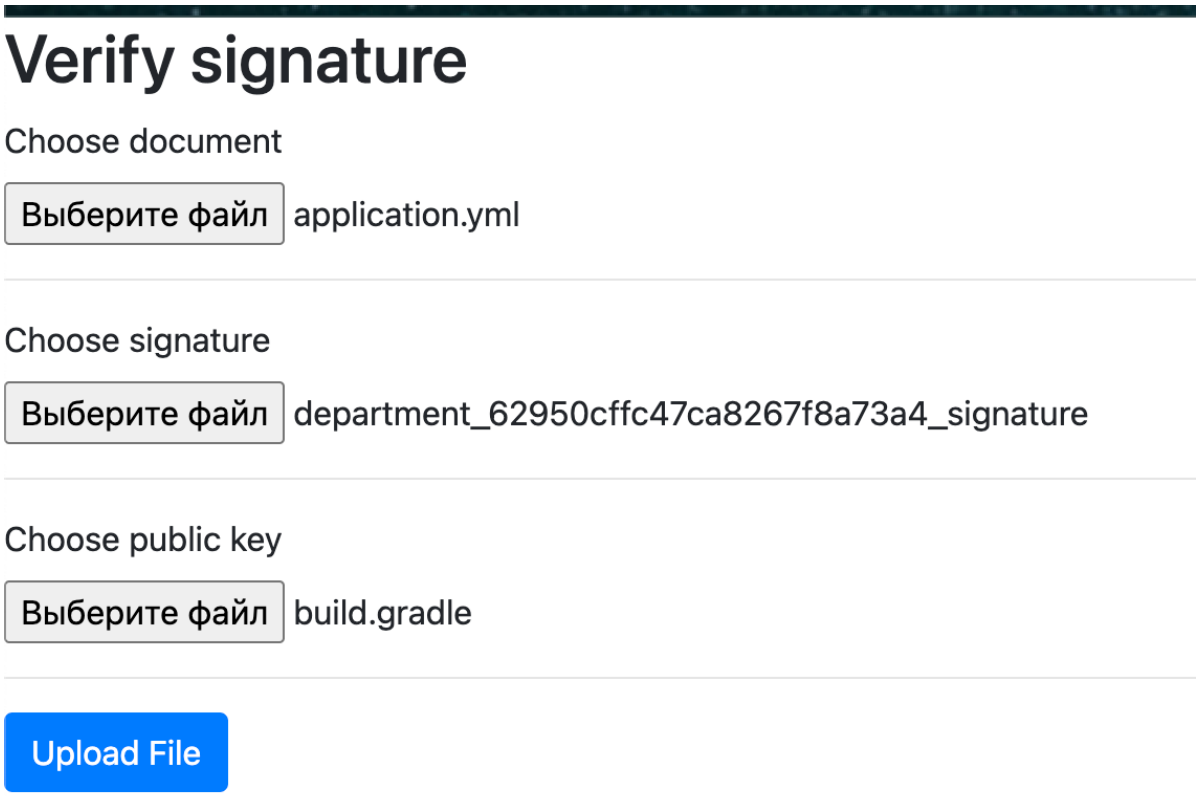

Рисунок 4.16 - Невалідна перевірка

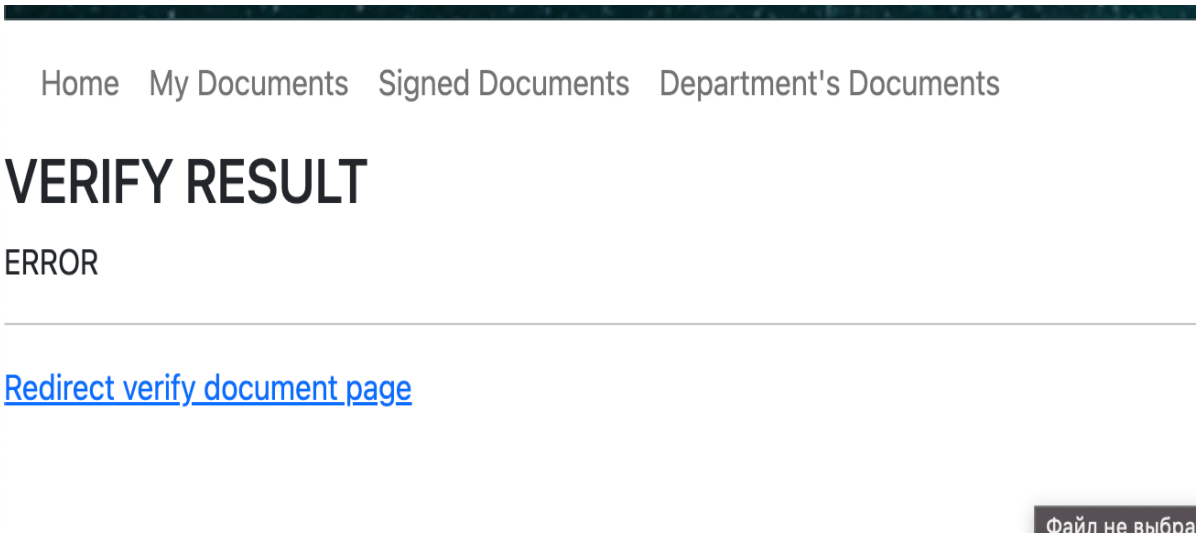

Рисунок 4.17 - Результат поганої верифікації

При загрузці правильних файлів, буде отримано відповідь «успішно» (рис. 4.18).

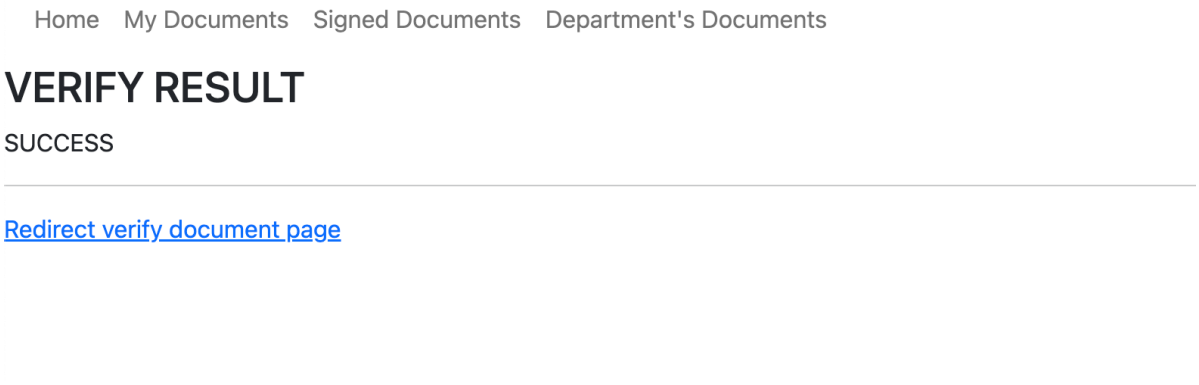

Рисунок 4.18 - Результат верифікації

4.4 Висновки до розділу 4

Проведено розробку основних сервісів. Було покрито основні завдання дипломного проекту. Важкість розробки сервісів найбільше проявляється при самостійній розробці, бо самотужки стежити за кожним з них – трудоємка задача.

Кожний сервіс є захищеним, бо шлюз не дає доступ до кінцевих точок, якщо людина не автентифікована. На кожному сервісі, майже у кожній функції використовується логування, що в майбутньому дасть змогу підключити якийсь сервіс по відстеженню логів всіх сервісів о дному місці.

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде проведена оцінка основних функціональних та вартісних характеристик нашого програмного продукту, розробленого для вирішення задачі автоматизації документообігу на основі мікросервісної архітектури

Цей продукт призначення для використання на ПК під управлінням будь-якої операційної системи, так як це буде веб-додаток.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної стратегії створення програмного продукту, враховуючи при цьому як економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

«Функціонально-вартісний аналіз – це технологія оцінки реальної вартості продукту або послуги незалежно від організаційної структури компанії. Прямі та побічні витрати розподіляються по продуктам та послугам у залежності від потрібних обсягів ресурсів на кожному етапі виробництва. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.»[29]

Мета ФВА полягає у забезпеченні самого оптимального розподілу ресурсів, що виділені на виробництво або надання послуг, на непрямі та прямі витрати

Метод ФВА розпочинається з визначення ряду функцій, необхідних для виготовлення продукту. Спочатку йдуть всі можливі функції, які діляться по двом групам: ті, що можуть впливати на вартість продукту і ті, що не впливають.

Для кожної функції визначається повний обсяг витрат, які додаються за рік та кількість робочих часів. За даними оцінок визначається кількісна характеристика джерел витрат. Після визначення джерел проводиться кінцевий розрахунок всіх витрат на виробництво продукту.

5.1 Постановка задачі техніко-економічного аналізу

«Метод ФВА був застосований для проведення техніко-економічний аналізу розробки системи для виявлення та фільтрування агресивних висловлювань в коментарях в соціальних мережах. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного обробки та фільтрування даних.»[29]

До продукту були визначені наступні технічні вимоги:

1) можливість виконання на персональних комп’ютерах із стандартною конфігурацією за допомогою веб-браузера;
2) висока швидкість обробки даних та відповідь користувачеві у реальному часі;
3) зручність та простота взаємодії;
4) можливість зручного налаштування, масштабування та обслуговування;
5) мінімальні витрати на впровадження програмного продукту.

5.1.1 Обґрунтування функцій програмного продукту

Нульова і головна функція F_0 – розробка програмного продукту, яка вирішує задачу дляп Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

F_1 – вибір мови програмування;

F_2 – вибір бібліотек;

F_3 – вибір середовища розробки.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

1) Kotlin/Java
2) JavaScript

Функція F_2:

1) Spring Boot
2) Natural.js
3) Tensorflow

Функція F_3:

1) Intellij IDEA
2) WebStorm
3) Visual Studio Code

5.1.2 Варіанти реалізації функцій

Варіанти реалізації основних функцій показані на морфологічній карті системи на рисунку 5.1. Ця карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

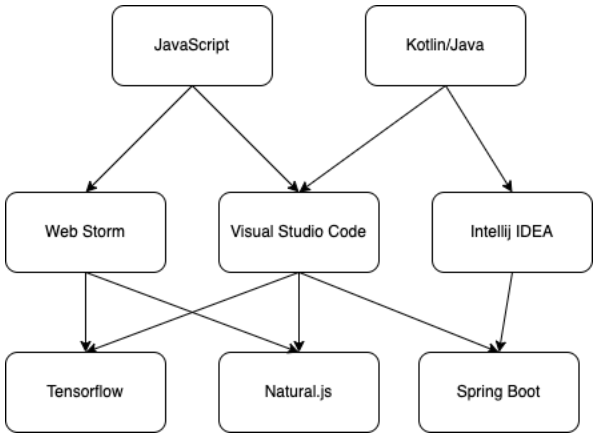

Рисунок 5.1 - Морфологічна карта варіантів реалізації функцій

На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 5-1).

Таблиця 5.1 - Позитивно-негативна матриця варіантів основних функцій

Функція	Варіант реалізації	Переваги	Недоліки
F_1	А	Кросплатформений, займає менше часу на виконання коду	Займає більше часу на написання коду
	Б	Займає менше часу на написання коду, кросплатформений	Займає більше часу для виконання коду
F_2	А	Безкоштовність, добра документація, постійно оновлюється, сильний застосунок для розробки серверної частиги	Потребує велику кількість потужності техніки
	Б	Безкоштовність, гарна документація, підтримка багатьох мов	Відсутня підтримка
F_3	А	Гарна підтримка, швидкодія	Тільки для JVM
	Б	Безкоштовний Підходить як для JS, так і для Python	Необхідність довгих налаштувань для оптимізації роботи
	В	Потребує менше пам’яті, можливо виконувати окремі ділянки коду	Більше підходить для JS, платний

На основі аналізу позитивно-негативної матриці робимо такий висновок, що при розробці продукту деякі варіанти реалізації функцій варто не використовувати, тому, що вони не відповідають поставленим перед програмним продуктом задачам.

Функція F_1:

Оскільки реалізація програми для вирішення поставленої задачі є концептуально складною, необхідно обрати мову, що спрощує написання коду для виконавця, тому варіант А має бути відкинутий.

Функція F_2:

Вибір фреймворку для розробки складної серверної частини краще підходить варіант А.

Функція F_3:

Оскільки, програмний продукт реалізується мовою Kotlin, можемо відкинути варіант всі варіанти окрім А

Таким чином, будемо розглядати такі варіанти реалізації програмного продукту:

– $F_1(A) – F_2(A) – F_3(A)$

5.2 Обґрунтування системи параметрів програмного продукту

5.2.1 Опис параметрів

На основі даних про обов’язкові функції, що повинен зробити програмний продукт, вимог до нього, визначаються основні параметри виробу для розрахунку коефіцієнта технічного рівня.

Щоб характеризувати програмний продукт, будемо використовувати наступні параметри:

– X1 – швидкодія операцій мови програмування в залежності від обраної серверної технології;
– X2 – об’єм пам’яті для збереження даних персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми;
– X3 – час, який витрачається на виконання коду;
– X4 – потенційний об’єм програмного коду, який необхідно створити безпосередньо розробнику.

5.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту як показано у таблиці 5-2.

Таблиця 5.2 - Основні параметри програмного продукту

Опис параметру	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/с	1500	3000	4755
Об’єм пам’яті для збереження даних	X2	Мб	256	128	32
Час виконання коду	X3	мс	2000	600	100
Потенційний об’єм програмного коду	X4	строк коду	5000	3500	2000

За даними таблиці 4 будуються графічні характеристики параметрів (див. рис. 5.2–5.5).

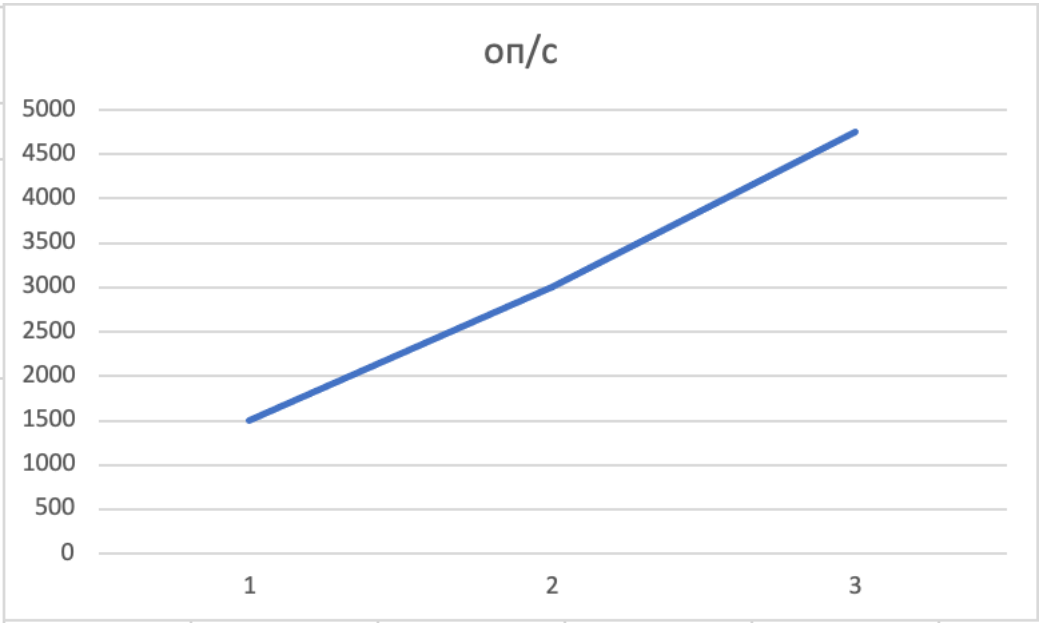

Рисунок 5.2 - Швидкодія мови програмування

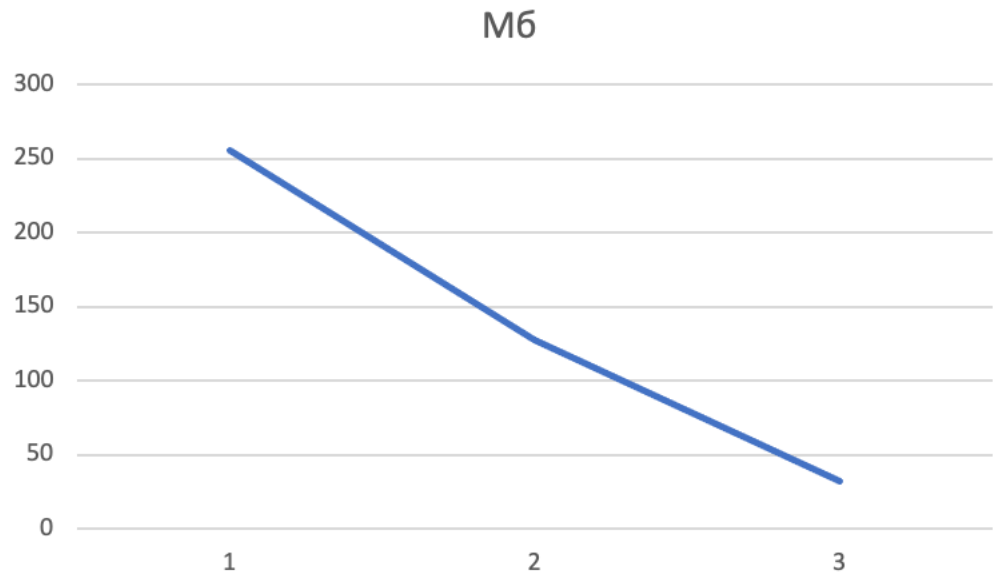

Рисунок 5.3 - Об’єм пам’яті для збереження даних

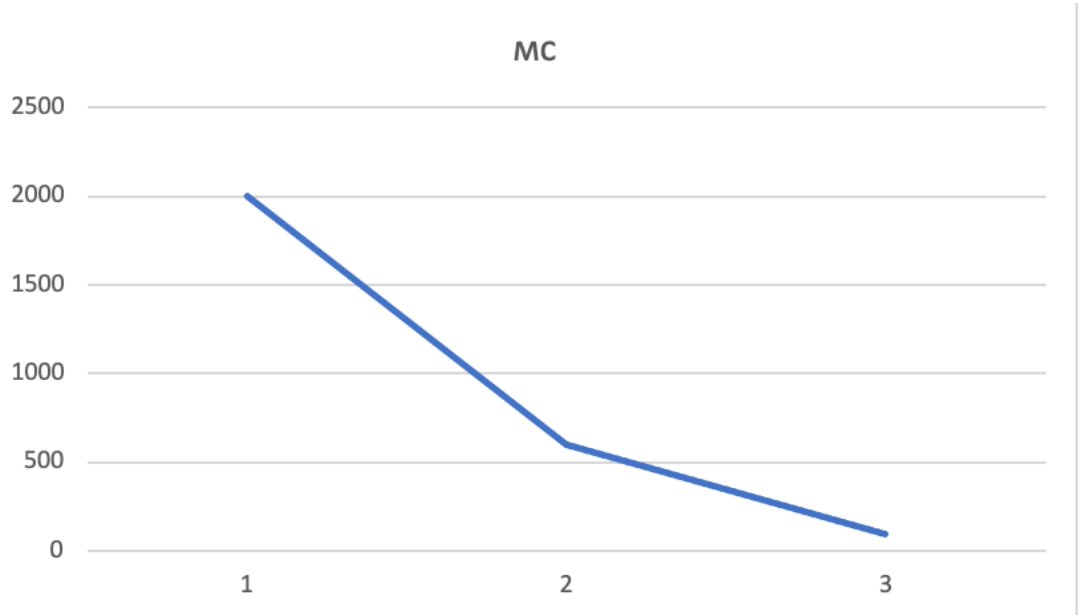

Рисунок 5.4 - Час виконання коду

Рисунок 5.5 - Потенційний об’єм програмного коду

5.2.3 Аналіз експертного оцінювання параметрів

Методом експертної оцінки, визначимо важливість кожного параметру для розробки програмного продукту для автоматизації документообігу.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 - Результати ранжування показників

Параметр	Ранг параметра за оцінкою експерта					Сума рангів	Відхилення, Δ_i	Δ_i^2
	1	2	3	4	5			
X1	1	3	2	5	5	16	0,75	0,5625
X2	2	5	3	4	2	16	0,75	0,5625
X3	3	3	4	5	2	17	1,75	3,0625
X4	5	1	1	2	3	12	-3,25	10,5625
Разом	11	12	10	16	12	61	0	14,75

Для перевірки ступеню достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} * r_{ij} = 61,$$

де r_{ij} – ранг *i*-го параметра, визначений *j*-м експертом;

N – число експертів.

б) середня сума рангів *T*:

$$T = \frac{1}{n} R_i = 15{,}25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{n} * \Delta_i^2 = 14.75.$$

д) коефіцієнт узгодженості (конкордації):

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 14.75}{5^2(4^3 - 4)} > W_k = 0{,}118.$$

Ранжирування можна є достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний - 0,118. Проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4 Числове значення, що визначає ступінь переваги *i*-го параметра над *j*-тим, a_{ij} визначається за формулою:

$$a_{ij} = \{1{,}5 \;\; x_i > x_j;\; 1{,}0 \;\; x_i = x_j;\; 0{,}5 \;\; x_i < x_j\}.$$

Таблиця 5.4 - Результати ранжування параметрів

Параметри	Експерти					Підсумкова оцінка	Числове значення коефіцієнтів переваги
	1	2	3	4	5		
X1,X2	<	<	<	>	>	<	0.5
X1,X3	<	=	<	=	>	<	0.5
X1,X4	<	>	>	>	>	>	1.5
X2,X3	<	>	<	<	=	=	1
X2,X4	<	>	>	>	<	>	1,5
X3,X4	<	>	>	>	<	>	1,5

З отриманих числових оцінок переваги зробимо матрицю $A = \|a_{ij}\|$. Для всіх параметрів розрахунок вагомості K_{B_i} проводиться за формулою:

$$K_{B_i} = \frac{b_i}{\sum_{i=1}^{n} * b_i},$$

«де $b_i = \sum_{j=1}^{N} a_{ij}$ – вагомість i-го параметра за результатами експертної оцінки;

a_{ij} –коефіцієнт переваги i-го на j-тим параметром.»

Відносні оцінки розраховуються декілька разів, поки наступні значення не будуть незначно відрізнятися від попередніх. На другому і наступних кроках відносні оцінки розраховуються за наступною формулою:

$$K_{B_i} = \frac{b_i'}{\sum_{i=1}^{n} * b_i'},$$

де $b_i' = \sum_{j=1}^{N} * a_{ij} b_j$.

Таблица 5.5 - Розрахунок вагомості параметрів

i	j				Перша ітерація		Друга ітерація	
	X1	X2	X3	X4	B_i	K_{B_i}	B_i^1	$K_{B_i}^1$
X1	1,0	0,5	0,5	1,5	5,5	0,344	21,25	0,36
X2	1,5	1,0	1	1,5	3,5	0,219	12,25	0,208
X3	1,5	1	1,0	0,5	2,5	0,156	9,25	0,157
X4	0,5	1,5	0,5	1,0	4,5	0,281	16,25	0,275
Разом					16,0	1,0	59,0	1,0

5.3 Аналіз рівня якості варіантів реалізації функцій

Рівень якості кожного варіанту основних функцій визначається окремо.

Абсолютні значення параметре X1(швидкодія мови програмування) та параметруX2 (об’єм пам’яті для збереження даних) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3 (час розрахунків) обрано не найгіршим, тобто це значення відповідає варіанту б) 600 або в) 100 мс.

Абсолютне значення параметра X4 (кількість строк коду) обрано не найгіршим б) 3500 або в) 2000 строк коду.

Коефіцієнт технічного рівня якості для кожного варіанта реалізації розраховується за формулою:

$$K_{TP} = \sum_{i=1}^{n} * K_{B_i} B_i,$$

де n – кількість параметрів;

K_{B_i} – коефіцієнт вагомості і–го параметра;

B_i – оцінка i–го параметра в балах.

Розрахунок показників рівня якості представлено відповідно в таблиці

Таблиця 5.6 - Розрахунок показників якості

Основні функції	Варіант реалізації	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	3000	10	0,36	3,6
F_2	А	128	5	0,208	1,04
F_3	А	100	5	0,157	0,785
	Б	600	10	0,157	1,75
F_4	А	2000	10	0,275	2,75
	Б	3500	5	0,275	1,375

Визначаємо рівень якості кожного з варіантів:

- F_1(А) – F_2(А) – F_3(А) F_4(А) = 3,6 + 1,04 + 0,785 + 2,75 = 8,175
- F_1(А) – F_2(А) – F_3(Б) F_4(Б) = 3,6 + 1,04 + 1,76 + 1,375 = 7,775
- F_1(А) – F_2(А) – F_3(Б) F_4(А) = 3,6 + 1,04 + 1,75 + 2,75 = 9,14
- F_1(А) – F_2(Б) – F_3(А) F_4(Б) = 3,6 + 1,04 + 0,785 + 1,375 = 6,8

Отже, найкращим є 3 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки програмного продукту спочатку проведемо розрахунок трудомісткості.

Для реалізації завдання 1 використовує інформацію у вигляді даних, а завдання 2 використовує стандартні методи. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється за такою формулою:

$$Т_О = Т_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}$$

«де $Т_Р$ – трудомісткість розробки програмного продукту;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.»

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р = 53$ людино-днів. Поправочний коефіцієнт, який враховує вид вхідної інформації для першого завдання: $К_П = 1{,}7$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації рівний $К_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ} = 0{,}8$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 53 \cdot 1.7 \cdot 0.8 = 72{,}08 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для другого завдання, в якому використовується алгоритм третьої групи складності зі ступенем новизни Б, тобто $Т_Р = 30$ людино-днів, $К_П = 0{,}9$, $К_{СК} = 1$, $К_{СТ} = 0{,}8$:

$$Т_2 = 30 \cdot 0.9 \cdot 0.8 = 21{,}6 \text{ людино-днів.}$$

Оскільки загальна трудомісткість усіх варіантів реалізації збігаються, їх можна об’єднати в одну групу.

Загальна трудомісткість складає:

$$Т_О = (72{,}08 + 21{,}6) \cdot 8 = 739{,}44 \text{ людино-годин;}$$

В розробці беруть участь програміст з окладом 45000 грн., один аналітик в області даних з окладом 30000 грн, та один інженер даних з окладом 20000 грн. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{M}{T_m \cdot t} \text{ грн.,}$$

«де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин в день.»

$$С_Ч = \frac{45000 + 30000 + 20000}{3 \cdot 20 \cdot 8} = 197{,}92 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_Ч \cdot T_i \cdot К_Д,$$

«де $С_Ч$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.»

Зарплата розробників за варіантами становить:

$$С_{ЗП} = 197{,}92 \cdot 739 \cdot 1{,}2 = 175\,515{,}456 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{СВ} = С_{ЗП} \cdot 0{,}22 = 175\,515{,}456 \cdot 0{,}22 = 38\,613{,}4 \text{ грн.}$$

Тепер визначимо витрати на одну машино-годину. Оскільки одна ЕОМ обслуговується одним інженером з окладом 10000 грн. та коефіцієнтом зайнятості $К_З = 0{,}2$ то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 10\,000 \cdot 0{,}2 = 24\,000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 45\,000 \cdot (1 + 0{,}2) = 54\,800 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{СВ} = С_{ЗП} \cdot 0{,}22 = 54\,800 \cdot 0{,}22 = 12\,056 \text{грн.}$$

Амортизаційні відрахування розраховуємо за формулою при амортизації 25% та вартості ЕОМ – 30 000 грн.:

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1{,}15 \cdot 0{,}25 \cdot 20000 = 8\,625 \text{ грн.}$$

«де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо за формулою:»

$$С_Р = К_{ТМ} \cdot К_Р \cdot Ц_{ПР} = 1{,}15 \cdot 0{,}05 \cdot 30000 = 1\,725 \text{ грн.}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t \cdot К_В, Т_{ЕФ} = (365 - 104 - 12 - 16) \cdot 8 \cdot 0{,}9 = 1\,667{,}6 \text{ год.}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В, Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_C \cdot К_З \cdot Ц_{ЕЛ} = 1\,677{,}6 \cdot 0{,}65 \cdot 0{,}2 \cdot 3{,}7969 = 828{,}06 \text{ грн.}$$

де N_C – середньо-споживча потужність приладу;

$К_З$ – коефіцієнтом зайнятості приладу;

$Ц_{ЕЛ}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0{,}67 = 30\,000 \cdot 0{,}67 = 20100 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть складати:

$$С_{ЕК} = С_{ЗП} + С_{СВ} + С_А + С_Р + С_{ЕЛ} + С_Н, С_{ЕК}$$
$$= 54\,800 + 12\,056 + 8\,625 + 1\,725 + 828{,}06 + 20\,100$$
$$= 98\,134{,}06 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{МГ} = \frac{С_{ЕК}}{Т_{ЕФ}} = \frac{98\,134{,}06}{1\,677{,}6} = 58{,}49 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати машинного часу складають:

$$С_М = С_{МГ} \cdot Т = 58{,}49 \cdot 833{,}12 = 48\,729{,}2 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0{,}67 = 175\,515{,}456 \cdot 0{,}67 = 117\,595{,}356 \text{ грн.}$$

Отже, вартість розробки програмного продукту за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{СВ} + С_М + С_Н, С_{ПП}$$
$$= 175\,515{,}456 + 12\,056 + 48\,729{,}2 + 117\,595{,}356$$
$$= 353\,896 \text{ грн.}$$

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР} = \frac{К_К}{С_{ПП}} = \frac{7{,}16}{353\,896} = 2{,}56 \cdot 10^{-5}$$

5.5 Висновки до розділу 5

В результаті виконання економічного розділу були систематизовані і закріплені теоретичні знання в галузі економіки та організації виробництва використанням їх для техніко-економічного обґрунтування розробки методом функціонально-вартісного аналізу.

На основі даних про основні функції, які повинен реалізувати програмний продукт, були визначені чотири найбільш перспективні варіанти реалізації продукту.

ВИСНОВКИ

В даній дипломній роботі було розглянуто проблему автоматизації документообігу на основі мікросервісної архітектури.

Проведений аналіз переваг мікросервісної архітектури. Розглянуто вже існуючі програмні продукти, які виконують цю задачу, та досліджено найпопулярніші з них. Виявлено, що кожна програма використовується для певного кола користувачів.

Були досліджені методи розробки мікросервісної архітектури та використані найбільш оптимальні рішення для швидкої розробки та полегшеного стеження за роботою програми.

Розроблено веб-додаток згідно до дипломного завдання. Були реалізовані такі сервіси: сервіс автентифікації, сервіс користувачу, сервіс відділу та сервіс документу. Використано єдиний шлюз для доступу користувача до цих сервісів через нього. Програма дає змогу завантажувати файл до застосунку або скачувати собі на пристрій для огляду, підтверджувати файли, створювати підпис до файлу та забезпечує захищеність й доступ до додатку завдяки автентифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microservices vs Monolithic. URL: https://ncube.com/blog/microservices-vs-monolithic-which-architecture-suits-best-for-your-project. (дата звернення: 11.06.2022).
2. BAS документообіг КОРП. URL: https://inteltech.com.ua/ru/bas-dokumentooborot-korp. (дата звернення: 11.06.2022).
3. JSolution – модуль документообігу. URL: https://jsolutions.ua/avtomatizatciya-dokumentooborota. (дата звернення: 11.06.2022).
4. FossDoc. URL: https://fossdoc.com/ru/elektronniy-dokumentooborot.
5. HotDocs. UR: https://www.hotdocs.com/. (дата звернення: 11.06.2022).
6. BAS документообіг КОРП. Короткий опис продукту. URL: https://www.bas-soft.eu/upload/content/BAS_Dokumentoobig/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D1%96%D0%B0%D0%BB%D0%B8/BAS%20%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D1%96%D0%B3%20%D0%9A%D0%9E%D0%A0%D0%9F.pdf. (дата звернення: 11.06.2022).
7. BAS документообіг КОРП. Ціни на програмні забеспечення. URL: https://inteltech.com.ua/ru/ceny-na-programmy-bas. (дата звернення: 12.06.2022).
8. Спілка автоматизаторів бізнесу. URL: https://unionba.com.ua/
9. OpenOffice. URL: https://www.openoffice.org/ru/. (дата звернення: 12.06.2022).
10. Libre Office. URL: https://ru.libreoffice.org/. (дата звернення: 12.06.2022).
11. Oracle Servers. URL: https://www.oracle.com/cis/servers/. (дата звернення: 12.06.2022).

12. Microsoft SQL server. URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019. (дата звернення: 12.06.2022).
13. FossLook. URL: https://fosslook.ru/. (дата звернення: 12.06.2022).
14. Spring Boot. URL: https://spring.io/. (дата звернення: 12.06.2022).
15. Gradle Manual. URL: https://docs.gradle.org/current/userguide/userguide.html. (дата звернення: 12.06.2022).
16. Spring Cloud Api Gateway. URL: https://spring.io/projects/spring-cloud-gateway. (дата звернення: 12.06.2022).
17. Spring Cloud Config Server. URL: https://cloud.spring.io/spring-cloud-config/reference/html/. (дата звернення: 12.06.2022).
18. Netflix Eureka Server. URL: https://cloud.spring.io/spring-cloud-netflix/reference/html/. (дата звернення: 12.06.2022).
19. JSON. URL: https://developers.squarespace.com/what-is-json. (дата звернення: 12.06.2022).
20. MongoDb. URL: https://www.mongodb.com/. (дата звернення: 13.06.2022).
21. Mongo Compass. URL: https://www.mongodb.com/docs/compass/current/. (дата звернення: 13.06.2022).
22. Redis. URL: https://redis.io/. (дата звернення: 13.06.2022).
23. JWT. URL: https://jwt.io/.. (дата звернення: 13.06.2022).
24. Kotlin. URL: https://kotlinlang.org/. (дата звернення: 13.06.2022).
25. IntellijIDEA. URL: https://www.jetbrains.com/ru-ru/idea/. (дата звернення: 13.06.2022).
26. Postman. URL: https://www.postman.com/. (дата звернення: 13.06.2022).
27. Proto. URL: https://developers.google.com/protocol-buffers. (дата звернення: 14.06.2022).

28. OpenFeign. URL: https://spring.io/projects/spring-cloud-openfeign. (дата звернення: 14.06.2022).
29. Функціонально-вартісний аналіз. URL: http://dspace.wunu.edu.ua/jspui/bitstream/316497/462/1/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE-%D0%B2%D0%B0%D1%80%D1%82%D1%96%D1%81%D0%BD%D0%B8%D0%B9%20%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7.pdf. (дата звернення: 14.06.2022).

ДОДАТОК А

GATEWAY-SVC

```
package learning.diplom.gatewaysvc.filter

import
learning.diplom.gatewaysvc.exception.JwtTokenMalformedException
import learning.diplom.gatewaysvc.exception.JwtTokenMissingException
import learning.diplom.gatewaysvc.util.JwtUtil
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.cloud.gateway.filter.GatewayFilter
import org.springframework.cloud.gateway.filter.GatewayFilterChain
import org.springframework.data.redis.core.RedisTemplate
import org.springframework.http.HttpStatus
import org.springframework.http.server.reactive.ServerHttpRequest
import org.springframework.stereotype.Component
import org.springframework.web.server.ServerWebExchange
import org.springframework.web.servlet.function.ServerResponse
import reactor.core.publisher.Mono
import java.net.URI
import java.util.function.Predicate

@Component
class JwtAuthenticationFilter(
    private val jwtUtil: JwtUtil,
    private val redisTemplate: RedisTemplate<String, Any>,
    @Value("\${jwt.info}")
    private val info: String? = null
) : GatewayFilter {

    override fun filter(exchange: ServerWebExchange, chain:
GatewayFilterChain): Mono<Void> {
        LOG.info("Started filter, exchange = {}", exchange)

        val apiEndpoints = listOf("/login")
        val isApiSecured = Predicate { r: ServerHttpRequest ->
            apiEndpoints.stream()
                .noneMatch { uri: String? ->
r.uri.path.contains(uri!!) }
        }

        val request = exchange.request
        val redisKey = exchange.request.cookies[info]?.get(0)?.value
?: "empty"

        if (isApiSecured.test(request)) {
            val token = redisTemplate.opsForValue().get(redisKey)

            if (redisKey == "empty" || token == null) {
                LOG.error("Unauthorized!")
                val response = exchange.response
                response.statusCode = HttpStatus.UNAUTHORIZED
                return response.setComplete()
            }
            try {
                jwtUtil.validateToken(token.toString())
            } catch (e: JwtTokenMalformedException) {
                LOG.error("JwtTokenMalformedException: ", e)
                val response = exchange.response
```

```
                response.statusCode = HttpStatus.BAD_REQUEST
                return response.setComplete()
            } catch (e: JwtTokenMissingException) {
                LOG.error("JwtTokenMissingException: ", e)
                val response = exchange.response
                response.statusCode = HttpStatus.BAD_REQUEST
                return response.setComplete()
            }
            val claims = jwtUtil.getClaims(token.toString())!!
            exchange.request.mutate()
                .headers {
                    it["email"] = claims["email"].toString()
                    it["role"] = claims["role"].toString()
                    it["name"] = claims["name"].toString()
                    it["surname"] = claims["surname"].toString()
                    it["patronymic"] =
claims["patronymic"].toString()
                    it["departmentId"] =
claims["departmentId"].toString()
                }.build()
        }
        return chain.filter(exchange)
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(JwtAuthenticationFilter::class.java)
    }
}

package learning.diplom.gatewaysvc.util

import io.jsonwebtoken.*
import
learning.diplom.gatewaysvc.exception.JwtTokenMalformedException
import learning.diplom.gatewaysvc.exception.JwtTokenMissingException
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.stereotype.Component
import java.lang.Exception
import java.lang.IllegalArgumentException

@Component
class JwtUtil(
    @Value("\${jwt.secret}")
    private val jwtSecret: String? = null
) {

    fun getClaims(token: String?): Claims? {
        LOG.info("Get claims: {}", token)
        try {
            return
Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token).body
        } catch (e: Exception) {
            LOG.error("${e.message}" + " => " + e)
        }
        return null
    }

    fun validateToken(token: String?) {
        try {
            Jwts.parser()
```

```
                .setSigningKey(jwtSecret)
                .parseClaimsJws(token)
        } catch (ex: SignatureException) {
            throw JwtTokenMalformedException("Invalid JWT signature")
        } catch (ex: MalformedJwtException) {
            throw JwtTokenMalformedException("Invalid JWT token")
        } catch (ex: ExpiredJwtException) {
            throw JwtTokenMalformedException("Expired JWT token")
        } catch (ex: UnsupportedJwtException) {
            throw JwtTokenMalformedException("Unsupported JWT token")
        } catch (ex: IllegalArgumentException) {
            throw JwtTokenMissingException("JWT claims string is
empty.")
        }
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(JwtUtil::class.java)
    }
}

package learning.diplom.gatewaysvc.config

import learning.diplom.gatewaysvc.filter.JwtAuthenticationFilter
import org.springframework.cloud.gateway.filter.GatewayFilter
import org.springframework.cloud.gateway.route.RouteLocator
import
org.springframework.cloud.gateway.route.builder.GatewayFilterSpec
import org.springframework.cloud.gateway.route.builder.PredicateSpec
import
org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder
import org.springframework.cloud.netflix.hystrix.EnableHystrix
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.http.HttpStatus

@EnableHystrix
@Configuration
class GatewayConfig(
    private val filter: JwtAuthenticationFilter
) {
    @Bean
    fun routes(builder: RouteLocatorBuilder): RouteLocator? {
        return builder.routes()
            .route("user-svc-dev") { r: PredicateSpec ->
                r.path("/user/**")
                    .filters { f: GatewayFilterSpec ->
f.filter(filter) }
                    .uri("lb://user-svc-dev")
            }
            .route("home-page") { r: PredicateSpec ->
                r.path("/home/**")
                    .filters { f: GatewayFilterSpec ->
f.filter(filter) }
                    .uri("lb://user-svc-dev")
            }
            .route("document-svc-dev") { r: PredicateSpec ->
                r.path("/document/**")
                    .filters { f: GatewayFilterSpec ->
f.filter(filter) }
                    .uri("lb://document-svc-dev")
```

```
            }
            .route("document-svc-dev") { r: PredicateSpec ->
                r.path("/sign/**")
                    .filters { f: GatewayFilterSpec ->
f.filter(filter) }
                    .uri("lb://document-svc-dev")
            }
            .route("authority-svc-dev") { r: PredicateSpec ->
                r.path("/auth/**")
                    .filters { f: GatewayFilterSpec ->
f.filter(filter) }
                    .uri("lb://authority-svc-dev")
            }
            .build()
    }
}

package learning.diplom.gatewaysvc.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import
org.springframework.data.redis.connection.RedisConnectionFactory
import org.springframework.data.redis.core.RedisTemplate
import
org.springframework.data.redis.serializer.GenericJackson2JsonRedisSer
ializer
import
org.springframework.data.redis.serializer.StringRedisSerializer

@Configuration
class RedisConfig {
    @Bean
    fun redisTemplate(connectionFactory: RedisConnectionFactory):
RedisTemplate<String, Any> {
        return RedisTemplate<String, Any>().apply {
            setConnectionFactory(connectionFactory)
            val stringRedisSerializer = StringRedisSerializer()
            val jsonSerializer = GenericJackson2JsonRedisSerializer()
            keySerializer = stringRedisSerializer
            valueSerializer = jsonSerializer
            hashKeySerializer = stringRedisSerializer
            hashValueSerializer = jsonSerializer
            setEnableTransactionSupport(true)
        }
    }
}

package learning.diplom.gatewaysvc

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import org.springframework.boot.web.servlet.ServletComponentScan
import
org.springframework.cloud.client.discovery.EnableDiscoveryClient

@SpringBootApplication
@EnableDiscoveryClient
class GatewaySvcApplication

fun main(args: Array<String>) {
```

```
    runApplication<GatewaySvcApplication>(*args)
}

<!DOCTYPE html>
<html lang="en"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>HOME PAGE</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
          crossorigin="anonymous"></script>

  <!--    <link th:href="@{/static/style.css}" rel="stylesheet" />-->
</head>
<body>

<div class="container">
  <!--  <div th:replace="blocks/header :: header"></div>-->
  <h1>HomePage</h1>
  <nav class="nav-bar">

    <a th:href="@{auth/login}">Login</a>
    <!--    <a th:href="@{auth/login}">Login</a>-->
  </nav>
  <hr>
</div>

<!--<footer class="footer" th:insert="blocks/footer ::
footer"></footer>-->

</body>
</html>

<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<header th:fragment="header">
  <!-- Navbar -->
  <nav class="navbar navbar-expand-lg navbar-light bg-white">
    <div class="container-fluid">
      <button
        class="navbar-toggler"
        type="button"
        data-mdb-toggle="collapse"
        data-mdb-target="#navbarExample01"
        aria-controls="navbarExample01"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <i class="fas fa-bars"></i>
      </button>
      <div class="collapse navbar-collapse" id="navbarExample01">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
```

```
          <li class="nav-item active">
            <a class="nav-link" aria-current="page"
th:href="@{/}">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Documents</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Pricing</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">About</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
  <!-- Navbar -->
</header>
</html>
```

ДОДАТОК Б

CLOUD-CONFIG-SVC

```
package learning.diplom.cloudconfig

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import
org.springframework.cloud.client.discovery.EnableDiscoveryClient
import org.springframework.cloud.config.server.EnableConfigServer

@SpringBootApplication
@EnableConfigServer
@EnableDiscoveryClient
class CloudConfigApplication

fun main(args: Array<String>) {
   runApplication<CloudConfigApplication>(*args)
}

server:
  port: 7006

gateway:
  host: localhost
  port: 7004

jwt:
  secret:
BvPHGM8C0ia4uOuxxqPD5DTbWC9F9TWvPStp3pb7ARo0oK2mJ3pd3YG4lxA9i8bj6OTba
dwezxgeEByY
  validity: 180000

app:
  service:
    name: Administration Service

spring:
  application:
    name: administration-svc-dev
  #  profiles:
  #    active: test
  data:
    mongodb:
      uri: mongodb://local-mongo-usr:local-mongo-
pwd@localhost:27017/administration-svc?authSource=admin
      database: administration-svc

  redis:
    host: localhost
    port: 6379
    password: secure_password

server:
  port: 7008

gateway:
  host: localhost
  port: 7004
```

```
jwt:
  secret:
BvPHGM8C0ia4uOuxxqPD5DTbWC9F9TWvPStp3pb7ARo0oK2mJ3pd3YG4lxA9i8bj6OTba
dwezxgeEByY
  validity: 180000
  info: info

app:
  service:
    name: Authority Service

spring:
  application:
    name: authority-svc-dev
  #  profiles:
  #    active: test
  data:
    mongodb:
      uri: mongodb://local-mongo-usr:local-mongo-
pwd@localhost:27017/user-svc?authSource=admin
      database: user-svc

  redis:
    host: localhost
    port: 6379
    password: secure_password
    database: 0

server:
  port: 7005

gateway:
  host: localhost
  port: 7004

app:
  service:
    name: Department Service

spring:
  application:
    name: department-svc-dev
  #  profiles:
  #    active: test
  data:
    mongodb:
      uri: mongodb://local-mongo-usr:local-mongo-
pwd@localhost:27017/department-svc?authSource=admin
      database: department-svc

  redis:
    host: localhost
    port: 6379
    password: secure_password

#values configs
administration-svc-feign-client: administration-svc-dev

server:
  port: 7007
```

```
gateway:
  host: localhost
  port: 7004

jwt:
  secret:
BvPHGM8C0ia4uOuxxqPD5DTbWC9F9TWvPStp3pb7ARo0oK2mJ3pd3YG4lxA9i8bj6OTba
dwezxgeEByY
  validity: 180000
  info: info

app:
  service:
    name: Document Service

spring:
  application:
    name: document-svc-dev
  #  profiles:
  #    active: test
  data:
    mongodb:
      uri: mongodb://local-mongo-usr:local-mongo-
pwd@localhost:27017/document-svc?authSource=admin
      database: document-svc

  redis:
    host: localhost
    port: 6379
    password: secure_password

server:
  port: 7004

app:
  service:
    name: Gateway Service DEV

spring:
  session:
    store-type: redis
  main:
    web-application-type: reactive
  jackson:
    default-property-inclusion: NON_NULL
  application:
    name: gateway-svc-dev
  redis:
    host: localhost
    port: 6379
    password: secure_password

  data:
    mongodb:
      uri: mongodb://local-mongo-usr:local-mongo-
pwd@localhost:27017/user-svc?authSource=admin
      database: user-svc

jwt:
  secret:
BvPHGM8C0ia4uOuxxqPD5DTbWC9F9TWvPStp3pb7ARo0oK2mJ3pd3YG4lxA9i8bj6OTba
dwezxgeEByY
```

```
  validity: 180000
  info: info

  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
          lower-case-service-id: true

      #ROUTES
      routes:
        - id: authority-svc-dev
          uri: lb://authority-svc-dev
          predicates:
            - Path=/auth/**

server:
  port: 7003

app:
  service:
    name: User Service

jwt:
  secret:
BvPHGM8C0ia4uOuxxqPD5DTbWC9F9TWvPStp3pb7ARo0oK2mJ3pd3YG4lxA9i8bj6OTba
dwezxgeEByY
  validity: 180000
  info: info

spring:
  application:
    name: user-svc-dev
#  profiles:
#    active: test
  data:
    mongodb:
      uri: mongodb://local-mongo-usr:local-mongo-
pwd@localhost:27017/user-svc?authSource=admin
      database: user-svc

  redis:
    host: localhost
    port: 6379
    password: secure_password

#values configs
department-svc-feign-client: department-svc-dev
document-svc-feign-client: document-svc-dev

# Application-wide configuration
server:
  port: 8000

spring:
  application:
    name: cloud-config
  profiles:
    active: native, test, dev
  cloud:
    config:
```

```
      server:
        git:
          uri: git@github.com:jeniknaum5/cloud-config.git
#          host-key-algorithm: ssh-rsa
#          host-key-algorithm: ssh-ed25519
#          host-key-algorithm: ecdsa-sha2-nistp521
#        native:
#          search-locations:
#            - classpath:/config
  security:
    user:
      name: configUser
      password: configPassword
      roles: [SYSTEM]

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8002/eureka

#        "--requirepass secure_password"

services:
  redis:
    image: redis:4.0.10-alpine
    command: >
        "--requirepass secure_password"
        "--bind 0.0.0.0"
    restart: always
    ports:
        - "6379:6379"
    healthcheck:
        test: [ "CMD", "redis-cli"]
        interval: 10s
        timeout: 3s
        retries: 5

  mongo:
    container_name: mongodb
    image: mongo:4.2.8
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: local-mongo-usr
      MONGO_INITDB_ROOT_PASSWORD: local-mongo-pwd
      MONGO_INITDB_DATABASE: user-svc
    volumes:
      - ./mongo:/docker-entrypoint-initdb.d
      - ./mongo-data:/data/db
    ports:
      - 27017:27017
    healthcheck:
      test: echo 'rs.initiate().ok || rs.status().ok' | mongo -u
$${MONGO_INITDB_ROOT_USERNAME} -p $${MONGO_INITDB_ROOT_PASSWORD} --
quiet
      interval: 10s
      timeout: 10s
      retries: 5
    stop_grace_period: 30s
    command: [--auth]

#  keycloak:
#      image: quay.io/keycloak/keycloak:15.0.1
```

```
#       ports:
#           - "8080:8080"
#       environment:
#           - KEYCLOAK_USER=admin
#           - KEYCLOAK_PASSWORD=admin
```

ДОДАТОК В

EUREKA-SERVER

```
package learning.diplom.eurekaserver

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import
org.springframework.cloud.netflix.eureka.server.EnableEurekaServer

@SpringBootApplication
@EnableEurekaServer
class EurekaServerApplication

fun main(args: Array<String>) {
    runApplication<EurekaServerApplication>(*args)
}

server:
  port: 8002

sping:
  application:
    name: eureka-svc

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
  instance:
    hostname: localhost
spring:
  redis:
    host: localhost
    port: 6379
    password: secure_password
```

ДОДАТОК Г

AUTHORITY-SVC

```
package learning.diplom.authority.svc.config

import learning.diplom.model.error.lib.handler.RestExceptionHandler
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder

@Configuration
class BeanConfig {

    @Bean
    fun bCryptPasswordEncoder(): BCryptPasswordEncoder {
        return BCryptPasswordEncoder()
    }

    @Bean
    fun restExceptionHandler(): RestExceptionHandler {
        return RestExceptionHandler()
    }
}

package learning.diplom.authority.svc.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import
org.springframework.data.redis.connection.RedisConnectionFactory
import org.springframework.data.redis.core.RedisTemplate
import
org.springframework.data.redis.serializer.GenericJackson2JsonRedisSer
ializer
import
org.springframework.data.redis.serializer.StringRedisSerializer

@Configuration
class RedisConfig {
    @Bean
    fun redisTemplate(connectionFactory: RedisConnectionFactory):
RedisTemplate<String, Any> {
        return RedisTemplate<String, Any>().apply {
            setConnectionFactory(connectionFactory)
            val stringRedisSerializer = StringRedisSerializer()
            val jsonSerializer = GenericJackson2JsonRedisSerializer()
            keySerializer = stringRedisSerializer
            valueSerializer = jsonSerializer
            hashKeySerializer = stringRedisSerializer
            hashValueSerializer = jsonSerializer
            setEnableTransactionSupport(true)
        }
    }
}

package learning.diplom.authority.svc.controller

import learning.diplom.authority.svc.model.LoginRequest
```

```
import learning.diplom.authority.svc.util.JwtUtil
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.data.redis.core.RedisTemplate
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder
import org.springframework.stereotype.Controller
import org.springframework.ui.Model
import org.springframework.web.bind.annotation.*
import java.net.URI
import javax.servlet.http.Cookie
import javax.servlet.http.HttpServletResponse

@Controller
@RequestMapping("/auth")
class AuthController(
    private val jwtUtil: JwtUtil,
    @Value("\${jwt.info}")
    private val info: String? = null,
    @Value("\${gateway.host}")
    private val host: String? = null,
    @Value("\${gateway.port}")
    private val port: String? = null,
    private val redisTemplate: RedisTemplate<String, Any>
) {

    @PostMapping("/login")
    fun login(
        @ModelAttribute @RequestBody request: LoginRequest,
        httpResponse: HttpServletResponse
    ): ResponseEntity<String> {
        val dbUser =
jwtUtil.getUserByEmailWithPasswordChecking(request.email!!,
request.password!!)
        LOG.info("Generate token for request = {}", request)
        val jwtToken = jwtUtil.generateToken(request.email,
request.password, dbUser)
        val encryptedKey =
jwtUtil.encryptInfoAndPutItToRedis(request.email, request.password,
jwtToken!!)
        LOG.info("Generating cookie")
        val cookie = Cookie(info, encryptedKey)
        cookie.path = "/"
        httpResponse.addCookie(cookie)

        return ResponseEntity
            .status(HttpStatus.FOUND)
            .location(URI("$HTTP_PREFIX$host:$port/home"))
            .build()
    }

    @PostMapping("/logout")
    fun logout(
        httpResponse: HttpServletResponse,
        @CookieValue("info") redisKey: String? = null,
        model: Model
    ): String {

        if (redisKey.isNullOrBlank()) {
            throw InternalError("NotFound info")
```

```
        }
        redisTemplate.delete(redisKey)
        model.addAttribute("loginRequest", LoginRequest())
        return "login-form"

    }

    @GetMapping("/login")
    fun getLoginForm(model: Model): String? {
        model.addAttribute("loginRequest", LoginRequest())
        return "login-form"
    }

    @GetMapping("/logout")
    fun getLogoutForm(): String {
        return "logout-form"
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(AuthController::class.java)
        private const val HTTP_PREFIX = "http://"
    }
}

package learning.diplom.authority.svc.controller

import org.slf4j.LoggerFactory
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RestController
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@RestController
@RequestMapping("/auth")
class RestControllers {

    @GetMapping("/login/createNewSession")
    fun test(request: HttpServletRequest, response:
HttpServletResponse): String {
        var sessionObj = request.getSession(false)
        //check session exist or not
        //if not available create new session
        if (sessionObj == null) {
            LOG.info("Session not available, creating new session.")
            sessionObj = request.getSession(true)
        }
        val activeSessions =
            if (sessionObj!!.getAttribute("activeSessions") != null)
sessionObj.getAttribute("activeSessions")
                .toString() else "0"
        return "Session is available now with total active sessions :
$activeSessions"
    }

    @GetMapping("/login/destroy-active-session")
    fun removeSession(request: HttpServletRequest, response:
HttpServletResponse?): String? {
        val sessionObj = request.getSession(false)
        if (sessionObj != null) {
            sessionObj.invalidate()
```

```
            return "Session destroyed, now there are no active
sessions."
        }
        return "Session not available to destroy."
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(RestControllers::class.java)
    }

}

package learning.diplom.authority.svc.exception

import javax.naming.AuthenticationException

class JwtTokenMalformedException(msg: String?) :
AuthenticationException(msg) {
    companion object {
        private const val serialVersionUID = 1L
    }
}

package learning.diplom.authority.svc.exception

import javax.naming.AuthenticationException

class JwtTokenMissingException(msg: String?) :
AuthenticationException(msg) {
    companion object {
        private const val serialVersionUID = 1L
    }
}

package learning.diplom.authority.svc.model

data class LoginRequest(
    val email: String? = null,
    val password: String? = null
)

package learning.diplom.authority.svc.model

enum class Role {
    HEAD_OF_ADMINISTRATION,
    DEPUTY_HEAD_OF_ADMINISTRATION,
    HEAD_OF_DEPARTMENT,
    DEPUTY_HEAD_OF_DEPARTMENT,
    EMPLOYEE
}

package learning.diplom.authority.svc.model

import com.fasterxml.jackson.annotation.JsonIgnore
import org.jetbrains.annotations.NotNull
import org.springframework.data.annotation.*
import java.time.Instant
import javax.validation.constraints.Email
import javax.validation.constraints.NotBlank
import javax.validation.constraints.NotEmpty
import javax.validation.constraints.Size
```

```
data class User(
    @Id
    @get:ReadOnlyProperty
    var userId: String? = null,
    @NotBlank
    @get:Size(min = 3, max = 16)
    var name: String? = null,
    @NotBlank
    @get:Size(min = 4, max = 20)
    var surname: String? = null,
    @NotBlank
    @get:Size(min = 4, max = 20)
    var patronymic: String? = null,
    @NotNull
    @get:Email
    @get:Size(max = 100)
    var email: String? = null,
    @get:NotNull
    var role: Role? = null,
    @NotNull
    @get:JsonIgnore
    var passwordHash: String? = null,
    @NotNull
    var departmentId: String? = null,
    @CreatedDate
    var createdDate: Instant? = null,
    @LastModifiedDate
    var lastModifiedDate: Instant? = null,
    var deleted: Boolean? = false,
    @Version
    var version: Long? = null,
) {
    enum class Role {
        HEAD_OF_ADMINISTRATION,
        DEPUTY_HEAD_OF_ADMINISTRATION,
        HEAD_OF_DEPARTMENT,
        DEPUTY_HEAD_OF_DEPARTMENT,
        EMPLOYEE
    }
}

package learning.diplom.authority.svc.repository

import learning.diplom.authority.svc.model.User
import org.springframework.data.mongodb.repository.MongoRepository

interface UserRepository: MongoRepository<User, String> {
    fun findByEmailAndPasswordHash(email: String, passwordHash:
String): User
    fun existsByEmailAndPasswordHash(email: String, passwordHash:
String): Boolean
    fun findByEmail(email: String): User
    fun existsByEmail(email: String): Boolean
}

package learning.diplom.authority.svc.util

import io.jsonwebtoken.*
import
learning.diplom.authority.svc.exception.JwtTokenMalformedException
import
```

```
learning.diplom.authority.svc.exception.JwtTokenMissingException
import learning.diplom.authority.svc.model.User
import learning.diplom.authority.svc.repository.UserRepository
import learning.diplom.model.error.lib.ServerEntity
import
learning.diplom.model.error.lib.exception.rest.EntityNotFoundRestExce
ption
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.data.redis.core.RedisTemplate
import org.springframework.http.ResponseEntity
import org.springframework.http.server.reactive.ServerHttpRequest
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder
import org.springframework.stereotype.Component
import java.security.SignatureException
import java.util.*
import javax.servlet.http.Cookie
import javax.servlet.http.HttpServletResponse

@Component
class JwtUtil(
    @Value("\${jwt.secret}")
    private val jwtSecret: String? = null,
    @Value("\${jwt.validity}")
    private val tokenValidity: Long = 0,
    private val userRepository: UserRepository,
    private val redisTemplate: RedisTemplate<String, Any>,
    private val bCryptPasswordEncoder: BCryptPasswordEncoder,
    @Value("\${jwt.info}")
    private val info: String? = null
) {

    fun getClaims(token: String?): Claims? {
        try {
            return
Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token).body
        } catch (e: Exception) {
            println(e.message + " => " + e)
        }
        return null
    }

    fun getUserByEmailAndPassword(email: String, password: String):
User {
        return if (userRepository.existsByEmailAndPasswordHash(email,
password)) {
            userRepository.findByEmailAndPasswordHash(email,
password)
        } else {
            throw EntityNotFoundRestException(ServerEntity.USER,
"email = $email, password = $password")
        }
    }

    fun getUserByEmailWithPasswordChecking(email: String, password:
String): User {
        return if (userRepository.existsByEmail(email)) {
            val user = userRepository.findByEmail(email)
            if(bCryptPasswordEncoder.matches(password,
user.passwordHash)) {
                throw EntityNotFoundRestException(ServerEntity.USER,
```

```
"email = $email, password = $password")
            }
            user
        } else {
            throw EntityNotFoundRestException(ServerEntity.USER,
"email = $email, password = $password")
        }
    }
    fun generateToken(email: String?, password: String?, dbUser:
User): String? {

        val claims: Claims = Jwts.claims()
            .setSubject(dbUser.email)
        with(dbUser) {
            claims["userId"] = userId
            claims["name"] = name
            claims["surname"] = surname
            claims["patronymic"] = patronymic
            claims["passwordHash"] = passwordHash
            claims["role"] = role
            claims["departmentId"] = departmentId
        }

        val nowMillis = System.currentTimeMillis()
//        val expMillis = nowMillis + tokenValidity
//        val exp = Date(expMillis)
        val exp = Date(Long.MAX_VALUE)
        return Jwts.builder()
            .setClaims(claims)
            .setIssuedAt(Date(nowMillis)).setExpiration(exp)
            .signWith(SignatureAlgorithm.HS256, jwtSecret).compact()
    }

    fun encryptInfoAndPutItToRedis(email: String, password: String,
jwtToken: String): String {
        LOG.info("Put token to redis")
        val redisKey = "$email:$password"
        val encryptedKey = bCryptPasswordEncoder.encode(redisKey)
        redisTemplate.opsForValue().set(encryptedKey, jwtToken)
        return encryptedKey
    }


    fun validateToken(token: String?) {
        try {

Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token)
        } catch (ex: SignatureException) {
            throw JwtTokenMalformedException("Invalid JWT signature")
        } catch (ex: MalformedJwtException) {
            throw JwtTokenMalformedException("Invalid JWT token")
        } catch (ex: ExpiredJwtException) {
            throw JwtTokenMalformedException("Expired JWT token")
        } catch (ex: UnsupportedJwtException) {
            throw JwtTokenMalformedException("Unsupported JWT token")
        } catch (ex: IllegalArgumentException) {
            throw JwtTokenMissingException("JWT claims string is
empty.")
        }
    }

    fun logout(httpRequest: ServerHttpRequest, httpResponse:
```

```
HttpServletResponse): ResponseEntity<String> {
        val redisKey = httpRequest.cookies[info]?.get(0)?.value ?:
"empty"
        if (redisKey == "empty") {
            return ResponseEntity.notFound().build()
        }
        redisTemplate.opsForValue().getAndDelete(redisKey)
        httpResponse.addCookie(Cookie(redisKey, null))
        return ResponseEntity.ok().build()
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(JwtUtil::class.java)
        private const val KEY = "profile"
    }
}

package learning.diplom.authority.svc

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import org.springframework.boot.web.servlet.ServletComponentScan
import
org.springframework.cloud.client.discovery.EnableDiscoveryClient

@SpringBootApplication
@EnableDiscoveryClient
@ServletComponentScan
class AuthoritySvcApplication

fun main(args: Array<String>) {
    runApplication<AuthoritySvcApplication>(*args)
}

<!doctype html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Login Form</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
          crossorigin="anonymous"></script>
</head>
<body>
<!--<header header th:insert="blocks/header :: header"></header>-->
<h1>Login Form</h1>

<div class="container">
```

```
  <form action="/auth/login" method="post"
th:object="${loginRequest}">
    <hr>
    <!--        <input type="number" name="id" placeholder="ID"
class="form-controller" required>-->
    <!--        <input type="text" name="id" th:field="${user.id}"
placeholder="id" class="form-control" required>-->
    <label>
      <input type="text" name="email"
th:field="${loginRequest.email}" placeholder="Email" class="form-
control"
             required>
    </label>
    <label>
      <input type="password" name="password"
th:field="${loginRequest.password}" placeholder="Password"
class="form-control"
             required>
    </label>

    <button class="btn btn-submit">Confirm</button>
    <hr>
  </form>
</div>

<!--<footer class="footer" th:insert="blocks/footer ::
footer"></footer>-->
</body>
</html>

<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
          crossorigin="anonymous"></script>
  <title>Logout Form</title>
</head>
<body>
<div th:replace="blocks/header :: header"></div>

<div class="container">
  <form action="/auth/logout" method="post">
    <hr>

    <button class="btn btn-submit">Confirm Logout</button>
    <hr>
  </form>
</div>

</body>
</html>
```

```
server:
  port: 8008

spring:
  application:
    name: authority-svc

  cloud:
    config:
      fail-fast: true
      discovery:
        enabled: true
        service-id: cloud-config
      username: configUser
      password: configPassword
#      retry:
#        initial-interval: 2000
#        multiplier: 1.5
#        max-interval: 60000
#        max-attempts: 100
#      uri: http://localhost:8000
#      fail-fast: true

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8002/eureka
    fetch-registry: true
  instance:
    lease-renewal-interval-in-seconds: 10
```

ДОДАТОК Д

USER-SVC

```
package learning.diplom.user.svc.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.data.redis.connection.RedisConnectionFactory
import org.springframework.data.redis.core.RedisTemplate
import org.springframework.data.redis.serializer.GenericJackson2JsonRedisSerializer
import org.springframework.data.redis.serializer.StringRedisSerializer

@Configuration
class RedisConfig {
    @Bean
    fun redisTemplate(connectionFactory: RedisConnectionFactory): RedisTemplate<String, Any> {
        return RedisTemplate<String, Any>().apply {
            setConnectionFactory(connectionFactory)
            val stringRedisSerializer = StringRedisSerializer()
            val jsonSerializer = GenericJackson2JsonRedisSerializer()
            keySerializer = stringRedisSerializer
            valueSerializer = jsonSerializer
            hashKeySerializer = stringRedisSerializer
            hashValueSerializer = jsonSerializer
            setEnableTransactionSupport(true)
        }
    }
}
```

```
package learning.diplom.user.svc.config

import feign.codec.Decoder
import feign.codec.Encoder
import feign.form.spring.SpringFormEncoder
import org.springframework.beans.factory.ObjectFactory
import org.springframework.beans.factory.ObjectProvider
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.boot.autoconfigure.http.HttpMessageConverters
import org.springframework.cloud.openfeign.support.HttpMessageConverterCustomizer
import org.springframework.cloud.openfeign.support.ResponseEntityDecoder
import org.springframework.cloud.openfeign.support.SpringDecoder
import org.springframework.cloud.openfeign.support.SpringEncoder
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.http.converter.protobuf.ProtobufHttpMessageConverter

@Configuration
class OpenFeignConfiguration {
    //Autowire the message converters.
    @Autowired
    private val messageConverters: ObjectFactory<HttpMessageConverters>? = null

    @Autowired
    private val messageCustomizer: ObjectProvider<HttpMessageConverterCustomizer>? = null

    //add the protobuf http message converter
    @Bean
    fun protobufHttpMessageConverter(): ProtobufHttpMessageConverter? {
```

```
        return ProtobufHttpMessageConverter()
    }

    //override the encoder
//    @Bean
//    fun springEncoder(): Encoder? {
//        return SpringEncoder(messageConverters)
//    }

    //override the encoder
    @Bean
    fun springDecoder(): Decoder? {
        return ResponseEntityDecoder(SpringDecoder(messageConverters, messageCustomizer))
    }

    @Bean
    fun feignFormEncoder(): Encoder? {
        return SpringFormEncoder(SpringEncoder(messageConverters))
    }

//    @Bean
//    fun multipartFormEncoder(): Encoder? {
//        return SpringFormEncoder(SpringEncoder {
HttpMessageConverters(RestTemplate().messageConverters) })
//    }
}

package learning.diplom.user.svc.config

import learning.diplom.model.error.lib.handler.ConstraintViolationExceptionHandler
import learning.diplom.model.error.lib.handler.RestExceptionHandler
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder

@Configuration
class BeanConfig {
    @Bean
    fun bCryptPasswordEncoder(): BCryptPasswordEncoder {
        return BCryptPasswordEncoder()
    }

    @Bean
    fun restExceptionHandler(): RestExceptionHandler {
        return RestExceptionHandler()
    }

    @Bean
    fun constraintViolationExceptionHandler(): ConstraintViolationExceptionHandler {
        return ConstraintViolationExceptionHandler()
    }

//    @Bean
//    fun internalExceptionHandler(): InternalExceptionHandler {
//        return InternalExceptionHandler()
//    }

//    @Bean
//    fun requestBodyValidationExceptionHandler(): RequestBodyValidationExceptionHandler {
//        return RequestBodyValidationExceptionHandler()
```

```
//    }
}

package learning.diplom.user.svc.controller.read

import learning.diplom.user.svc.repository.UserRepository
import learning.diplom.user.svc.service.UserService
import learning.diplom.user.svc.util.JwtUtil
import org.springframework.stereotype.Controller
import org.springframework.ui.Model
import org.springframework.web.bind.annotation.CookieValue
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.RequestMapping

@Controller
@RequestMapping("/home")
class HomeController(
    private val jwtUtil: JwtUtil,
    private val userService: UserService
) {

    @GetMapping("")
    fun getHomePage(@CookieValue("info") redisKey: String, model: Model): String {
        val claims = jwtUtil.getClaims(redisKey)!!
        val id = claims["userId"].toString()
        val user = userService.getUserByUserId(id)
        model.addAttribute("user", user)
        return "home"
    }
}

package learning.diplom.user.svc.controller.read

import learning.diplom.user.svc.model.User
import learning.diplom.user.svc.openfeign.client.DocumentSvcClient
import learning.diplom.user.svc.service.UserService
import org.slf4j.LoggerFactory
import org.springframework.http.ResponseEntity
import org.springframework.stereotype.Controller
import org.springframework.ui.Model
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.PathVariable
import org.springframework.web.bind.annotation.RequestMapping

@Controller
@RequestMapping("/user")
class UserReadController(
    private val userService: UserService
) {

    @GetMapping("")
    fun test( ): ResponseEntity<String> {
        return ResponseEntity.ok().body("USER")
    }

    @GetMapping("/{userId}")
    fun getUserById( @PathVariable userId: String, model: Model): String {
        LOG.info("Get user: user = {}", userId)
        val user = userService.getUserByUserId(userId)
        model.addAttribute("user", user)
        return "user"
```

```
    }

    companion object {
        private val LOG = LoggerFactory.getLogger(UserReadController::class.java)
    }
}

package learning.diplom.user.svc.controller.update

import learning.diplom.user.svc.model.User
import learning.diplom.user.svc.openfeign.client.DocumentSvcClient
import learning.diplom.user.svc.service.UserService
import org.slf4j.LoggerFactory
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.validation.annotation.Validated
import org.springframework.web.bind.annotation.*
import org.springframework.web.multipart.MultipartFile
import javax.validation.Valid
import javax.ws.rs.core.MediaType

@RestController
@Validated
@RequestMapping("/user")
class UserUpdateController(
    private val userService: UserService,
    private val documentSvcClient: DocumentSvcClient
) {

    @PostMapping("/create")
    fun createUser(@Valid @RequestBody user: User): ResponseEntity<User> {
        LOG.info("Create user: user = {}", user)
        return ResponseEntity(userService.createUser(user), HttpStatus.CREATED)
    }

//    @PostMapping("/document/upload", consumes = [MediaType.MULTIPART_FORM_DATA])
//    fun uploadDocument(
//        @RequestParam file: MultipartFile,
//        @CookieValue("info") redisKey: String? = null
//    ): ResponseEntity<String> {
//        LOG.info("Started upload file = {}")
//        return documentSvcClient.uploadDocument(file, redisKey)
//    }

    companion object {
        private val LOG = LoggerFactory.getLogger(UserUpdateController::class.java)
    }
}

package learning.diplom.user.svc.model

import com.fasterxml.jackson.annotation.JsonIgnore
import learning.diplom.model.error.lib.department.svc.DepartmentType
import org.jetbrains.annotations.NotNull
import org.springframework.data.annotation.*
import java.time.Instant
import javax.validation.constraints.Email
import javax.validation.constraints.NotBlank
import javax.validation.constraints.Size

data class User(
```

```
    @Id
    @get:ReadOnlyProperty
    var userId: String? = null,
    @NotBlank
    @get:Size(min = 3, max = 16)
    var name: String? = null,
    @NotBlank
    @get:Size(min = 4, max = 20)
    var surname: String? = null,
    @NotBlank
    @get:Size(min = 4, max = 20)
    var patronymic: String? = null,
    @NotNull
    @get:Email
    @get:Size(max = 100)
    var email: String? = null,
    @get:NotNull
    var role: Role? = null,
    @NotNull
    @get:JsonIgnore
    var passwordHash: String? = null,
    @NotNull
    var departmentId: String? = null,
    @CreatedDate
    var createdDate: Instant? = null,
    @LastModifiedDate
    var lastModifiedDate: Instant? = null,
    var deleted: Boolean? = false,
    @Version
    var version: Long? = null,
) {
    enum class Role {
        HEAD_OF_ADMINISTRATION,
        DEPUTY_HEAD_OF_ADMINISTRATION,
        HEAD_OF_DEPARTMENT,
        DEPUTY_HEAD_OF_DEPARTMENT,
        EMPLOYEE
    }
}
```

```
package learning.diplom.user.svc.openfeign.client

import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentRequest
import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentResponse
import learning.diplom.user.svc.config.OpenFeignConfiguration
import org.springframework.cloud.openfeign.FeignClient
import org.springframework.web.bind.annotation.PathVariable
import org.springframework.web.bind.annotation.PostMapping
import org.springframework.web.bind.annotation.RequestBody
import javax.ws.rs.core.MediaType

@FeignClient("\${department-svc-feign-client}", configuration = [OpenFeignConfiguration::class])
interface DepartmentSvcClient {

    @PostMapping(
        "department/{departmentId}/add-user",
        consumes = ["application/x-protobuf"],
        produces = ["application/x-protobuf"]
    )
    fun addUserToDepartment(
        @PathVariable departmentId: String,
```

```
        @RequestBody request: AddUserToDepartmentRequest
    ): AddUserToDepartmentResponse
}

package learning.diplom.user.svc.repository

import learning.diplom.user.svc.model.User
import org.springframework.data.mongodb.repository.MongoRepository

interface UserRepository: MongoRepository<User, String> {
    fun deleteByUserId(userId: String)
    fun findByUserId (userId: String): User
    fun findByName(firstName: String): User
    fun findByEmail(email: String): User
    fun existsByEmail(email: String): Boolean
    fun existsByUserId(userId: String): Boolean
}

package learning.diplom.user.svc.service

import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentRequest
import learning.diplom.model.error.lib.ServerEntity
import learning.diplom.model.error.lib.exception.rest.EntityAlreadyExistsRestException
import learning.diplom.model.error.lib.exception.rest.EntityNotFoundRestException
import learning.diplom.model.error.lib.exception.rest.RestException
import learning.diplom.model.error.lib.mapper.ErrorMapper.mapToRestException
import learning.diplom.model.error.lib.user.svc.Role
import learning.diplom.user.svc.openfeign.client.DepartmentSvcClient
import learning.diplom.user.svc.model.User
import learning.diplom.user.svc.repository.UserRepository
import learning.diplom.user.svc.util.codec.asEnumOf
import learning.diplom.user.svc.util.codec.mapAsEnumOf
import org.slf4j.LoggerFactory
import org.springframework.http.HttpStatus
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder
import org.springframework.stereotype.Service

@Service
class UserService(
    private val userRepository: UserRepository,
    private val departmentSvcClient: DepartmentSvcClient,
    private val bCryptPasswordEncoder: BCryptPasswordEncoder
) {
    //// UPDATE METHODS ////
    fun createUser(user: User): User {
        LOG.info("Started creating user: user = {}", user)
        if (userRepository.existsByEmail(user.email!!.toString())) {
            throw EntityAlreadyExistsRestException(ServerEntity.USER, user.toString())
        }

        user.passwordHash = bCryptPasswordEncoder.encode(user.passwordHash.toString())

        val savedUser = userRepository.save(user)
        LOG.info("Started adding user = {}, to department = {}", savedUser.departmentId)

        val addUserToDepartmentResponse = try {
            sendAddUserToDepartmentRequest(user)
        } catch (ex: Exception) {
            LOG.error("ERROR from department-svc: ", ex)
            userRepository.deleteByUserId(savedUser.userId!!)
            throw RestException(HttpStatus.INTERNAL_SERVER_ERROR, "Internal error")
```

```
        }

        if (addUserToDepartmentResponse.hasError()) {
            userRepository.deleteByUserId(savedUser.userId!!)
            throw addUserToDepartmentResponse.error.mapToRestException()
        }
        return savedUser
    }

    //// READ METHODS ////
    fun getUserByUserId(userId: String): User {
        return if (userRepository.existsByUserId(userId)) {
            userRepository.findByUserId(userId)
        } else {
            throw EntityNotFoundRestException(ServerEntity.USER, "userId = $userId")
        }
    }

    private fun sendAddUserToDepartmentRequest(user: User) =
        departmentSvcClient.addUserToDepartment(
            user.departmentId!!,
            AddUserToDepartmentRequest.newBuilder()
                .setUserId(user.userId)
                .setName(user.name)
                .setSurname(user.surname)
                .setRole(user.role.asEnumOf())
                .build()
        )

    companion object {
        private val LOG = LoggerFactory.getLogger(UserService::class.java)
    }
}
```

```
package learning.diplom.user.svc.util.codec

inline fun <reified V : Enum<V>> Enum<*>?.asEnumOf(): V? = this?.let {
java.lang.Enum.valueOf(V::class.java, it.name) }
inline fun <reified V : Enum<V>> Enum<*>?.safeAsEnumOf(): V? {
    return try {
        this.asEnumOf<V>()
    } catch (e: IllegalArgumentException) {
        null
    }
}

inline fun <reified V : Enum<V>> List<Enum<*>>.mapAsEnumOf(): List<V?> {
    return map {
        it.asEnumOf<V>()
    }
}
```

```
package learning.diplom.user.svc.util

import io.jsonwebtoken.Claims
import io.jsonwebtoken.Jwts
import learning.diplom.model.error.lib.ServerEntity
import learning.diplom.model.error.lib.exception.rest.EntityNotFoundRestException
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.data.redis.core.RedisTemplate
```

```
import org.springframework.stereotype.Component

@Component
class JwtUtil(
    @Value("\${jwt.secret}")
    private val jwtSecret: String? = null,
    @Value("\${jwt.validity}")
    private val tokenValidity: Long = 0,
    @Value("\${jwt.info}")
    private val info: String? = null,
    private val redisTemplate: RedisTemplate<String, Any>
) {

    fun getClaims(redisKey: String?): Claims? {
        val token = getJwtToken(redisKey)
        if (redisKey.isNullOrBlank() || token == null) {
            LOG.error("Token did not find")
            throw EntityNotFoundRestException(ServerEntity.TOKEN, "redisKey = $redisKey")
        }
        try {
            return Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token.toString()).body
        } catch (e: Exception) {
            println(e.message + " => " + e)
        }
        return null
    }

    fun getJwtToken(redisKey: String?): Any? {
        return redisTemplate.opsForValue().get(redisKey ?: "empty")
    }

    companion object {
        private val LOG = LoggerFactory.getLogger(JwtUtil::class.java)
    }

}

package learning.diplom.user.svc

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import org.springframework.cloud.openfeign.EnableFeignClients
import org.springframework.data.mongodb.config.EnableMongoAuditing

@SpringBootApplication
@EnableFeignClients
@EnableMongoAuditing
class UserSvcApplication

fun main(args: Array<String>) {
    runApplication<UserSvcApplication>(*args)
}

<!doctype html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
```

```
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
        crossorigin="anonymous"></script>
  <title>User Page</title>
</head>
<body>
<!--<header header th:insert="blocks/header :: header"></header>-->
<div th:replace="blocks/header :: header"></div>

<div class="container">

  <div th:object="${user}">
    <h4>name</h4>
    <h3 th:text="${user.getName()}"></h3>
    <hr>

    <h4>surname</h4>
    <h3 th:text="${user.getSurname()}"></h3>
    <hr>

    <h4>patronymic</h4>
    <h3 th:text="${user.getPatronymic()}"></h3>
    <hr>

    <h4>email</h4>
    <h3 th:text="${user.getEmail()}"></h3>
    <hr>

    <h4>role</h4>
    <h3 th:text="${user.getRole()}"></h3>
    <!--    <a th:href="@{/users/{id}/edit(id =${userInfo.getId()})}">Edit</a>-->
    <hr>
  </div>

</div>

<!--<footer class="footer" th:insert="blocks/footer :: footer"></footer>-->
</body>
</html>

<!doctype html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
        crossorigin="anonymous"></script>
  <title>User Page</title>
</head>
<body>
<!--<header header th:insert="blocks/header :: header"></header>-->
```

```
<div th:replace="blocks/header :: header"></div>

<div class="container">

  <div th:object="${user}">
    <h4>name</h4>
    <h3 th:text="${user.getName()}"></h3>
    <hr>

    <h4>surname</h4>
    <h3 th:text="${user.getSurname()}"></h3>
    <hr>

    <h4>patronymic</h4>
    <h3 th:text="${user.getPatronymic()}"></h3>
    <hr>

    <h4>email</h4>
    <h3 th:text="${user.getEmail()}"></h3>
    <hr>

    <h4>role</h4>
    <h3 th:text="${user.getRole()}"></h3>
    <hr>
  </div>
</div>

<!--<footer class="footer" th:insert="blocks/footer :: footer"></footer>-->
</body>
</html>
```

ДОДАТОК Е

DEPARTMENT-SVC

```
syntax = "proto3";

package learning.diplom.department.svc.api.reqresp.user.svc;
option java_package = "learning.diplom.department.svc.api.reqresp.user.svc";
option java_multiple_files = true;

import "learning/diplom/model/error/lib/success_response.proto";
import "learning/diplom/model/error/lib/error.proto";
import "learning/diplom/model/error/lib/user/svc/role.proto";

message AddUserToDepartmentRequest {
   string user_id = 1;
   string name = 2;
   string surname = 3;
   learning.diplom.model.error.lib.user.svc.Role role = 4;
}

message AddUserToDepartmentResponse {
   oneof response {
      learning.diplom.model.error.lib.Success success = 1;
      learning.diplom.model.error.lib.Error error = 2;
   }
}

package learning.diplom.department.svc.controller.update

import learning.diplom.department.svc.model.Department
import learning.diplom.department.svc.openfeign.service.OpenfeignDepartmentService
import learning.diplom.department.svc.service.DepartmentService
import org.slf4j.LoggerFactory
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*
import javax.validation.Valid

@RestController
@RequestMapping("/department")
class DepartmentUpdateController(
   private val departmentService: DepartmentService
) {

   @PostMapping("/")
   fun createDepartment(@Valid @RequestBody department: Department):
ResponseEntity<Department> {
      LOG.info("Create department = {}", department)
      return ResponseEntity(departmentService.createDepartment(department), HttpStatus.CREATED)

   }

   companion object {
      private val LOG = LoggerFactory.getLogger(DepartmentUpdateController::class.java)
   }
}
```

```
package learning.diplom.department.svc.model

import javax.validation.constraints.NotBlank
import javax.validation.constraints.NotNull

data class BriefUser(
    @NotNull
    var userId: String? = null,
    @NotBlank
    var name: String? = null,
    @NotBlank
    var surname: String? = null,
    @NotNull
    var role: Role? = null
) {
    enum class Role {
        HEAD_OF_ADMINISTRATION,
        DEPUTY_HEAD_OF_ADMINISTRATION,
        HEAD_OF_DEPARTMENT,
        DEPUTY_HEAD_OF_DEPARTMENT,
        EMPLOYEE
    }
}

package learning.diplom.department.svc.model

import org.springframework.data.annotation.CreatedDate
import org.springframework.data.annotation.Id
import org.springframework.data.annotation.LastModifiedDate
import org.springframework.data.annotation.Version
import java.time.Instant
import javax.validation.constraints.NotBlank
import javax.validation.constraints.NotNull

data class Department(
    @Id
    var departmentId: String? = null,
    @get:NotNull
    var administrationId: String? = null,
    @get:NotBlank
    var name: String? = null,
    var headOfDepartmentId: String? = null,
    var users: MutableList<BriefUser> = mutableListOf(),
    @get:NotNull
    var usersLimit: Int? = null,
    @get:NotNull
    var departmentType: DepartmentType? = null,
    @CreatedDate
    var createdDate: Instant? = null,
    @LastModifiedDate
    var lastModifiedDate: Instant? = null,
    @Version
    val version: Long? = null,
    var deleted: Boolean? = false
) {
    enum class DepartmentType {
        GENERAL_DEPARTMENT,
        FINANCIAL_DEPARTMENT,
        HUMAN_RESOURCES_DEPARTMENT,
        ARCHITECTURE_DEPARTMENT
```

```
    }
}

package learning.diplom.department.svc.openfeign.client

import
learning.diplom.administration.svc.api.reqresp.department.svc.AddDepartmentToAdministrationReques
t
import
learning.diplom.administration.svc.api.reqresp.department.svc.AddDepartmentToAdministrationRespo
nse
import learning.diplom.department.svc.config.OpenFeignConfiguration
import org.springframework.cloud.openfeign.FeignClient
import org.springframework.web.bind.annotation.PathVariable
import org.springframework.web.bind.annotation.PostMapping
import org.springframework.web.bind.annotation.RequestBody

@FeignClient("\${administration-svc-feign-client}", configuration = [OpenFeignConfiguration::class])
interface AdministrationSvcClient {

    @PostMapping(
        "/administration/{administrationId}/add-department",
        consumes = ["application/x-protobuf"],
        produces = ["application/x-protobuf"]
    )
    fun addDepartmentToAdministration(
        @PathVariable administrationId: String,
        @RequestBody request: AddDepartmentToAdministrationRequest
    ): AddDepartmentToAdministrationResponse
}

package learning.diplom.department.svc.openfeign.controller.update

import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentRequest
import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentResponse
import learning.diplom.department.svc.openfeign.service.OpenfeignDepartmentService
import org.slf4j.LoggerFactory
import org.springframework.web.bind.annotation.*

@RestController
@RequestMapping(
    "/department",
    consumes = ["application/x-protobuf"],
    produces = ["application/x-protobuf"]
)
class OpenfeignUpdateController(
    private val openfeignDepartmentService: OpenfeignDepartmentService
) {

    @PostMapping("/{departmentId}/add-user")
    fun addUserToDepartment(
        @PathVariable departmentId: String,
        @RequestBody request: AddUserToDepartmentRequest
    ): AddUserToDepartmentResponse {
        LOG.info("Add user to department: departmentId = {}, userId = {}", departmentId, request.userId)
        return openfeignDepartmentService.addUserToDepartment(departmentId, request)
    }

    companion object {
        private val LOG = LoggerFactory.getLogger(OpenfeignUpdateController::class.java)
```

```
    }
}

package learning.diplom.department.svc.openfeign.service

import com.fasterxml.jackson.annotation.JsonIgnoreProperties
import learning.diplom.department.svc.util.codec.mapAsEnumOf
import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentRequest
import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentResponse
import learning.diplom.department.svc.model.BriefUser
import learning.diplom.department.svc.repository.DepartmentRepository
import learning.diplom.department.svc.util.codec.asEnumOf
import learning.diplom.model.error.lib.Entity
import learning.diplom.model.error.lib.Success
import learning.diplom.model.error.lib.exception.proto.ProtoErrorUtil
import org.slf4j.LoggerFactory
import org.springframework.dao.EmptyResultDataAccessException
import org.springframework.stereotype.Service

@Service
class OpenfeignDepartmentService(
    private val departmentRepository: DepartmentRepository,
    private val openfeignUtilService: OpenfeignUtilService
) {
    fun addUserToDepartment(departmentId: String, request: AddUserToDepartmentRequest):
AddUserToDepartmentResponse {
        val responseBuilder = AddUserToDepartmentResponse.newBuilder()

        val department = try {
            departmentRepository.findByDepartmentId(departmentId)
        } catch (ex: EmptyResultDataAccessException) {
            return responseBuilder
                .setError(ProtoErrorUtil.createEntityNotFoundErrorBuilder(
                    Entity.DEPARTMENT,
                    "departmentId = $departmentId")
                )
                .build()
        }

        openfeignUtilService.addUserToDepartmentCheckPreconditions(request, department,
responseBuilder)
            ?.let { return it }

        if (department.headOfDepartmentId == null) {
            department.headOfDepartmentId = request.userId
        }
        department.users.add(
            BriefUser(
                request.userId,
                request.name,
                request.surname,
                request.role.asEnumOf<BriefUser.Role>()
            )
        )
        departmentRepository.save(department)

        LOG.info("Adding user = {} was successful", request)

        return responseBuilder
            .setSuccess(Success.getDefaultInstance())
            .build()
```

```
    }

    companion object {
        private val LOG = LoggerFactory.getLogger(OpenfeignDepartmentService::class.java)
    }
}

package learning.diplom.department.svc.openfeign.service

import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentRequest
import learning.diplom.department.svc.api.reqresp.user.svc.AddUserToDepartmentResponse
import learning.diplom.department.svc.model.Department
import learning.diplom.model.error.lib.Entity
import
learning.diplom.model.error.lib.exception.proto.ProtoErrorUtil.createEntityAlreadyExistsErrorBuilder
import
learning.diplom.model.error.lib.exception.proto.ProtoErrorUtil.createEntityLimitExceededErrorBuilder
import org.slf4j.LoggerFactory
import org.springframework.stereotype.Service

@Service
class OpenfeignUtilService {
    fun addUserToDepartmentCheckPreconditions(
        request: AddUserToDepartmentRequest,
        department: Department,
        responseBuilder: AddUserToDepartmentResponse.Builder
    ): AddUserToDepartmentResponse? {
        LOG.info("Check preconditions for adding user = {} to Department = {}", request, department)
        if (department.users.isEmpty()) {
            return null
        }

        val usersSizeWithNewUser = department.users.size + 1
        if (usersSizeWithNewUser >= department.usersLimit!!) {
            return responseBuilder
                .setError(
                    createEntityLimitExceededErrorBuilder(
                        Entity.USER,
                        "request = $request, department = $department"
                    )
                )
                .build()
        }

        if (department.users.map { it.userId }.contains(request.userId)) {
            return responseBuilder
                .setError(
                    createEntityAlreadyExistsErrorBuilder(
                        Entity.USER,
                        "request = $request, department = $department"
                    )
                )
                .build()
        }
        return null
    }

    companion object {
        private val LOG = LoggerFactory.getLogger(OpenfeignUtilService::class.java)
    }
}
```

```
package learning.diplom.department.svc.repository

import learning.diplom.department.svc.model.Department
import org.springframework.data.mongodb.repository.MongoRepository

interface DepartmentRepository : MongoRepository<Department, String> {
    fun findByDepartmentId(departmentId: String): Department
    fun deleteByDepartmentId(departmentId: String)
    fun existsByDepartmentId(departmentId: String): Boolean
}

package learning.diplom.department.svc.service

import
learning.diplom.administration.svc.api.reqresp.department.svc.AddDepartmentToAdministrationReques
t
import learning.diplom.department.svc.model.Department
import learning.diplom.department.svc.openfeign.client.AdministrationSvcClient
import learning.diplom.department.svc.repository.DepartmentRepository
import learning.diplom.department.svc.util.codec.asEnumOf
import learning.diplom.model.error.lib.ServerEntity
import learning.diplom.model.error.lib.exception.rest.EntityAlreadyExistsRestException
import learning.diplom.model.error.lib.exception.rest.RestException
import learning.diplom.model.error.lib.mapper.ErrorMapper.mapToRestException
import org.slf4j.LoggerFactory
import org.springframework.http.HttpStatus
import org.springframework.stereotype.Service

@Service
class DepartmentService(
    private val departmentRepository: DepartmentRepository,
    private val administrationSvcClient: AdministrationSvcClient
) {
    //// UPDATE METHODS ////
    fun createDepartment(department: Department): Department {
        if (departmentRepository.existsByDepartmentId(department.departmentId!!)) {
            throw EntityAlreadyExistsRestException(ServerEntity.USER, "department = $department")
        }

        val savedDepartment = departmentRepository.save(department)
        LOG.info("Started adding department = {}, to administration = {}",
savedDepartment.departmentId)

        val addDepartmentToAdministrationResponse = try {
            sendAddDepartmentToAdministrationRequest(department)
        } catch (ex: Exception) {
            LOG.error("ERROR from administration-svc: ", ex)
            departmentRepository.deleteByDepartmentId(savedDepartment.departmentId!!)
            throw RestException(HttpStatus.INTERNAL_SERVER_ERROR, "Internal error")
        }

        if (addDepartmentToAdministrationResponse.hasError()) {
            departmentRepository.deleteByDepartmentId(savedDepartment.departmentId!!)
            throw addDepartmentToAdministrationResponse.error.mapToRestException()
        }
        return savedDepartment
    }

    //// READ METHODS ////
```

```
    private fun sendAddDepartmentToAdministrationRequest(department: Department) =
        administrationSvcClient.addDepartmentToAdministration(
            department.administrationId!!,
            AddDepartmentToAdministrationRequest.newBuilder()
                .setDepartmentId(department.departmentId)
                .setHeadOfDepartmentId(department.headOfDepartmentId)
                .setDepartmentType(department.departmentType.asEnumOf())
                .build()
        )

    companion object {
        private val LOG = LoggerFactory.getLogger(DepartmentService::class.java)
    }
}

package learning.diplom.department.svc.util.codec

inline fun <reified V : Enum<V>> Enum<*>?.asEnumOf(): V? = this?.let {
java.lang.Enum.valueOf(V::class.java, it.name) }
inline fun <reified V : Enum<V>> Enum<*>?.safeAsEnumOf(): V? {
    return try {
        this.asEnumOf<V>()
    } catch (e: IllegalArgumentException) {
        null
    }
}

inline fun <reified V : Enum<V>> List<Enum<*>>.mapAsEnumOf(): List<V?> {
    return map {
        it.asEnumOf<V>()
    }
}

package learning.diplom.department.svc

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import org.springframework.cloud.openfeign.EnableFeignClients
import org.springframework.data.mongodb.config.EnableMongoAuditing

@SpringBootApplication
@EnableFeignClients
@EnableMongoAuditing
class DepartmentSvcApplication

fun main(args: Array<String>) {
    runApplication<DepartmentSvcApplication>(*args)
}

package learning.diplom.department.svc.config

import learning.diplom.model.error.lib.handler.ConstraintViolationExceptionHandler
import learning.diplom.model.error.lib.handler.RequestBodyValidationExceptionHandler
import learning.diplom.model.error.lib.handler.RestExceptionHandler
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder

@Configuration
class BeanConfig {
```

```
    @Bean
    fun bCryptPasswordEncoder(): BCryptPasswordEncoder {
        return BCryptPasswordEncoder()
    }

    @Bean
    fun restExceptionHandler(): RestExceptionHandler {
        return RestExceptionHandler()
    }

    @Bean
    fun constraintViolationExceptionHandler(): ConstraintViolationExceptionHandler {
        return ConstraintViolationExceptionHandler()
    }

//    @Bean
//    fun requestBodyValidationExceptionHandler(): RequestBodyValidationExceptionHandler {
//        return RequestBodyValidationExceptionHandler()
//    }
}
package learning.diplom.department.svc.config

import feign.codec.Decoder
import feign.codec.Encoder
import org.springframework.beans.factory.ObjectFactory
import org.springframework.beans.factory.ObjectProvider
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.boot.autoconfigure.http.HttpMessageConverters
import org.springframework.cloud.openfeign.support.HttpMessageConverterCustomizer
import org.springframework.cloud.openfeign.support.ResponseEntityDecoder
import org.springframework.cloud.openfeign.support.SpringDecoder
import org.springframework.cloud.openfeign.support.SpringEncoder
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.http.converter.protobuf.ProtobufHttpMessageConverter

@Configuration
class OpenFeignConfiguration {
    //Autowire the message converters.
    @Autowired
    private val messageConverters: ObjectFactory<HttpMessageConverters>? = null

    @Autowired
    private val messageCustomizer: ObjectProvider<HttpMessageConverterCustomizer>? = null

    //add the protobuf http message converter
    @Bean
    fun protobufHttpMessageConverter(): ProtobufHttpMessageConverter? {
        return ProtobufHttpMessageConverter()
    }

    //override the encoder
    @Bean
    fun springEncoder(): Encoder? {
        return SpringEncoder(messageConverters)
    }

    //override the encoder
    @Bean
    fun springDecoder(): Decoder? {
        return ResponseEntityDecoder(SpringDecoder(messageConverters, messageCustomizer))
```

```
    }
}

server:
  port: 8005

spring:
  application:
    name: department-svc
  cloud:
    config:
      username: configUser
      password: configPassword
      fail-fast: true
      discovery:
        enabled: true
        service-id: cloud-config

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8002/eureka
    fetch-registry: true
  instance:
    lease-renewal-interval-in-seconds: 10
```

ДОДАТОК Є

DOCUMENT-SVC

```
package learning.diplom.document.svc.config

import com.mongodb.client.MongoClient
import com.mongodb.client.gridfs.GridFSBucket
import com.mongodb.client.gridfs.GridFSBuckets
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration

@Configuration
class BeanConfig {
    @Bean
    fun getGridFSBucket(mongoClient: MongoClient): GridFSBucket? {
        val database = mongoClient.getDatabase("document-svc")
        return GridFSBuckets.create(database)
    }
}

package learning.diplom.document.svc.config

import feign.codec.Decoder
import feign.codec.Encoder
import org.springframework.beans.factory.ObjectFactory
import org.springframework.beans.factory.ObjectProvider
import org.springframework.beans.factory.annotation.Autowired
import
org.springframework.boot.autoconfigure.http.HttpMessageConverters
import
org.springframework.cloud.openfeign.support.HttpMessageConverterCusto
mizer
import
org.springframework.cloud.openfeign.support.ResponseEntityDecoder
import org.springframework.cloud.openfeign.support.SpringDecoder
import org.springframework.cloud.openfeign.support.SpringEncoder
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import
org.springframework.http.converter.protobuf.ProtobufHttpMessageConver
ter

@Configuration
class OpenFeignConfiguration {
    //Autowire the message converters.
    @Autowired
    private val messageConverters:
ObjectFactory<HttpMessageConverters>? = null

    @Autowired
    private val messageCustomizer:
ObjectProvider<HttpMessageConverterCustomizer>? = null

    //add the protobuf http message converter
    @Bean
    fun protobufHttpMessageConverter(): ProtobufHttpMessageConverter?
{
```

```
        return ProtobufHttpMessageConverter()
    }

    //override the encoder
    @Bean
    fun springEncoder(): Encoder? {
        return SpringEncoder(messageConverters)
    }

    //override the encoder
    @Bean
    fun springDecoder(): Decoder? {
        return ResponseEntityDecoder(SpringDecoder(messageConverters,
messageCustomizer))
    }
}

package learning.diplom.document.svc.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import
org.springframework.data.redis.connection.RedisConnectionFactory
import org.springframework.data.redis.core.RedisTemplate
import
org.springframework.data.redis.serializer.GenericJackson2JsonRedisSer
ializer
import
org.springframework.data.redis.serializer.StringRedisSerializer

@Configuration
class RedisConfig {
    @Bean
    fun redisTemplate(connectionFactory: RedisConnectionFactory):
RedisTemplate<String, Any> {
        return RedisTemplate<String, Any>().apply {
            setConnectionFactory(connectionFactory)
            val stringRedisSerializer = StringRedisSerializer()
            val jsonSerializer = GenericJackson2JsonRedisSerializer()
            keySerializer = stringRedisSerializer
            valueSerializer = jsonSerializer
            hashKeySerializer = stringRedisSerializer
            hashValueSerializer = jsonSerializer
            setEnableTransactionSupport(true)
        }
    }
}

package learning.diplom.document.svc.controller.read

import learning.diplom.document.svc.model.DocumentRequest
import learning.diplom.document.svc.service.DocumentService
import org.springframework.stereotype.Controller
import org.springframework.ui.Model
import org.springframework.web.bind.annotation.*

@Controller
@RequestMapping("/document")
class DocumentReadController(
    private val documentService: DocumentService
) {
```

```
    @GetMapping("/list")
    fun getAllDocuments(
        @CookieValue("info") redisKey: String,
        model: Model
    ): String {
        val documentList =
documentService.getAllDocumentsByUserId(redisKey)
        val documentId: String? = null
        model.addAttribute("documentList", documentList)
        model.addAttribute("documentRequest", DocumentRequest())
        model.addAttribute("documentId", documentId)
        return "document-list"
    }

    @GetMapping("/upload-document")
    fun uploadDocument(
        @CookieValue("info") redisKey: String,
        model: Model
    ): String {

        return "upload-document"
    }

    @GetMapping("/approve-document")
    fun approveDocument(
        @CookieValue("info") redisKey: String,
        @ModelAttribute @RequestParam result: Boolean,
        model: Model
    ): String {
        model.addAttribute("result", result)
        return "approve-document"
    }
}

package learning.diplom.document.svc.controller.read

import learning.diplom.document.svc.service.SignService
import org.slf4j.LoggerFactory
import org.springframework.stereotype.Controller
import org.springframework.ui.Model
import org.springframework.web.bind.annotation.*

@Controller
@RequestMapping("/sign")
class SignReadController(
    private val signService: SignService
) {

    @GetMapping("/keys")
    fun getKeys(
        @CookieValue("info") redisKey: String,
        model: Model
    ): String {
        val privateKey = signService.getPrivateKey(redisKey)
        model.addAttribute("privateKey", privateKey)

        val publicKey = signService.getPublicKey(redisKey)
        model.addAttribute("publicKey", publicKey)
        return "keys"
    }

    @GetMapping("/verify")
```

```
    fun verifySignature(): String {
        LOG.info("Verify document page")

        return "verify-signature"
    }

    @GetMapping("/verify-result")
    fun approveDocument(
        @CookieValue("info") redisKey: String,
        @ModelAttribute @RequestParam result: Boolean,
        model: Model
    ): String {
        LOG.info("Verify-result  page")

        model.addAttribute("result", result)
        return "verify-result"
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(SignReadController::class.java)
    }
}

package learning.diplom.document.svc.controller.update

import learning.diplom.document.svc.model.DocumentRequest
import learning.diplom.document.svc.service.DocumentService
import learning.diplom.document.svc.util.JwtUtil
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.stereotype.Controller
import org.springframework.web.bind.annotation.*
import org.springframework.web.multipart.MultipartFile
import org.springframework.web.servlet.mvc.support.RedirectAttributes
import org.springframework.web.servlet.view.RedirectView
import javax.servlet.http.HttpServletResponse
import javax.ws.rs.core.MediaType

@Controller
@RequestMapping("/document")
@CrossOrigin("*")
class DocumentUpdateController(
    private val documentService: DocumentService,
    private val jwtUtil: JwtUtil,
    @Value("\${gateway.host}")
    private val host: String? = null,
    @Value("\${gateway.port}")
    private val port: String? = null,
) {

    @PostMapping("/upload-document", consumes =
[MediaType.MULTIPART_FORM_DATA])
    fun uploadDocument(
        @RequestParam file: MultipartFile,
        @CookieValue("info") redisKey: String? = null
    ): RedirectView {
        LOG.info("Started storing file = {}", file)
        documentService.uploadDocument(file, redisKey)
        return RedirectView(
            "$HTTP_PREFIX$host:$port/document/list"
        )
```

```
    }

    @PostMapping("/sign-document", consumes =
[MediaType.MULTIPART_FORM_DATA])
    fun signDocument(
        @RequestParam id: String,
        @RequestBody file: MultipartFile,
        @CookieValue("info") redisKey: String? = null
    ) {
        println("fgergwhwerh \n\n$id")
    }

    @PostMapping("/approve-document")
    fun approveDocument(
        @ModelAttribute @RequestBody documentRequest:
DocumentRequest? = null,
        @CookieValue("info") redisKey: String? = null,
        redirectionAttribute: RedirectAttributes
    ): RedirectView {
        LOG.info("Approve document = {}", documentRequest!!.id)
        val result = documentService.approveDocument(redisKey,
documentRequest.id)
        redirectionAttribute.addAttribute("result", result)
        return RedirectView(
            "$HTTP_PREFIX$host:$port/document/approve-document"
        )
    }

    @PostMapping(
        "/download-document",
        produces = [MediaType.APPLICATION_FORM_URLENCODED],
        consumes = [MediaType.APPLICATION_FORM_URLENCODED]
    )
    fun downloadDocument(
        @ModelAttribute @RequestBody request: DocumentRequest,
        httpServletResponse: HttpServletResponse,
    ) {
        LOG.info("Download file by id = {}", request.id)
        documentService.downloadDocument(request.id!!,
httpServletResponse)
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(DocumentUpdateController::class.java)
        private const val HTTP_PREFIX = "http://"
        private const val USER_ID_JWT_CLAIM = "userId"
    }
}

package learning.diplom.document.svc.controller.update

import learning.diplom.document.svc.service.DocumentService
import learning.diplom.document.svc.service.SignService
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*
import org.springframework.web.multipart.MultipartFile
import org.springframework.web.servlet.mvc.support.RedirectAttributes
import org.springframework.web.servlet.view.RedirectView
import javax.ws.rs.core.MediaType
```

```
@RestController
@RequestMapping("/sign")
class SignUpdateController(
    private val signService: SignService,
    private val documentService: DocumentService,
    @Value("\${gateway.host}")
    private val host: String? = null,
    @Value("\${gateway.port}")
    private val port: String? = null
) {

    @PostMapping("")
    fun generateKeyPairForDepartment(departmentId: String):
ResponseEntity<String> {
        LOG.info("Generate key pair for department = {}",
departmentId)
        signService.generateRsaKeyPair(departmentId)
        return ResponseEntity.ok().body("SUCCESS")
    }

    @PostMapping("/download-private-key")
    fun downloadPrivateKey() {

    }

    @PostMapping(
        "/verify",
        produces = [MediaType.MULTIPART_FORM_DATA],
        consumes = [MediaType.MULTIPART_FORM_DATA]
    )
    fun verifySignature(
        @ModelAttribute @RequestBody document: MultipartFile,
        @ModelAttribute @RequestBody signature: MultipartFile,
        @ModelAttribute @RequestBody publicKey: MultipartFile,
        @CookieValue("info") redisKey: String? = null,
        redirectionAttribute: RedirectAttributes
    ): RedirectView {
        LOG.info("Verify signature {} by publicKey {} for document",
signature, publicKey, document.name)
        val result = signService.verifySignature(publicKey,
signature, document)
        redirectionAttribute.addAttribute("result", result)
        return RedirectView(
            "${HTTP_PREFIX}$host:$port/sign/verify-result"
        )
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(SignUpdateController::class.java)
        private const val HTTP_PREFIX = "http://"
    }
}

package learning.diplom.document.svc.model

import learning.diplom.model.error.lib.department.svc.DepartmentType
import org.springframework.data.annotation.CreatedDate
import org.springframework.data.annotation.Id
import org.springframework.data.annotation.LastModifiedDate
import org.springframework.data.annotation.Version
```

```
import java.time.Instant
import javax.validation.constraints.NotBlank

data class Document(
    @Id
    var documentId: String? = null,
    @get:NotBlank
    var userId: String? = null,
    var fileName: String? = null,
    var contentType: String? = null,
    var size: Long? = null,
    var departmentId: String,
    // Approve fields
    var headOfAdministrationApprove: Boolean = false,
    var deputyHeadOfAdministrationApprove: Boolean = false,
    var headOfDepartmentApprove: Boolean = false,
    var deputyHeadOfDepartmentApprove: Boolean = false,
    var employeeApprove: Boolean = false,
    // SIGNATURE
    var signatureExists: Boolean = false,
    var signature: ByteArray = byteArrayOf(),
    var publicKey: ByteArray = byteArrayOf(),

    var departmentType: DepartmentType? = null,
    @CreatedDate
    var createdDate: Instant? = null,
    @LastModifiedDate
    var lastModifiedDate: Instant? = null,
    @Version
    val version: Long? = 0,
    var deleted: Boolean? = false
) {
    enum class DepartmentType {
        GENERAL_DEPARTMENT,
        FINANCIAL_DEPARTMENT,
        HUMAN_RESOURCES_DEPARTMENT,
        ARCHITECTURE_DEPARTMENT
    }

    override fun equals(other: Any?): Boolean {
        if (this === other) return true
        if (javaClass != other?.javaClass) return false

        other as Document

        if (documentId != other.documentId) return false
        if (userId != other.userId) return false
        if (fileName != other.fileName) return false
        if (contentType != other.contentType) return false
        if (size != other.size) return false
        if (departmentId != other.departmentId) return false
        if (headOfAdministrationApprove !=
other.headOfAdministrationApprove) return false
        if (deputyHeadOfAdministrationApprove !=
other.deputyHeadOfAdministrationApprove) return false
        if (headOfDepartmentApprove != other.headOfDepartmentApprove)
return false
        if (deputyHeadOfDepartmentApprove !=
other.deputyHeadOfDepartmentApprove) return false
        if (employeeApprove != other.employeeApprove) return false
        if (signatureExists != other.signatureExists) return false
        if (signature != null) {
```

```
            if (other.signature == null) return false
            if (!signature.contentEquals(other.signature)) return
false
        } else if (other.signature != null) return false
        if (departmentType != other.departmentType) return false
        if (createdDate != other.createdDate) return false
        if (lastModifiedDate != other.lastModifiedDate) return false
        if (version != other.version) return false
        if (deleted != other.deleted) return false

        return true
    }

    override fun hashCode(): Int {
        var result = documentId?.hashCode() ?: 0
        result = 31 * result + (userId?.hashCode() ?: 0)
        result = 31 * result + (fileName?.hashCode() ?: 0)
        result = 31 * result + (contentType?.hashCode() ?: 0)
        result = 31 * result + (size?.hashCode() ?: 0)
        result = 31 * result + departmentId.hashCode()
        result = 31 * result + headOfAdministrationApprove.hashCode()
        result = 31 * result +
deputyHeadOfAdministrationApprove.hashCode()
        result = 31 * result + headOfDepartmentApprove.hashCode()
        result = 31 * result +
deputyHeadOfDepartmentApprove.hashCode()
        result = 31 * result + employeeApprove.hashCode()
        result = 31 * result + signatureExists.hashCode()
        result = 31 * result + (signature?.contentHashCode() ?: 0)
        result = 31 * result + (departmentType?.hashCode() ?: 0)
        result = 31 * result + (createdDate?.hashCode() ?: 0)
        result = 31 * result + (lastModifiedDate?.hashCode() ?: 0)
        result = 31 * result + (version?.hashCode() ?: 0)
        result = 31 * result + (deleted?.hashCode() ?: 0)
        return result
    }
}

package learning.diplom.document.svc.model

data class DocumentRequest(
    var id: String? = null
)

package learning.diplom.document.svc.model

enum class Role {
    HEAD_OF_ADMINISTRATION,
    DEPUTY_HEAD_OF_ADMINISTRATION,
    HEAD_OF_DEPARTMENT,
    DEPUTY_HEAD_OF_DEPARTMENT,
    EMPLOYEE
}

package learning.diplom.document.svc.model

import org.bson.types.Binary
import org.springframework.data.annotation.*
import java.time.Instant
import javax.validation.constraints.NotNull

data class Sign(
```

```
    @Id
    @get:ReadOnlyProperty
    var signId: String? = null,
    @get:NotNull
    var departmentId: String? = null,
    var publicKey: Binary? = null,
    var privateKey: Binary? = null,
    @CreatedDate
    var createdDate: Instant? = null,
    @LastModifiedDate
    var lastModifiedDate: Instant? = null,
    @Version
    var version: Long? = null
) {
}

package learning.diplom.document.svc.repository

import learning.diplom.document.svc.model.Sign
import org.springframework.data.mongodb.repository.MongoRepository

interface SignRepository: MongoRepository<Sign, String> {

    fun findSignByDepartmentId(departmentId: String): Sign
    fun existsByDepartmentId(departmentId: String): Boolean
}

package learning.diplom.document.svc.service

import com.mongodb.client.gridfs.model.GridFSFile
import learning.diplom.document.svc.model.Document
import learning.diplom.document.svc.util.JwtUtil
import org.bson.types.Binary
import org.slf4j.LoggerFactory
import org.springframework.data.mongodb.core.MongoOperations
import org.springframework.data.mongodb.core.query.Criteria
import org.springframework.data.mongodb.core.query.Query
import org.springframework.data.mongodb.gridfs.GridFsResource
import org.springframework.data.mongodb.gridfs.GridFsTemplate
import org.springframework.http.HttpHeaders
import org.springframework.stereotype.Service
import org.springframework.web.multipart.MultipartFile
import reactor.kotlin.core.publisher.zip
import java.io.ByteArrayOutputStream
import java.io.FileInputStream
import java.util.zip.ZipEntry
import java.util.zip.ZipOutputStream
import javax.servlet.http.HttpServletResponse
import javax.ws.rs.core.MediaType

@Service
class DocumentService(
    private val gridFsTemplate: GridFsTemplate,
    private val jwtUtil: JwtUtil,
    private val mongoOperations: MongoOperations,
    private val signService: SignService
) {

    //UPDATE

    fun uploadDocument(file: MultipartFile, redisKey: String?) {
        val claims = jwtUtil.getClaims(redisKey)!!
```

```
        val document = Document(
            userId = claims["userId"].toString(),
            fileName = file.name,
            contentType = file.contentType,
            size = file.size,
            departmentId = claims["departmentId"].toString()
        )

        gridFsTemplate.store(file.inputStream, file.originalFilename,
file.contentType, document)
        LOG.info("Document = {} stored", document)
    }

    fun downloadDocument(id: String, response: HttpServletResponse) {
        val gridFsResource = getGridFsResourceById(id)
        val gridFsFile = getGridFsFileById(id)

        if (gridFsFile.metadata!!["signatureExists"] == true) {
            val headerKey = HttpHeaders.CONTENT_DISPOSITION
            val headerValue = "attachment; filename=\"zipFile.zip\""
            response.setHeader(headerKey, headerValue)

            LOG.info("Download zip")
            val byteArrayOutputStream = ByteArrayOutputStream()
            val zipOutputStream =
ZipOutputStream(byteArrayOutputStream)
            val gridFsResourceInputStream = gridFsResource.content
            // put document to zip

zipOutputStream.putNextEntry(ZipEntry(gridFsFile.filename))

zipOutputStream.write(gridFsResourceInputStream.readAllBytes())
            zipOutputStream.closeEntry()
            gridFsResourceInputStream.close()

            // put signature to zip
            zipOutputStream.putNextEntry(

ZipEntry("department_${gridFsFile.metadata!!["departmentId"].toString
()}_signature")
            )
            zipOutputStream.write((gridFsFile.metadata!!["signature"]
as Binary).data)
            zipOutputStream.closeEntry()

            // put publicKey to zip
            zipOutputStream.putNextEntry(

ZipEntry("department_${gridFsFile.metadata!!["departmentId"].toString
()}_publicKey.key")
            )
            zipOutputStream.write((gridFsFile.metadata!!["publicKey"]
as Binary).data)
            zipOutputStream.closeEntry()

            zipOutputStream.flush()
            zipOutputStream.close()

            response.contentType = "application/zip"
            val outputStream = response.outputStream
            outputStream.write(byteArrayOutputStream.toByteArray())
```

```
            outputStream.flush()
            outputStream.close()
        } else {
            val headerKey = HttpHeaders.CONTENT_DISPOSITION
            val headerValue = "attachment;
filename=\"${gridFsFile.filename}\""
            response.setHeader(headerKey, headerValue)

            LOG.info("Download file")
            response.contentType = MediaType.APPLICATION_OCTET_STREAM
            val outputStream = response.outputStream
            outputStream.write(gridFsResource.content.readAllBytes())
            outputStream.close()
        }
    }

    fun approveDocument(redisKey: String?, documentId: String?):
Boolean {
        val claims = jwtUtil.getClaims(redisKey)!!
        val gridFsFile = getGridFsFileById(documentId!!)
        val (_, approveField) =
jwtUtil.getRoleWithApproveFieldFromClaims(claims) ?: return false
        gridFsFile.metadata!![approveField] = true
        mongoOperations.save(gridFsFile, "fs.files")

        signService.signDocumentIfHasConditions(gridFsFile,
redisKey!!)
        return true
    }

    // READ

    fun getAllDocumentsByUserId(redisKey: String): List<GridFSFile> {
        val userId =
jwtUtil.getClaims(redisKey)!!["userId"].toString()
        LOG.info("Get all documents for departmentId = {}", userId)

        val fileList: MutableList<GridFSFile> = mutableListOf()
        gridFsTemplate.find(
            Query.query(
                Criteria.where("metadata.userId").`is`(userId)
//                    .and("metadata.signatureExists").`is`(false)
            )
        ).into(fileList)

        LOG.info("Found = {}", fileList)
        return fileList
    }

    fun getGridFsResourceById(fileId: String): GridFsResource {
        LOG.info("Find gridFsResource by fileId = {}", fileId)
        val gridFsFile = getGridFsFileById(fileId)
        return gridFsTemplate.getResource(gridFsFile)
    }

    fun getGridFsFileById(fileId: String): GridFSFile {
        LOG.info("Find gridFsFile by fileId = {}", fileId)

        return gridFsTemplate.findOne(
            Query.query(
                Criteria.where("_id").`is`(fileId)
            )
```

```
        )
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(DocumentService::class.java)
    }
}

package learning.diplom.document.svc.service

import com.mongodb.client.gridfs.GridFSBucket
import com.mongodb.client.gridfs.model.GridFSFile
import learning.diplom.document.svc.repository.SignRepository
import learning.diplom.document.svc.util.JwtUtil
import org.slf4j.LoggerFactory
import org.springframework.data.mongodb.core.MongoOperations
import org.springframework.data.mongodb.core.query.Criteria
import org.springframework.data.mongodb.core.query.Query
import org.springframework.data.mongodb.gridfs.GridFsResource
import org.springframework.data.mongodb.gridfs.GridFsTemplate
import org.springframework.stereotype.Service
import org.springframework.web.multipart.MultipartFile
import reactor.kotlin.core.util.function.*
import reactor.util.function.Tuple5
import reactor.util.function.Tuples
import java.io.*
import java.math.BigInteger
import java.security.*
import java.security.spec.PKCS8EncodedKeySpec
import java.security.spec.RSAPrivateKeySpec
import java.security.spec.RSAPublicKeySpec
import java.security.spec.X509EncodedKeySpec
import kotlin.math.sign

@Service
class SignService(
    private val signRepository: SignRepository,
    private val gridFsTemplate: GridFsTemplate,
    private val jwtUtil: JwtUtil,
    private val gridFsBucket: GridFSBucket,
    private val mongoOperations: MongoOperations
) {
    // WRITE

    fun generateDsaKeyPair(departmentId: String) {
        val keyPairGen = KeyPairGenerator.getInstance("DSA")
        val random: SecureRandom =
SecureRandom.getInstance("SHA1PRNG")
        keyPairGen.initialize(2048, random)
        val keyPair = keyPairGen.genKeyPair()
        val fact = KeyFactory.getInstance("DSA")
        saveKeyPairToFiles(keyPair, departmentId)

        val privateFileName = "${departmentId}_private.key"
        val publicFileName = "${departmentId}_public.key"

        val privateFile = File(privateFileName)
        val publicFile = File(publicFileName)

        gridFsTemplate.store(privateFile.inputStream(),
privateFileName)
```

```
        gridFsTemplate.store(publicFile.inputStream(),
publicFileName)
        File(privateFileName).delete()
        File(publicFileName).delete()
        println()
    }

    private fun saveKeyPairToFiles(keyPair: KeyPair, departmentId:
String) {
        val privateKey = keyPair.private
        val publicKey = keyPair.public

        // Store Public Key.
        val x509EncodedKeySpec = X509EncodedKeySpec(
            publicKey.encoded
        )
        var fos = FileOutputStream("${departmentId}_public.key")
        fos.write(x509EncodedKeySpec.encoded)
        fos.close()

        // Store Private Key.
        val pkcs8EncodedKeySpec = PKCS8EncodedKeySpec(
            privateKey.encoded
        )
        fos = FileOutputStream("${departmentId}_private.key")
        fos.write(pkcs8EncodedKeySpec.encoded)
        fos.close()
    }

    private fun loadKeyPair(filename: String, algorithm: String):
KeyPair {
        // Read Public Key.
        val filePublicKey = File(filename)
        var fis = FileInputStream(filename)
        val encodedPublicKey =
ByteArray(filePublicKey.length().toInt())
        fis.read(encodedPublicKey)
        fis.close()

        // Read Private Key.
        val filePrivateKey = File(filename)
        fis = FileInputStream(filename)
        val encodedPrivateKey =
ByteArray(filePrivateKey.length().toInt())
        fis.read(encodedPrivateKey)
        fis.close()

        // Generate KeyPair.
        val keyFactory = KeyFactory.getInstance(algorithm)
        val publicKeySpec = X509EncodedKeySpec(
            encodedPublicKey
        )
        val publicKey = keyFactory.generatePublic(publicKeySpec)

        val privateKeySpec = PKCS8EncodedKeySpec(
            encodedPrivateKey
        )
        val privateKey = keyFactory.generatePrivate(privateKeySpec)

        return KeyPair(publicKey, privateKey)
    }
```

```
    fun generateRsaKeyPair(departmentId: String) {
        val keyPairGen = KeyPairGenerator.getInstance("RSA")
        val random: SecureRandom =
SecureRandom.getInstance("SHA1PRNG")
        keyPairGen.initialize(2048, random)
        val keyPair = keyPairGen.genKeyPair()
        val fact = KeyFactory.getInstance("RSA")

        val publicKey = fact.getKeySpec(
            keyPair.public,
            RSAPublicKeySpec::class.java
        )

        val privateKey = fact.getKeySpec(
            keyPair.private,
            RSAPrivateKeySpec::class.java
        )

        val privateFileName = "${departmentId}_private.txt"
        val publicFileName = "${departmentId}_public.txt"

        saveToFile(privateFileName, privateKey.modulus,
privateKey.privateExponent)
        saveToFile(publicFileName, publicKey.modulus,
publicKey.publicExponent)

        val privateFile = File(privateFileName)
        val publicFile = File(publicFileName)


        gridFsTemplate.store(privateFile.inputStream(),
privateFileName)
        gridFsTemplate.store(publicFile.inputStream(),
publicFileName)
        File(privateFileName).delete()
        File(publicFileName).delete()
        println()
    }

    fun signDocumentIfHasConditions(gridFsFile: GridFSFile, redisKey:
String) {
        val (headOfAdministrationApprove,
            deputyHeadOfAdministrationApprove,
            headOfDepartmentApprove,
            deputyHeadOfDepartmentApprove,
            employeeApprove) = gridFsFile.getApproveFields()

        val privateKey = getPrivateKey(redisKey)
        val publicKey = getPublicKey(redisKey)
        val gridFsResource = gridFsTemplate.getResource(gridFsFile)

        if (headOfDepartmentApprove
            || (deputyHeadOfDepartmentApprove && employeeApprove)
        ) {
            signDocument(gridFsFile, gridFsResource, privateKey,
publicKey)
        }
    }

    fun signDocument(
        gridFsFile: GridFSFile,
        gridFsResource: GridFsResource,
```

```
        privateKey: PrivateKey,
        publicKey: PublicKey
    ) {
        LOG.info("Sign document{} ", gridFsFile.filename)

        val signature = Signature.getInstance(privateKey.algorithm)
        signature.initSign(privateKey)
        signature.update(gridFsResource.content.readAllBytes())
        val completedSignature = signature.sign()

        gridFsFile.metadata!!["signature"] = completedSignature
        gridFsFile.metadata!!["signatureExists"] = true
        gridFsFile.metadata!!["publicKey"] = publicKey.encoded

        mongoOperations.save(gridFsFile, "fs.files")
    }

    fun verifySignature(publicKey: MultipartFile, signature:
MultipartFile, document: MultipartFile): Boolean {
        var verifies: Boolean
        try {
            val publicKeyFromFile =
readDsaPublicKey(publicKey.inputStream)

            val signatureByPublicKey =
Signature.getInstance("SHA1withDSA")
            signatureByPublicKey.initVerify(publicKeyFromFile)

signatureByPublicKey.update(document.inputStream.readAllBytes())

            verifies =
signatureByPublicKey.verify(signature.inputStream.readAllBytes())

            println("signature verifies: $verifies")
        } catch (e: Exception) {
            verifies = false
        }


        document.inputStream.close()
        signature.inputStream.close()
        publicKey.inputStream.close()
        return verifies
    }

    // READ

    fun getPrivateKey(redisKey: String): PrivateKey {
        val claims = jwtUtil.getClaims(redisKey)!!
        val departmentId = claims["departmentId"].toString()

        return findPrivateKey(departmentId)
    }

    fun getPublicKey(redisKey: String): PublicKey {
        val claims = jwtUtil.getClaims(redisKey)!!
        val departmentId = claims["departmentId"].toString()

        return findPublicKey(departmentId)
    }

    private fun saveToFile(
```

```
        fileName: String,
        mod: BigInteger, exp: BigInteger
    ) {
        val oout = ObjectOutputStream(
            BufferedOutputStream(FileOutputStream(fileName))
        )
        try {
            oout.writeObject(mod)
            oout.writeObject(exp)
        } catch (e: Exception) {
            throw InternalError()
        } finally {
            oout.close()
        }
    }

    private fun findPrivateKey(departmentId: String): PrivateKey {
        val savedPrivateKey = gridFsTemplate.findOne(
            Query.query(

Criteria.where("filename").`is`("${departmentId}_private.key")
            )
        )

        val downloadStream =
gridFsBucket.openDownloadStream("${departmentId}_private.key")
        val gridFsResource = GridFsResource(savedPrivateKey,
downloadStream)

        return readDsaPrivateKey(gridFsResource.inputStream)
    }

    private fun findPublicKey(departmentId: String): PublicKey {
        val savedPrivateKey = gridFsTemplate.findOne(
            Query.query(

Criteria.where("filename").`is`("${departmentId}_public.key")
            )
        )

        val downloadStream =
gridFsBucket.openDownloadStream("${departmentId}_public.key")
        val gridFsResource = GridFsResource(savedPrivateKey,
downloadStream)

        return readDsaPublicKey(gridFsResource.inputStream)
    }

    private fun readDsaPrivateKey(inputStream: InputStream):
PrivateKey {
        val keyFactory = KeyFactory.getInstance("DSA")
        val privateKeySpec = PKCS8EncodedKeySpec(
            inputStream.readAllBytes()
        )
        return keyFactory.generatePrivate(privateKeySpec)
    }

    private fun readDsaPublicKey(inputStream: InputStream): PublicKey
{
        val keyFactory = KeyFactory.getInstance("DSA")
        val publicKeySpec = X509EncodedKeySpec(
            inputStream.readAllBytes()
```

```
        )
        return keyFactory.generatePublic(publicKeySpec)
    }

    // RSA KEYS
//    private fun readRsaPrivateKey(fileName: String): PrivateKey {
//        val `in`: InputStream = FileInputStream(fileName)
//        val oin = ObjectInputStream(BufferedInputStream(`in`))
//        return getRsaPrivateKeyFromInputStream(oin)
//    }
//
//    private fun readRsaPublicKey(fileName: String): PublicKey {
//        val `in`: InputStream = FileInputStream(fileName)
//        val oin = ObjectInputStream(BufferedInputStream(`in`))
//        return getRsaPublicKeyFromInputStream(oin)
//    }
//    private fun readRsaPrivateKey(inputStream: InputStream):
PrivateKey {
//        val oin =
ObjectInputStream(BufferedInputStream(inputStream))
//        return getRsaPrivateKeyFromInputStream(oin)
//    }
//
//    private fun readRsaPublicKey(inputStream: InputStream):
PublicKey {
//        val oin =
ObjectInputStream(BufferedInputStream(inputStream))
//        return getRsaPublicKeyFromInputStream(oin)
//    }
//
//    private fun getRsaPublicKeyFromInputStream(oin:
ObjectInputStream): PublicKey {
//        return try {
//            val m = oin.readObject() as BigInteger
//            val e = oin.readObject() as BigInteger
//            val keySpec = RSAPublicKeySpec(m, e)
//            val fact = KeyFactory.getInstance("RSA")
//            fact.generatePublic(keySpec)
//        } catch (e: java.lang.Exception) {
//            throw InternalError()
//        } finally {
//            oin.close()
//        }
//    }
//
//    private fun getRsaPrivateKeyFromInputStream(oin:
ObjectInputStream): PrivateKey {
//        return try {
//            val m = oin.readObject() as BigInteger
//            val e = oin.readObject() as BigInteger
//            val keySpec = RSAPrivateKeySpec(m, e)
//            val fact = KeyFactory.getInstance("RSA")
//            fact.generatePrivate(keySpec)
//        } catch (e: java.lang.Exception) {
//            throw InternalError()
//        } finally {
//            oin.close()
//        }
//    }

    private fun GridFSFile.getApproveFields(): Tuple5<Boolean,
Boolean, Boolean, Boolean, Boolean> {
```

```
        return with(this.metadata!!) {
            Tuples.of(
                this["headOfAdministrationApprove"] as Boolean,
                this["deputyHeadOfAdministrationApprove"] as Boolean,
                this["headOfDepartmentApprove"] as Boolean,
                this["deputyHeadOfDepartmentApprove"] as Boolean,
                this["employeeApprove"] as Boolean
            )
        }

    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(SignService::class.java)
    }
}
package learning.diplom.document.svc.util

import io.jsonwebtoken.Claims
import io.jsonwebtoken.Jwts
import learning.diplom.document.svc.model.Role
import learning.diplom.model.error.lib.ServerEntity
import
learning.diplom.model.error.lib.exception.rest.EntityNotFoundRestExce
ption
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Value
import org.springframework.data.redis.core.RedisTemplate
import org.springframework.stereotype.Component

@Component
class JwtUtil(
    @Value("\${jwt.secret}")
    private val jwtSecret: String? = null,
    @Value("\${jwt.validity}")
    private val tokenValidity: Long = 0,
    @Value("\${jwt.info}")
    private val info: String? = null,
    private val redisTemplate: RedisTemplate<String, Any>
) {

    fun getClaims(redisKey: String?): Claims? {
        val token = getJwtToken(redisKey)
        if (redisKey.isNullOrBlank() || token == null) {
            LOG.error("Token did not find")
            throw EntityNotFoundRestException(ServerEntity.TOKEN,
"redisKey = $redisKey")
        }
        try {
            return
Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token.toString(
)).body
        } catch (e: Exception) {
            println(e.message + " => " + e)
        }
        return null
    }

    fun getRoleWithApproveFieldFromClaims(claims: Claims): Pair<Role,
String>?{
        return claims["role"].toString().mapClaimRole()
```

```
    }

    fun getJwtToken(redisKey: String?): Any? {
        return redisTemplate.opsForValue().get(redisKey ?: "empty")
    }

    private fun String.mapClaimRole(): Pair<Role, String>? {
        return when {
            equals("HEAD_OF_ADMINISTRATION") ->
Role.HEAD_OF_ADMINISTRATION to "headOfAdministrationApprove"
            equals("DEPUTY_HEAD_OF_ADMINISTRATION") ->
Role.DEPUTY_HEAD_OF_ADMINISTRATION to
"deputyHeadOfAdministrationApprove"
            equals("HEAD_OF_DEPARTMENT") -> Role.HEAD_OF_DEPARTMENT
to "headOfDepartmentApprove"
            equals("DEPUTY_HEAD_OF_DEPARTMENT") ->
Role.DEPUTY_HEAD_OF_DEPARTMENT to "deputyHeadOfDepartmentApprove"
            equals("EMPLOYEE") -> Role.EMPLOYEE to "employeeApprove"

            else -> null
        }
    }

    companion object {
        private val LOG =
LoggerFactory.getLogger(JwtUtil::class.java)
    }

}

package learning.diplom.document.svc

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import
org.springframework.cloud.client.discovery.EnableDiscoveryClient
import org.springframework.data.mongodb.config.EnableMongoAuditing

@SpringBootApplication
@EnableMongoAuditing
@EnableDiscoveryClient
class DocumentSvcApplication

fun main(args: Array<String>) {
    runApplication<DocumentSvcApplication>(*args)
}

<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
          crossorigin="anonymous"></script>
  <meta charset="UTF-8">
```

```
  <title>Approve document</title>
</head>
<body>
<div th:replace="blocks/header :: header"></div>

<div th:object="${result}">

  <td th:text="${result ?: false} ? 'SUCCESS' : 'ERROR'"></td>
  <hr>
  <a th:href="@{/document/list}">Redirect to document list</a>
</div>
</body>
</html>

<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
          crossorigin="anonymous"></script>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<div th:replace="blocks/header :: header"></div>

<div class="container">

  <div th:each="document: ${documentList}">
    <form class="document" method="POST"
th:action="@{/document/download-document}"
th:object="${documentRequest}">
      <h3 th:text="${document.getFilename()}"></h3>
      <h3 th:if="${document.getMetadata().get('signatureExists') ==
true }"> (SIGNED) </h3>

      <label>
        <input type="hidden" name="id" placeholder="ID" class="form-
control"
               th:value="${document.getId().getValue()}">
      </label>
      <button type="submit" name="submit" value="value" class="link-
button">DOWNLOAD</button>
    </form>

    <form class="document" method="POST"
th:action="@{/document/approve-document}"
th:object="${documentRequest}">

      <label>
        <input type="hidden" name="id" placeholder="ID" class="form-
control"
               th:value="${document.getId().getValue()}">
      </label>
```

```
      <button type="submit" name="submit" value="value" class="link-
button">APPROVE</button>
    </form>
    <hr>
  </div>

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

<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Download document</title>
</head>
<body>
SUCCESS
</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>

<h1>Keys PAGE</h1>

<div class="container">

  <div th:object="${privateKey}">
    <h3  th:text="${privateKey}"></h3>

    <!--    <a th:href="@{/users/{id}/edit(id
=${userInfo.getId()})}">Edit</a>-->
    <hr>
  </div>

  <div th:object="${publicKey}">
    <h3  th:text="${publicKey}"></h3>

    <!--    <a th:href="@{/users/{id}/edit(id
=${userInfo.getId()})}">Edit</a>-->
    <hr>
  </div>
</div>


</body>
</html>

<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Spring Boot & Thymeleaf File Upload</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstra
```

```
p.min.css">
  <meta charset="UTF-8">
</head>
<body>

<h2>Upload File Example</h2>
<form method="post" th:action="@{/document/upload-document}"
enctype="multipart/form-data">
  <div class="form-group">
    <input type="file" name="file" class="form-control-file">
  </div>
  <button type="submit" class="btn btn-primary">Upload File</button>
</form>
</body>
</html>


<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
          crossorigin="anonymous"></script>
  <meta charset="UTF-8">
  <title>VERIFY RESULT</title>
</head>
<body>

<div th:replace="blocks/header :: header"></div>

<div th:object="${result}">
  <h3>VERIFY RESULT</h3>
  <td th:text="${result ?: false} ? 'SUCCESS' : 'ERROR'"></td>
  <hr>
  <a th:href="@{/sign/verify}">Redirect verify document page</a>
</div>
</body>
</html>

<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstra
p.min.css">
</head>
<body>

<h2>Verify signature</h2>
<form method="post" th:action="@{/sign/verify}"
enctype="multipart/form-data">
```

```
  <label for="document">Choose document</label>
  <div class="form-group">
    <input id="document" type="file" name="document" class="form-
control-file">
  </div>
  <hr>

  <label for="signature">Choose signature</label>
  <div class="form-group">
    <input id="signature" type="file" name="signature" class="form-
control-file">
  </div>
  <hr>

  <label for="publicKey">Choose public key</label>
  <div class="form-group">
    <input id="publicKey" type="file" name="publicKey" class="form-
control-file">
  </div>
  <hr>

  <button type="submit" class="btn btn-primary">Upload File</button>
</form>

</body>
</html>

spring.servlet.multipart.file-size-threshold=50MB
#spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-request-size=50MB
spring.servlet.multipart.max-file-size=50MB

spring.webflux.multipart.max-in-memory-size=50MB
spring.webflux.multipart.max-headers-size=50MB

server:
  port: 8007

spring:
  application:
    name: document-svc
  cloud:
    config:
      username: configUser
      password: configPassword
      fail-fast: true
      discovery:
        enabled: true
        service-id: cloud-config

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8002/eureka
    fetch-registry: true
  instance:
    lease-renewal-interval-in-seconds: 10
```