МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

РОГОЖУК НАЗАР ВЯЧЕСЛАВОВИЧ

Допускається до захисту:
Завідувач кафедри інформаційних технологій,
кандидат технічних наук, доцент
_____ Т.В. Нескородева
«__» _____ 20__р.

РОЗРОБКА СЕРВІСУ ДЛЯ ОБРОБКИ ВІДЕО З ФУНКЦІЄЮ ГОРИЗОНТАЛЬНОГО МАСШТАБУВАННЯ ТА ПРІОРИТЕЗАЦІЇ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Бабаков Р. М., доцент
кафедри інформаційних технологій

Оцінка:____/____/_____
(бали за шкалою СКТС/за національною шкалою)

Голова ЕК:_____
(підпис)

Вінниця - 2021

АНОТАЦІЯ

Рогожук Н. В. Розробка сервісу для обробки відео з функцією горизонтального масштабування та пріоритезації. Спеціальність 122 «Комп'ютерні науки», спеціалізація «Інформаційні технології»
Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній (бакалаврській) роботі досліджені сучасні методи побудови мікросервісної архітектури для обробки та аналізу відео. Розгляд аналогів, пошук та вивчення представлених веб-джерел по роботі з відео. Розробка додатку, котрий працює в режимі реального часу. Розробка клієнт-серверного додатку з ціллю надання зручного, гнучкого рішення для роботи з відео.

Ключові слова: обробка відео, аналіз, архітектура, сучасні технології, розробка.

51 с., 12 рис., 3 дод., 27 джерел.

ABSTRACT

Rohozhuk NV Development of a service for video processing with the function of horizontal scaling and prioritization. Specialty 122 "Computer Science", specialization "Information Technology". Vasyl Stus Donetsk National University, Vinnytsia, 2021.

In the qualification (bachelor's) work the modern methods of construction of microservice architecture for video processing and analysis are investigated. Review of analogues, search and study of the presented web sources on work with video. Development of an application that works in real time. Development of a client-server application to provide a convenient, flexible solution for working with video.

Key words: video processing, analysis, architecture, modern technologies, development.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

JS – JavaScript (Мова програмування JavaScript);

DOM – Document Object Model (Об’єкт модель документа)

API – Application programming interface (Програмний інтерфейс додатку)

JSON – JavaScript Object Notation (Запис об’єктів JavaScript)

AMQP – протокол для передачі повідомлень між компонентами системи (Advanced Message Queuing Protocol)

NPM – Node Packet Manager (Менеджер пакетів Node.js)

JSON – JavaScript Object Notation (Запис об’єктів JavaScript)

ВСТУП

Відео контент складає вагому частину у мережі Інтернет, новини, розважальний контент, фільми, серіали, навчальні курси – основні галузі де основним засобом доставки є потокова передача відео. При цьому кількість пристроїв з яких відбувається споживання контенту також дуже велика, це мобільні телефону, ноутбуки, планшети, телевізори, у всіх пристроїв різні потужності та підтримувані формати. Тому дуже важливо щоб відео працювало на всіх пристроях.

Коли йде мова про створення сервісу, основною метою якою буде завантаження, аналіз, обробка та безпосередньо доставка, необхідно розуміти що навантаження буде дуже велике, і потрібно заздалегідь будувати сервіс, який легко буде масштабуватись.

Тема бакалаврської роботи – «Розробка сервісу для обробки відео з функцією горизонтального масштабування та пріоритезації». Основною метою роботи є:

- Огляд існуючих аналогів
- Ознайомлення з роботою програмного забезпечення на протоколі AMQP для комунікації між сервісами
- Ознайомлення з особливостями бази даних PostgreSQL
- Побудова архітектури, яка буде легко масштабуватись горизонтально
- Побудова мікросервісної архітектури
- Розробка front-end частини для безпосередньої роботи з відео
- Тестування

Також потрібно забезпечити для користувачів максимально зрозумілий інтерфейс, де буде зрозуміло на якому етапі знаходить обробка відео, і зрозумілі помилки, якщо вони виникли. Кінцевий продукт має бути універсальним і надавати можливість вбудови відео в будь який мобільний додаток чи веб-сайт. При цьому витримувати велику завантаженість.

Розділ 1.

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

1.1. Актуальність теми

Розповсюдженість відеоконтенту важко переоцінити, адже він майже всюди, на будь якому ресурсі, мобільному додатку, навчальній платформі. Перегляд фільму з онлайн кінотеатру, перегляд навчального контенту, перегляд роликів для дозвілля. І за таким речами стоять потужні, важкі системи та сервіси.

Також можна зазначити, що з популяризацією мережі Інтернет, збільшенням пропускної здатності просто зникає потреба у тому щоб йти у магазин купувати диск з фільмом, чи завантажувати на носій. Потокове відео вирішує усі проблеми та незручності, при цьому дає змогу насолоджуватись стрічкою у максимальній якості. Напевно першим популяризатором такого підходу був Netflix. Це велика компанія, яка надавала послуги по оренді дисків через звичайну пошту. Але з часом вона стала величезною розважальною компанією, яка випускає свої високоякісні бюджетні свої фільми та серіали, та дає змогу переглядати їх на своєму сервісі. При цьому надаючи надзвичайно зручний інтерфейс та будуючи розумні рекомендації виходячи з глядацького досвіду користувача.

Отже розробка подібного сервісу для роботи з відео, є як ніколи актуальним.

1.2. Розгляд аналогів

1.2.1. Netflix

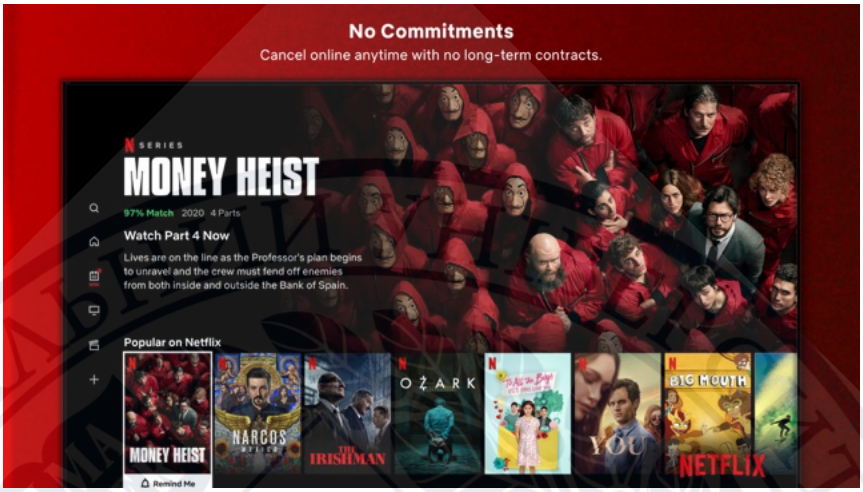


Рисунок 1.1 - огляд Netflix

Розпочнемо з вище згаданого сервісу Netflix. Netflix – американська розважальна компанія, постачальник фільмів та серіалів на основі потокового мультимедіа. Заснована 29 серпня 1997 року Рідом Хастінгсом та Марком Рендельфом. Штаб квартира знаходиться в Каліфорнії.

Бібліотека Netflix доступна для перегляду на більшості сучасних інтернет-браузерів, також існують додатки для телефонів, планшетів та телевізорів.

Одною з особливостей Netflix стосовно побудови структури свого контенту це те, що фільм чи серіал захвачує глядача з перших секунд. Також сервіс має дуже просунуту систему рекомендацій, яка у свій час, ще у еру оренди дисків та касет була найпершої в своєму роді. Тому за рахунок цього у Netflix дуже велика клієнтська база.

1.2.2. Youtube

Рисинок 1.2 – огляд YouTube

YouTube – найпопулярніший і найбільший відеохостинг в світі. Завдяки якому, користувачі зберігають свої відео, дивляться, коментують і діляться з іншими користувачами.

Ресурс містить в собі професійні, художні, музичні, розважальні відео, та багато іншого. Також завдяки сервісу контент мейкери мають змогу монетизувати свої роботи. Кількість користувачів сервісу це мільярди користувачів та десятки мільйонів одночасних користувачів. Щоб забезпечувати стабільну роботу в таких умовах потрібно великі потужності та правильно побудована архітектура.

1.2.3. TikTok

Рисунок 1.3 – огляд TikTok

TikTok - соцмедійний застосунок для створення та поширення відеофайлів та онлайн-трансляцій. Сервіс запущено у вересні 2016 китайською компанією ByteDance. Це найпопулярніша платформа для коротких відео в Азії, яка поширилася на інші частини світу і швидко набирає популярність. Застосунок дозволяє користувачам створювати музичні, танцювальні, комедійні та інші відео тривалістю до 15 секунд. Кількість користувачів програми сягнула 1 млрд з 170 країн.

Мобільний додаток TikTok дозволяє користувачам переглядати музичні відео, створювати lip-sync до пісень, знімати короткі відеокліпи з можливістю їхнього редагування з вбудованими ефектами, фільтрами та стикерами. Аби створити музичне відео у програмі, користувачі можуть обрати фонову музику з великої кількості музичних напрямків, включаючи хіп-хоп, попмузику або електронну музику, і записати 15-секундне відео перед вивантаженням та поширенням з іншим людьми.

Додаток надає користувачам можливість виставити свої акаунти у приватний режим, дозволивши перегляд контенту лише вибраним користувачам. Користувачі також можуть дозволити всім або лише друзям надсилання їм коментарі або повідомлення, а також «duet» з чи «react» на їхні відеофайли.

Функція «duet» дозволяє користувачам створювати нові відео поряд з вже таким, що існує, а функція «react» — поміщати нове відео у менший фрейм, який можна перетягнути на вже записаний відеофайл.

1.2.4. Twitch

Рисунок 1.4 – огляд Twitch

Twitch - це платформа для онлайн відеотрансляцій, що належить Twitch Interactive, дочірній компанії Amazon.com. Представлений в червні 2011 р. як відгалуження від платформи відеотрансляцій загального поширення, Justin.tv, сайт, в першу чергу, зосереджується на потоковій трансляції відеоігор, включаючи трансляції кіберспортивних змагань, креативного контенту, розділ «реальне життя» та останнім часом музичні передачі. Вміст на сайті можна переглянути в прямому ефірі або за допомогою відео за запитом.

Метою сайту є забезпечення максимального зв'язку між аудиторією та гравцем. Цей зв'язок забезпечує «живий» чат, який дає змогу моментально коментувати події і таким чином донести до гравця враження та думки глядачів. Крім того, на сайті існують платні підписки та повідомлення, які мають виділення у чаті. За допомогою цих особливостей у глядачів складається враження присутності поруч із гравцем та вплив на ігровий процес.

1.3. Постановка задачі

Мета бакалаврської роботи полягає в побудові сервісу для аналізу, обробки та роздачі відео контенту. Основною ціллю є створення саме проекту з використанням мікросервісної архітектури для легкого горизонтального масштабування.

Для досягнення поставленої мети потрібно:

- Проаналізувати існуючі сервіси, котрі надають доступ до потокового відео
- Визначити основні принципи мікросервісної архітектури
- Обрати брокер повідомлень
- Побудувати архітектуру сервісу
- Побудувати схему бази даних
- Розробити контракти комунікації між сервісами та між кінцевими користувачами
- Розробка основного gateway сервісу, з допомогою якого будуть завантажуватись відео, та публікуватись задачі для обробки відео іншими сервісами, роботою з базою даних та надаватись зручний GraphQL API для комунікації з клієнтською частиною.

Розділ 2.
АНАЛІЗ ТА ВИБІР АКТУАЛЬНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Вибір програмних засобів розробки

В даному розділі розглянуто програмні засоби та технології, які знадобились для розробки сервісу, також розглянуто архітектурні підходи для побудови мікросервісної архітектури.

2.1.1. Мікросервісна архітектура, брокери повідомлень, масштабування

Для початку слід визначити основні поняття, що стосуються мікросервісної архітектури

Брокер повідомлень (або диспетчер черги) - це програмний застосунок, який приймає і віддає сполучення між окремими модулями / додатками всередині деякої складної системи, де модулі / додатки повинні спілкуватися між собою - тобто пересилати дані один одному.

Розподілена система - така система, яка працює відразу на безлічі машин, що утворюють цілісний кластер. Кластер це набір комп'ютерів / серверів, об'єднаних мережею і взаємодіючих між собою. Найважливіші плюси такого підходу - високодоступних і відмовостійкість.

Гарантія доставка – підхід коли повідомлення попадає до брокера, і не зникає з черги доки сервіс не відповість, що повідомлення доставлене та оброблене.

Вертикальна масштабованість - це нарощування ресурсів, тобто збільшення кількості ядер, оперативної пам'яті і т.д. на одному сервері.

Переваги:

- досить просто і зрозуміло використовувати

Недоліки:

- не можна нарощувати нескінченно.
- при додаванні ресурсів (оперативна пам'ять і т.д.) зазвичай доводиться вимикати сервер, що погано впливає на відмовостійкість.

Горизонтальна масштабованість - це додавання нових серверів з будь-якими характеристиками в обчислювальний кластер.

Переваги:

- Немає таких проблем, як у вертикальній масштабованості

Недоліки:

- Не скрізь підтримується горизонтальна масштабованість
- Не всі системи працюють в кластерах, а ті, які в них працюють, зазвичай досить складні в експлуатації

Наприклад більшість баз даних неможливо або досить важко налаштувати під роботу у кластерному режимі.

Далі наведено основні брокери повідомлень, визначимо переваги та недоліки, та оберемо, який буде використовуватись у сервісі обробки відео.

Apache Kafka - розподілений горизонтально масштабований відмовостійкий програмний брокер повідомлень.

Додатки (генератори) відправляють повідомлення (записи) на вузол Kafka (брокер), і повідомлення про прийняття обробляються іншими додатками, так званими споживачами. Зазначені повідомлення зберігаються, а споживачі підписуються для отримання нових повідомлень. Kafka гарантує, що всі повідомлення будуть упорядковані саме в тій послідовності, в якій надійшли. Kafka не відслідковує, які записи зчитуються споживачем і після цього видаляються, а просто зберігає їх протягом заданого періоду часу. Споживачі

самі опитують Kafka, чи не з'явилося у нього нових повідомлень, і вказують, які записи їм потрібно прочитати.

Цей підхід схожий на бібліотеку з читальним залом, коли хто завгодно, бере книгу (повідомлення), читає її в читальному залі, потім віддає назад. А книги, коли застарівають, просто викидаються з бібліотеки.

RabbitMQ, як і Kafka, теж розподілений горизонтально масштабований відмовостійкий програмний брокер повідомлень.

Додатки (publishers) надсилають повідомлення на вузол RabbitMQ (exchange), при цьому RabbitMQ відсилає назад додаткам підтвердження, що повідомлення отримано. Одержувачі (consumers) завжди пов'язані з RabbitMQ по TCP і чекають, коли RabbitMQ проштовхне (push) їм повідомлення. Одержувачі підтверджують отримання повідомлення або повідомляють про невдачу. Якщо доставка невдала, то RabbitMQ проштовхує повідомлення до тих пір, поки воно не буде доставлене. Після успішної доставки повідомлення видаляється з черги.

Цей підхід можна порівняти з поштовим відділенням і листоношою, коли посилки (повідомлення) приходять від відправників на пошту, звідти розсилаються по поштовим відділенням, а потім листоноша розносить їх по адресах і переконується, що посилка дійшла до одержувача.

NATS – відносно молодий проект, написаний на мові Go. По замовчуванню має лише можливість Pub-Sub (без гарантії доставки, без черги повідомлень). Проте є налаштування та модулі які додають черги (NATS Streaming, STAN). Та найпотужніше над налаштування Jet Stream, яке включає в себе вище перераховані можливості.

Отже проаналізувавши доступні AMQP вибір впав на RabbitMQ з наступних причин

- Production ready рішення (на відміну від NATS)
- Перевірене та протестоване на великій кількості проектів
- Просте у використання (на відміну від Apache Kafka)
- Багато можливостей на налаштувань з коробки

2.1.2. Golang

Go — компільована мова програмування із вбудованими засобами для паралельних обчислень і засобами віддаленого керування пакунками. Цю мову програмування розробив Google як частину проекту з розробки операційної системи Inferno. Початкова розробка Go почалася у вересні 2007 року, а безпосередньо проектували її Роберт Гризмер, Роб Пайк і Кен Томпсон. Офіційно мову представили у листопаді 2009 року.Підтримка мови здійснюється для операційних систем Linux, Android, Mac OS X та Windows.

Концепція мови Golang розроблена на основі досвіду рішення повсякденних завдань і не переслідує мету зробити прорив в програмуванні. Крім того, Go не реалізує ряд функцій, які роблять інші мови (C++, Java і Python) настільки потужними. Але є три причини, щоб замислитися про використання цієї мови.

Як тільки звик до синтаксису Go, прочитати чужий код - тривіальне завдання. У кожної людини власний "правильний" спосіб робити речі, тому Go нав'язує свій стиль програмування. В результаті питання на зразок використання або невикористання фігурних дужок відпадають, і залишається тільки писати код за заданими правилами.

За короткий проміжок часу можна написати швидку програму. Продуктивність роботи пакету на Go, можливо, буде менше, ніж при використанні C, зате швидкість компіляції для великих програм буде вища. Для більшості проектів - прийнятний компроміс, причому при необхідності досить просто переписати код на мові Go.

Більшість помилок виникають в неперевіреному або складному коді. Go надає зручні засоби для тестування. Крім того, строга типізація усуває помилки

на зразок випадкового порівняння кількості яблук з кількістю груш, які визначаються ще на етапі компіляції.

У практичній частині бакалаврської роботи Go використовується для написання Worker сервіса, сервіс який отримує задачі з RabbitMQ та запускає процес обробки відео.

До особливостей Go можна додати про Go рутини, це дуже зручний механізм роботи з потокам, які виходять дуже легкі, та на запуск час майже не витрачається. Для синхронізації та комунікації з основним потоком, чи іншими потоками використовують, так звані канали.

Go використовують тоді, коли потрібно написати легкий, невеликий додаток. Також Go дуже популярний у Dev-ops середовищі, Docker, Kubernetes, ArgoCD, Flux, та багато іншого – всі ці проекти написані мовою Go.

Нижче наведено фрагмент коду на Go, для ознайомлення. На ньому використовують канали та горутини.

У Go немає, таки званих Exception, якщо потрібно повернути та опрацювати помилку, використовується функція, яка просто повертає додаткове значення з типом error.

```go
if ctx != nil {
        req = req.WithContext(ctx)
}
resp, err := c.client.Do(req)
defer func() {
        if resp != nil {
                resp.Body.Close()
        }
}()

if err != nil {
        return nil, nil, err
}

var body []byte
done := make(chan struct{})
go func() {
        body, err = ioutil.ReadAll(resp.Body)
        close(done)
}()

select {
case <-ctx.Done():
        <-done
        err = resp.Body.Close()
        if err == nil {
                err = ctx.Err()
        }
case <-done:
}

return resp, body, err
```

Рисунок 2.1.2 – приклад коду на Go

2.1.3. PostgreSQL

PostgreSQL - це популярна вільна об'єктно-реляційна система управління базами даних. PostgreSQL базується на мові SQL і підтримує численні можливості.

Переваги:

- Підтримка БД необмеженого розміру
- Потужні та надійні механізми транзакцій та реплікацій
- Розширювана система вбудованих мов програмування і підтримка завантаження С-сумісних модулів
- Велика кількість унікальних функцій та алгоритмів для побудови індексів та запитів

Порівняно з іншими проєктами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.

PostgreSQL підтримує одночасну модифікацію БД декількома користувачами за допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому виконуються вимоги ACID, і практично відпадає потреба в блокуванні зчитування.

2.1.4. Redis

Redis – високопродуктивна key-value база даних, яка в якості сховище використовує оперативну пам’ять, доступ до яких здійснюється за ключем, також Redis можна використовувати як брокер повідомлень. Також є можливість з певним інтервалом записувати дані диск, для того щоб забезпечити надійність при раптових перезапусках. Основна перевага цією БД – швидкість роботи.

Зазвичай використовується для кешу даних, примітивного спілкування між сервісами, коли не потрібна гарантія доставки.

Також в Redis є багато алгоритмів та структур даних для зберігання різних структур. Має велику кількість мов які підтримують взаємодію з БД.

В сервісу обробки відео буде використовуватись саме для взаємодії між екземплярами одного контейнера, при роботі з WebSocket.

Основні типи даних Redis:

- Стрічки
- Списки
- Множини
- Хеш таблиці
- Впорядковані множини

2.1.5. Amazon S3

S3 - сервіс-сховище даних, один з Amazon Web Services. Сервіс надає можливість для зберігання й отримання будь-якого обсягу даних, у будь-який час з будь-якої точки мережі, тобто так званий файловий хостинг.

З допомогою Amazon S3 досягається висока масштабованість, надійність, висока швидкість, недорога інфраструктура зберігання даних. Вперше з'явилася в березні 2006 року в США та в листопаді 2007 року в Європі. Надає гнучкий API для роботи з різними мовами та cli утилітами.

У практичній частині бакалаврської роботи використовується для зберігання оригінальних відео та оброблених, і відповідно для обміну ними.

2.1.6. Minio

Minio - це популярний сервер зберігання об'єктів з відкритим вихідним кодом, сумісний з хмарним сховищем Amazon S3. Додатки, які були налаштовані для зв'язку з Amazon S3, можуть також бути налаштовані для зв'язку з Minio, що дозволяє Minio служити життєздатною альтернативою S3, якщо вам буде потрібно більш ефективний контроль сервера сховища об'єктів. Служба може зберігати неструктуровані дані, такі як фото, відео, файли журналу, резервні копії і образи контейнерів / віртуальних машин, і навіть може надати один сервер зберігання об'єктів, що поєднує в пул безліч дисків, розміщених на різних серверах.

Minio написана на мові Go, має клієнт командного рядка і інтерфейс браузера і підтримує просту службу черги для розширеного протоколу організації черги повідомлень (AMQP), Elasticsearch, Redis, NATS, і PostgreSQL. З усіх цих причин знайомство з налаштуванням сервера зберігання об'єктів Minio може забезпечити більшу гнучкість і практичну користь для вашого проекту.

2.1.7. JavaScript

JavaScript - обєктно-прототипна мова програмування, що здебільшого використовується у веб-розробці. До особливостей можна віднести те, що завдяки бібліотекам та фреймворкам можна реалізувати будь яку архітектуру, завдяки чому мова досить універсальна. Використовується як єдина можливість писати браузерні додатки. Має безліч бібліотек та фреймворків для реалізації майже будь чого. Мова JavaScript використовується для:

- написання сценаріїв вебсторінок для надання їм інтерактивності
- створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js)
- програмування на боці сервера (Node.js(Express.js))
- стаціонарних застосунків (Electron, NW.js)
- мобільних застосунків (React Native, Cordova)
- сценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter)
- всередині PDF-документів тощо.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. В перші роки існування, більшість професійних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-аматорів. Поява AJAX змінила ситуацію та звернула увагу професійної спільноти до мови, а її подальші модифікації за стандартами ES6+ внесли багато корисних можливостей, яких не вистачало для ефективного програмування. В результаті, були розроблені та покращені багато практик використання

JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

Оскільки JavaScript є інтерпретованою мовою програмування, без строгої типізації, і може виконуватися в різних середовищах, кожне зі своїми власними особливостями сумісності, програміст має бути уважним, і повинен перевіряти, що його код виконується як очікується в широкому переліку можливих конфігурацій. Типізація вважається одною з ключових проблем JavaScript, тому восени 2012 року, компанія Microsoft презентувала мову програмування TypeScript, що компілюється в JavaScript та містить декілька важливих для програмістів доповнень, що полегшують розробку.

2.1.8. NodeJS

NodeJS - платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників. В платформі використовується розроблений компанією Google рушій V8.

Додаток Node.js працює в одному процесі, не створюючи нового потоку для кожного запиту. Node.js надає набір асинхронних примітивів вводу-виводу у своїй стандартній бібліотеці, які перешкоджають блокуванню коду JavaScript, і загалом бібліотеки в Node.js записуються з використанням неблокуючих парадигм, що робить поведінку блокування винятком, а не нормою.

У Node.js нові стандарти ECMAScript можна використовувати без проблем, оскільки вам не доведеться чекати, поки всі ваші користувачі оновлять свої браузери - вам належить вирішити, яку версію ECMAScript використовувати, змінивши версію Node.js, а також можна ввімкнути певні експериментальні функції, запустивши Node.js із прапорами.

2.1.9. NestJS

NestJS – потужний фреймворк для побудовий додатків з використанням платформи Node.JS, при створенні автори надихались екосистемою Sping з всесвіту Java та front-end фреймворком Angular.

Має велику кількість вбудованих та адаптованих бібліотек, для роботи з Rest API, GraphQL, реляційними базами даних з допомгою бібліотеки typeorm та налаштувань над нею, роботу з не реляційними базами, такими як MongoDB з допомогою бібліотеки typegoose, побудови мікросервісної архітектури, роботи з брокерами повідомлень.

Також використовує патерн dependency injection.

Dependency Injection (DI) - шаблон проєктування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» (англ. Inversion of control, IoC) для розв'язання (отримання) залежностей. Впровадження — це передача залежності (тобто, сервісу) залежному об'єкту (тобто, клієнту). Передавати залежності клієнту замість дозволити клієнту створити сервіс є фундаментальною вимогою до цього шаблону проєктування.

У сервісі є багато вбудованих компонент, одна з базових це, так званий, модуль. Модуль - це клас з декоратором @Module (). Декоратор @Module () надає метадані, які Nest використовує для організації структури програми. Кожна програма Nest має як мінімум один модуль, корневий модуль. Корневий модуль - це місце, де Nest починає впорядковувати дерево додатків. Фактично, корневий модуль може бути єдиним модулем в вашому додатку, особливо коли додаток маленький, але це не має сенсу. У більшості випадків у вас буде кілька модулів, кожен з яких має тісно пов'язаний опцій. В Nest модулі за замовчуванням є Синглетон, тому ви можете без проблем використовувати один і той же екземпляр компонента між двома і більше модулями.

Модульна система Nest поставляється з функцією динамічних модулів. Це дозволяє створювати власні модулі без будь-яких зусиль.

В якості мови використовується TypeScript. Багато функціоналу реалізовується досить зручним способом через декоратори.

Розпочати роботу з nest проектом можна всього з 2 команд:

- npm i -g @nestjs/cli
- nest new demo-app

Після цього буде згенерований кістяк сервісу.

В практичній частині бакалаврської роботи використовується у основному сервісі обробки запитів та комунікації з сервісами обробки відео.

```
import { Controller, Get, Query, Post, Body, Put, Param, Delete } from '@nestjs/common';
import { CreateCatDto, UpdateCatDto, ListAllEntities } from './dto';

@Controller('cats')
export class CatsController {
  @Post()
  create(@Body() createCatDto: CreateCatDto) {
    return 'This action adds a new cat';
  }

  @Get()
  findAll(@Query() query: ListAllEntities) {
    return `This action returns all cats (limit: ${query.limit} items)`;
  }

  @Get(':id')
  findOne(@Param('id') id: string) {
    return `This action returns a #${id} cat`;
  }

  @Put(':id')
  update(@Param('id') id: string, @Body() updateCatDto: UpdateCatDto) {
    return `This action updates a #${id} cat`;
  }
```

Рисунок 2.1.8 Приклад контролера на nest.js

2.1.10. ffmpeg

FFmpeg є безкоштовне програмне забезпечення з відкритим кодом проект, що складається з великого набору бібліотеки і програм для обробки відео, аудіо та іншого мультимедіа файли та потоки. В його основі лежить сама програма FFmpeg, розроблена для командний рядокна основі обробки відео- та аудіофайлів. Він широко використовується для форматування перекодування, основне редагування (обрізка та конкатенація), масштабування відео, відео пост-продакшн ефекти та відповідність стандартам (SMPTE, MCE).

FFmpeg є частиною робочого циклу сотень інших програмних проектів, а його бібліотеки є основною частиною програмних медіаплеєрів, таких як VLC, і був включений в обробку ядра для YouTube і iTunes. Включено кодеки для кодування та / або декодування більшості форматів аудіо- та відеофайлів, що робить його дуже корисним для перекодування загальних та незвичайних медіафайлів в єдиний загальний формат.

В практичній частині бакалаврської роботи використовується, як основний засіб транскодування відео у HLS.

2.1.11. ffprobe

Ffprobe - утиліта яка дозволяю в зручному форматі отримати медіа інформацію з файла, таку як, наприклад, використані кодеки, тривалість відео, інше.

Використовується в практичній частині роботи для отримання медіа інформацію, у сервісі, що відповідає за аналіз та транскодинг відео.

2.1.12. websocket

WebSocket — це протокол, що призначений для обміну інформацією між браузером та веб-сервером в режимі реального часу. Він забезпечує двонаправлений повнодуплексний канал зв'язку через один TCP-сокет.

WebSocket спроектовано для втілення у веб-браузерах та веб-серверах, але може також використовуватись будь-яким клієнт-серверним застосунком.

З допомогою WebSocket можна будувати real-time веб додатки, використовується у меседжерах, онлайн чатах, біржах.

В практичній частині бакалаврської роботи використовується як канал для відображення статусу відео при обробці.

2.1.13. typeorm

Typeorm – потужна ORM для роботи з реляційними базами даних, написана мовою TypeScript та використовує багато його можливостей, у вигляді рефлексії та декораторів. Має дуже потужну систему автоматичних міграцій. В NestJS, який використовується в практичній частині бакалаврської роботи, присутні адаптери для typeorm для того щоб зробити роботу максимально зручною.

В практичній частині бакалаврської роботи використовується для комунікації з PostgreSQL.

2.1.14. class-validator, class-transformer

class-validator, class-transformer – бібліотеки для зручної роботи та валідації даних та автоматичного створення об’єктів базуючись на схемах. Генерація схем здійснюється автоматично, базуючись на декораторах.

2.1.15 React JS

React — це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду — “компонентів”.

З самого початку React був спроектований так, щоб його можна було впроваджувати поступово. Тобто ви можете додавати так мало або так багато React-у, як вам потрібно.

Досить швидкий, так як використовується таке поняття, як shadow dom, тобто перед тим як рендерити об’єкти безпосередньо в браузері, спочатку рендеряться в віртуальний dom, і потім тільки зміни застосовуються до справжнього dom.

Використовується так званий формат JSX, це такий підхід коли html розмітку можна вставляти в один файл з js кодом.

У практичній частині бакалаврської роботи застовується для написання клієнсткої частини для роботи з відео, з відображенням real-time статусу.

```
const useDocumentTitle = title => {
  useEffect(
    () => {
      document.title = title;
    },
    [title]
  );
};
function Counter() {
  const [count, setCount] = useState(0);
  const increment = () => setCount(count + 1);
  const title = `You clicked ${count} times.`;
  useDocumentTitle(title);
  return (
    <div>
      <h3>{count}</h3>
      <button onClick={increment}>Increment</button>
    </div>
  );
}
```

Рис. 2.1.14 – приклад коду з використанням React.

2.1.16. Ant Design

Ant Design – це поєднання дизайн системи та набору UI компонент. Має велику кількість компонентів та піконів. Простий у використанні та легкий до модифікацій. Має гарну прив'язку до React.JS та Angular.

У практичній частині бакалаврської роботи використовується для побудування інтерфейсу.

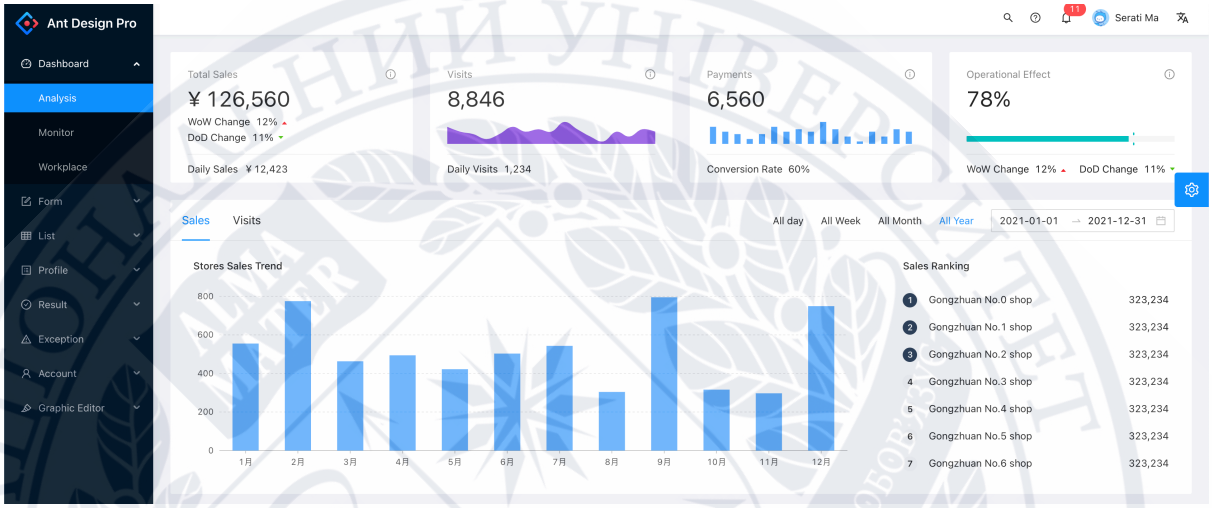


Рисунок 2.1.15 – приклад ant.d

2.1.17. Docker

Docker — інструментарій для управління ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів. Зокрема, Docker дозволяє не переймаючись вмістом контейнера запускати довільні процеси в режимі ізоляції і потім переносити і клонувати сформовані для даних процесів контейнери на інші сервери, беручи на себе всю роботу зі створення, обслуговування і підтримки контейнерів.

За рахунок Docker ми з легкістю зможемо перенести додаток на будь який сервер та запустити в ізольованому середовищі.

2.1.18. docker-compose

docker-compose – утиліта для зручного декларативного розвертання docker контейнерів. Принцип роботи досить простий, на вхід приймається маніфест docker-compose.yaml де описано які контейнери з якими образами потрібно запустити, та яка мережева взаємодія повинна бути налаштована, і утиліта створює потрібні ресурси та контейнери.

```yaml
version: '3.7'
services:
  redis:
    environment:
      - ALLOW_EMPTY_PASSWORD=yes
    image: bitnami/redis:latest
    restart: always
    hostname: redis
    networks:
      - redis-net
    volumes:
      - redis-data:/data
    ports:
      - "6379:6379"
  mongo:
    image: mongo:latest
    restart: always
    environment:
      - MONGO_INITDB_DATABASE=test
    volumes:
      - mongo-data:/data/db
    ports:
      - '27017:27017'
  elastic:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.7.0
    environment:
      - discovery.type=single-node
    volumes:
      - elastic-data:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
      - 9300:9300
  minio:
    image: minio/minio:RELEASE.2020-07-31T03-39-05Z
    volumes:
      - minio-data:/minio/data
    ports:
      - "9000:9000"
    environment:
```

Рисунок 2.1.17 – приклад docker-compose маніфесту

2.1.19. HLS

HLS - це протокол, визначений Apple для реалізації адаптивного формату потокового потоку, який може підтримуватися на їх пристроях та програмному забезпеченні. З часом він отримав широку підтримку.

Найважливішою особливістю HLS є його здатність адаптувати бітрейт відео до фактичної швидкості з'єднання. Це оптимізує якість користувацького досвіду. Тобто коли Інтернет не якісний завантажується фрагмент у низькій якості, коли канал досить швидкий – якісний. За рахунок цього у користувача,

навіть з мобільним Інтернетом, відео стабільно програється. Це дуже актуально стало в останні роки з розповсюдженням мобільних пристроїв та мобільного Інтернету.

Структура HLS складається з, так званих маніфестів, є один основний – майстер маніфест, який зберігає інформацію про маніфести в залежності від якості, і в свою чергу дочірні маніфести містять посилання на чанки – фрагменти відео.

Розділ 3.

ОПИС РОЗРОБКИ ТА ПРОЕКТУВАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Загальна структура веб-додатку

Щоб краще зрозуміти комунікації між сервісами зообразимо її на діаграмі

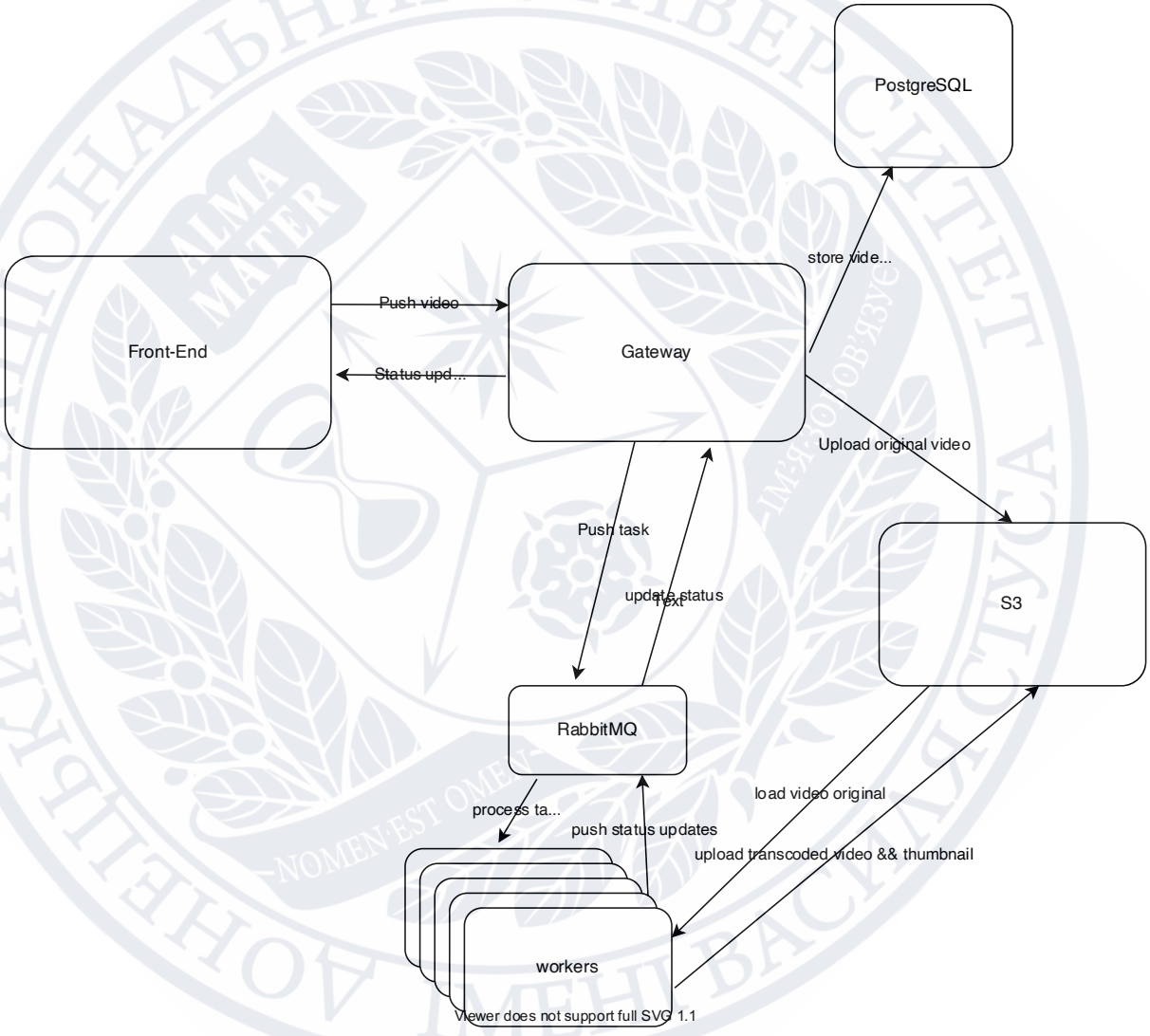


Рисунок 3.1

Як видно з діаграми, для розробки проекти були створені наступні додатки:

Gateway – основний сервіс, який також масштабується, який спілкується з клієнтами, з RabbitMQ, S3, Redis.

Єдине для чого використовується Redis це отримання та публікація назад повідомлень, котрі потрібно відправити по WebSocket. Такий підхід дозволяє зробити сокети масштабованими. Так як цих повідомлень багато, гарантія доставка не потрібна, черги не потрібні, а потрібен саме PubSub Redis з цією задачею ідеально справляється і не займає зайвих ресурсів. Без цього, якщо клієнт спілкується з одним екземпляром сервісу по WebSocket, а оновлення статусу з RabbitMQ прийшло на інший екземпляр, то повідомлення просто загубиться і не дійде.

Також основна задача сервісу це завантаження оригінального відео на S3 сховище, у бакалаврській роботі використовується Minio, як було сказано рашіне, це сховище, яке реалізовує протокол S3, і його можна запустити на своєму сервері.

Після завантаження відео, публікується, так звана задача у RabbitMQ, у цій задачі міститься інформація, про те яке відео потрібно обробити, його ідентифікатор, та інше.

Інформація про обробку відео, потрапляє у ще одну чергу RabbitMQ, з якої Gateway читає інформацію, оновляє статус у PostgreSQL та публікує повідомлення клієнтам через Redis.

Рисунок 3.2 – структура Gateway

Структура проекту досить проста, у нас є глобальний модуль app, в який через DI імпортуються інші підмодулі.

Основні модулі проекту:

- Users – відповідає за користувачів, та роботу з ними
- Auth – взаємодіє з Users для авторизації користувачів
- Subscriptions – модуль який працює для:
    - обробки підписок по websocket від користувачів
    - відправки повідомлень, обробки підключень
    - комунікації через redis
- Videos – модуль створений для:
    - Створенню задач по транскодуванню відео
    - Відправки оновлень у subscriptions
- Tasks – основна задача цього модулю це комунікація через RabbitMQ для публікації задач

У сервісі активно використовується typeorm для роботи з PostgreSQL, як було описано вище, це дуже зручно.

Рисунок 3.3 – приклад typeorm entity

На рис. 3.3 наведений приклад TypeORM Entity, на основі якої також генерується міграція, досить просто та дуже зручно використовувати.

Ось приклад використання генерованих Entity.

videos.gateway.ts × videos.controller.ts × users.service.ts × tasks.service.ts × subscriptions.

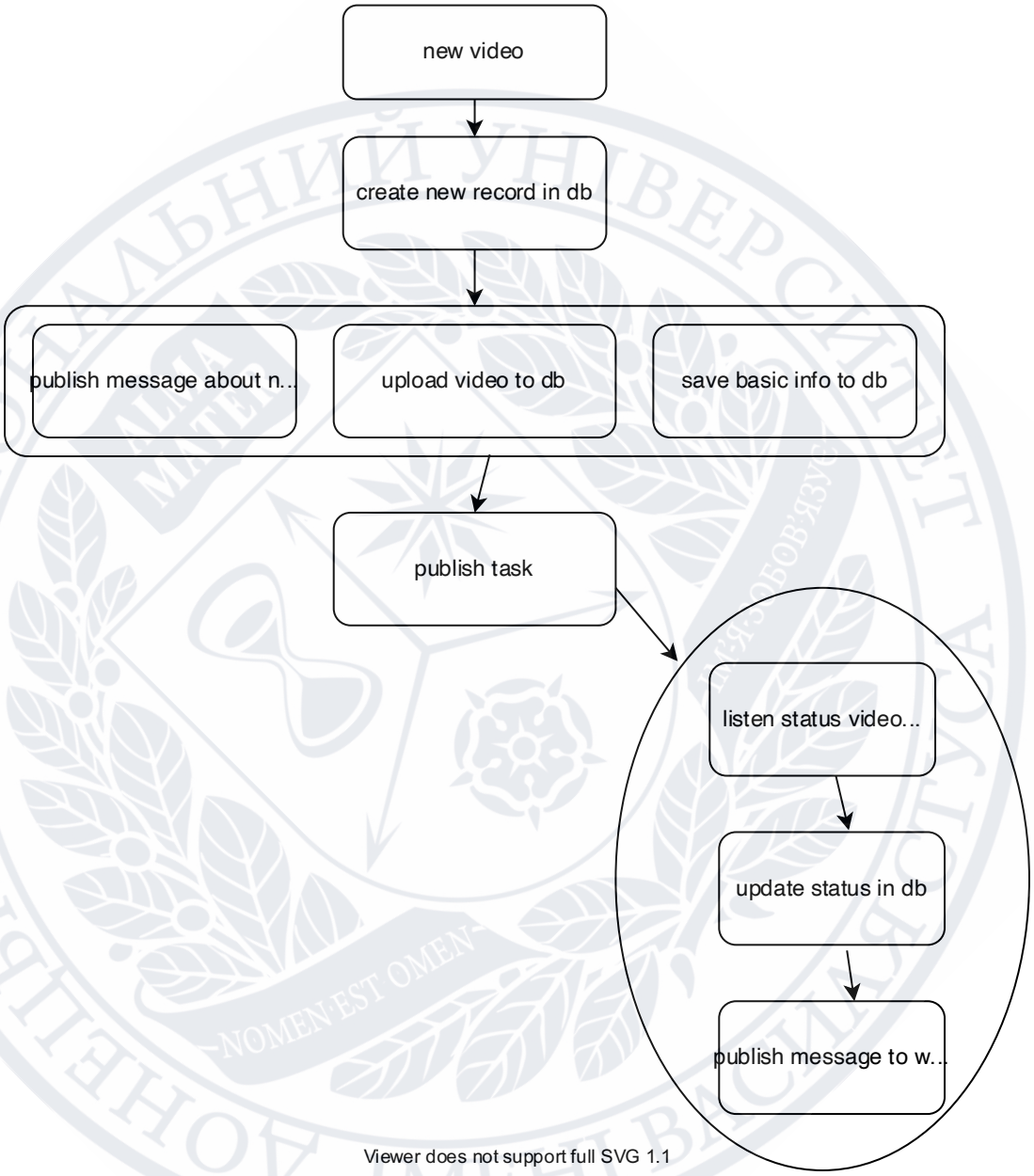
```ts
import { Injectable } from '@nestjs/common';
import { InjectS3, S3 } from 'nestjs-s3';
import { Body } from 'aws-sdk/clients/s3';
import { Video } from './videos.entity';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { VideoStatus } from './constants/status.constants';

@Injectable()
export class VideosService {
  constructor(
    @InjectS3() private readonly s3: S3,
    @InjectRepository(Video)
    private readonly videoRepository: Repository<Video>,
  ) {}

  async create(name: string): Promise<Video> {
    let video = this.videoRepository.create({
      name,
      status: VideoStatus.PREPARING,
    });

    video = await this.videoRepository.save(video);
    return video;
  }

  update(id: string, data: Partial<Video>): Promise<Video> {
    return this.videoRepository.save( entity: {
      id,
      ...data,
    });
  }

  findAll(): Promise<Video[]> {
    return this.videoRepository.find();
  }
```

Рисунок 3.5 – використання TypeORM

Архітектура сервіса є класичною для фреймоврка NestJS. Є, так званий, головний модуль app.module.ts в який підключаються залежності, це можуть

буди провайдери для роботи з базою даних, чи брокерами повідомлень, або просто під модулі додатку.

Щоб краще зрозуміти, що саме робить сервіс, нижче наведено спрощену діаграму процесів сервісу.



Працювати з таким проектом дуже зручно, так як є nestjs-cls, це утиліта з консольним інтерфейсом, яка дає змогу генерувати базові елементи, такі як:

- Application – власне сам, додаток
- Class – просто клас, який може використовуватись, будь яким, чином
- Controller – контролер для REST API

- Decorator – декоратор для TypeScript, як для функції, так і для класу чи поля
- Gateway – обробник websocket повідомлень

  Підтримує роботу з різними адаптерами, це може бути стандартий WebSocket, а може бути і socket-io, наприклад.

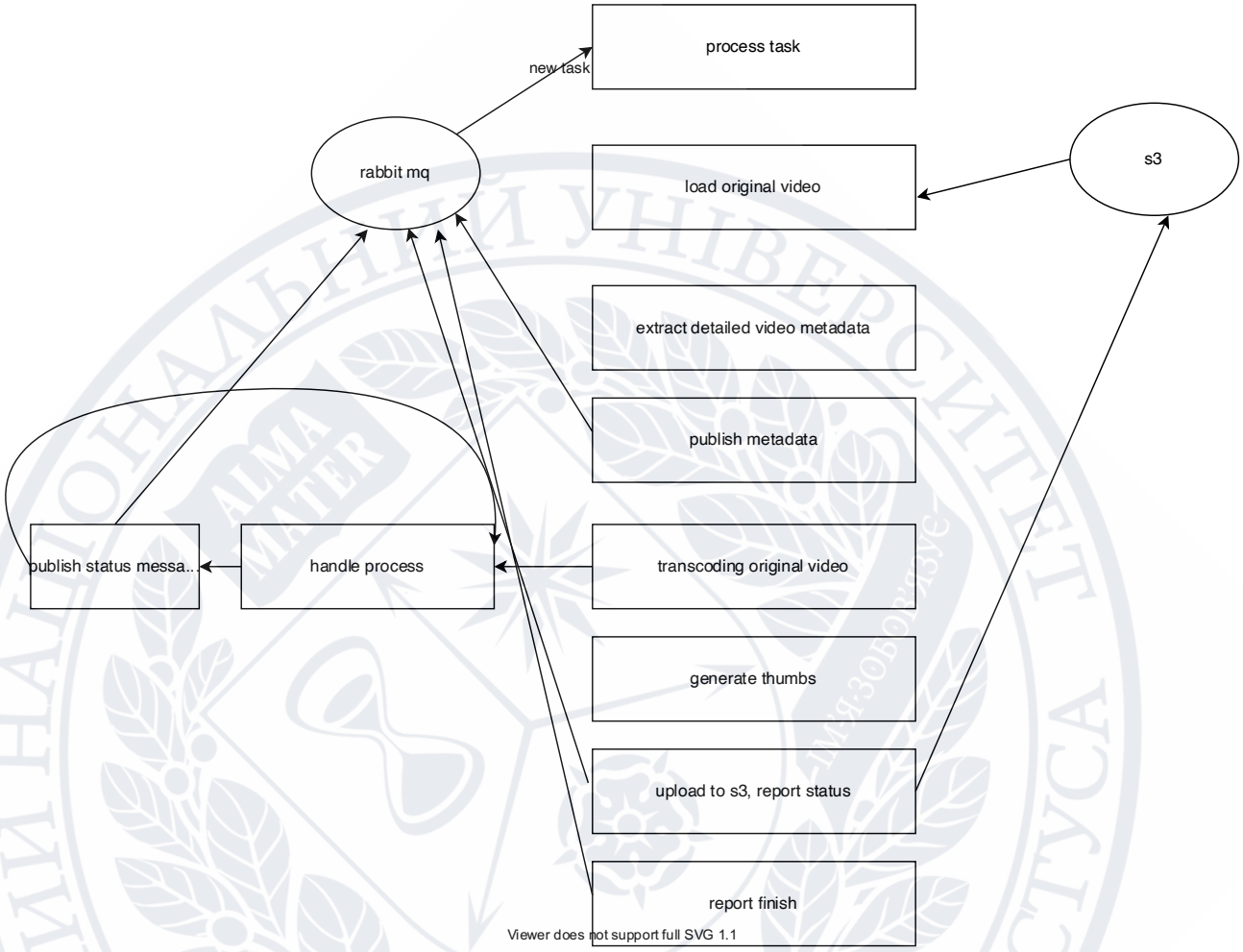
  Також є можливість створення своїх адаптерів, наприклад для роботи з WebSocket разом з Redis, як у нашому випадку.
- Guard – механізм захисту, кінцевих точок rest api, resolver GraphQL, або патернів у WebSocket
- Middleware – проміжний шар, використовується, наприклад, для того, щоб покласти у контекст запиту додаткову інформацію
- Module – модуль NestJS
- Resolver – обробник для GraphQL запитів
- Service – зазвичай тут пишеться логіка роботи з базою даних, обробки даних, тощо.

Це основні елементи, насправді їх набагато більше, але в практичній частині бакалаврської роботи використовуються лише ці.

В якості підсумку можна виділити, що сервіс Gateway є важливим в екосистемі проекту, він є зв’язною точкою мій клієнтами та іншими сервісами, базами даних, та брокерами повідомлень.

Далі розглянемо Worker сервіс, ця частина відповідає безпосередньо за обробку задач та конвертацію відео.

Нижче наведено спрощену діаграму процесів цього сервісу



Задача сервісу це під ’єднатись до RabbitMQ, слухати чергу повідомлень з задачами, коли з’являються нова задача, почати обробляти її, а саме

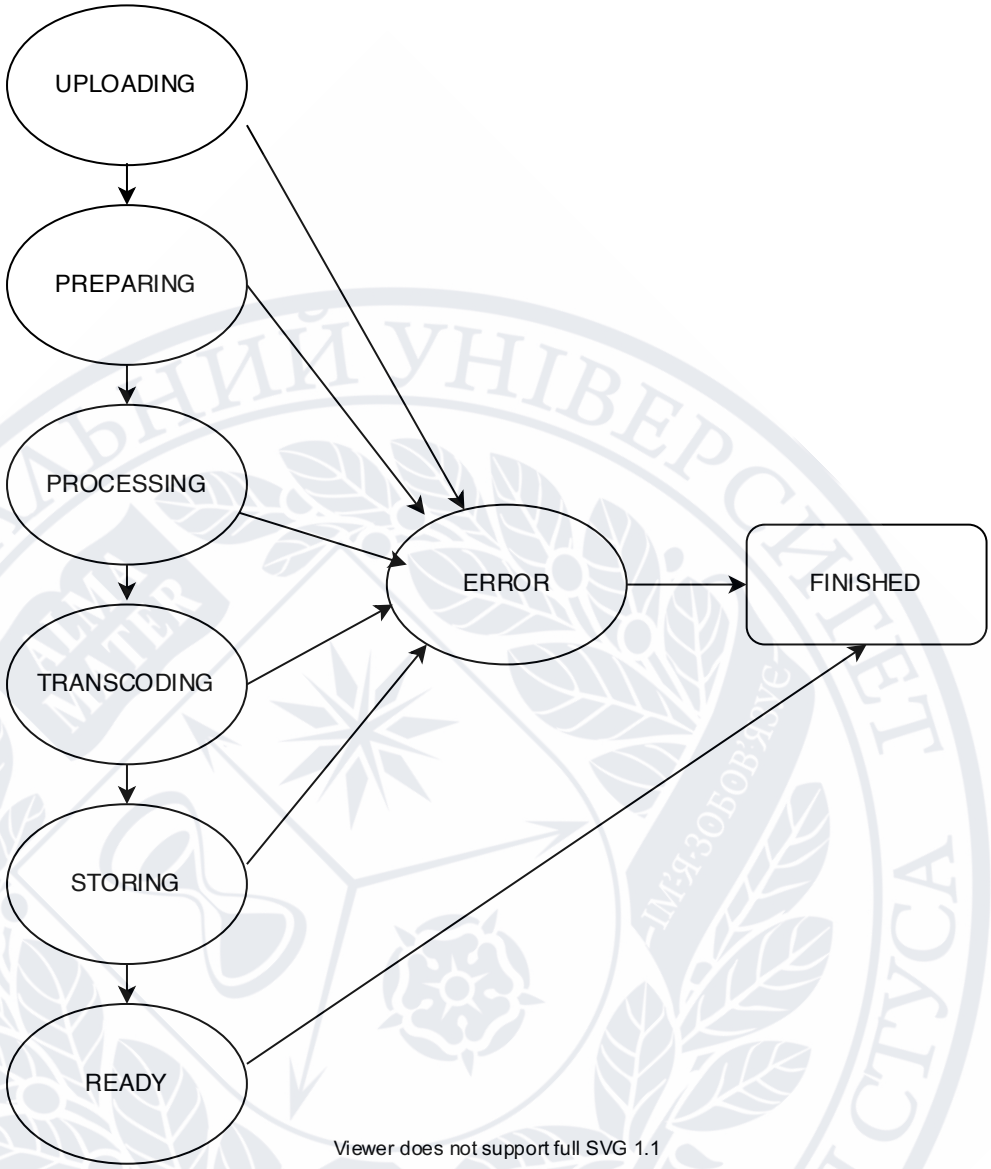
- Спочатку завантажити оригінальне відео з s3 сховища
- Дістати метаінформацію про відео, та опублікувати її у RabbitMQ канал
- Запустити транскодування відео через ffmpeg
- Слідкувати за статусом виконання, та доповідати про нього, у відповідний RabbitMQ канал
- Згенерувати картинку попереднього перегляду
- Завантажити перекодоване відео на s3
- Відзвітувати що задача виконана

При роботі з чергою задач, слід виділити основні моменти

- Підтримка одночасної (паралельної) обробки задач, з можливістю регулювати це
- Правильна робота з acknowledge (звітування RabbitMQ, що задача оброблена), щоб у випадку коли виник незапланований рестарт сервісу, і задача не встигла обробитись, при наступному запуску задача не зникла з черги
- Звітування про помилки, які можуть виникнути впродовж роботи
- Стабільність роботи повинна бути на високому рівні

Завдяки RabbitMQ було досягнуто досить гнучкої конфігурації та масштабування. Є можливість як і вертикального масштабування, так і горизонтального, регулюючи кількість екземплярів сервісу та кількості паралельної обробки задач.

Нижче наведена більш конкретизована зміна статусів відео



Як видно вхідною точкою є Uploading – коли відео тільки почало завантажуватись, створився запис у базі, та відправлені повідомлення підписникам. Проходячи через кожен етап, він може закінчитись помилкою, і це буде фінальним кроком для статусу. Або в позитивному варіанті, всі етапи пройшли успішно, і статус перейшов до READY, значить що відео повністю готове, і його можна переглянути.

Також в контексті бакалаврської роботи було створена front-end частину проекту, для PoC (Proof of concept) з обмеженим функціоналом.

Створення клієнтської частини було здійснено з допомогою бібліотеки ReactJS та набору компонентів Ant Design. Принцип роботи полягає у наступному:

- Відкрити WebSocket з’єднання до Gateway, підписатись на оновлення статусів відео
- При завантажені відео підписуватись на нове відео
- При зміні статусу реагувати на це, відображаючи прогрес виконання задачі
- Коли відео готове формувати посилання на відтворення відео з плеєром, з можливістю вибору якості

Нижче наведено фрагмент сторінки, де відображено список завантажених відео, та відео які тільки почали завантажуватись

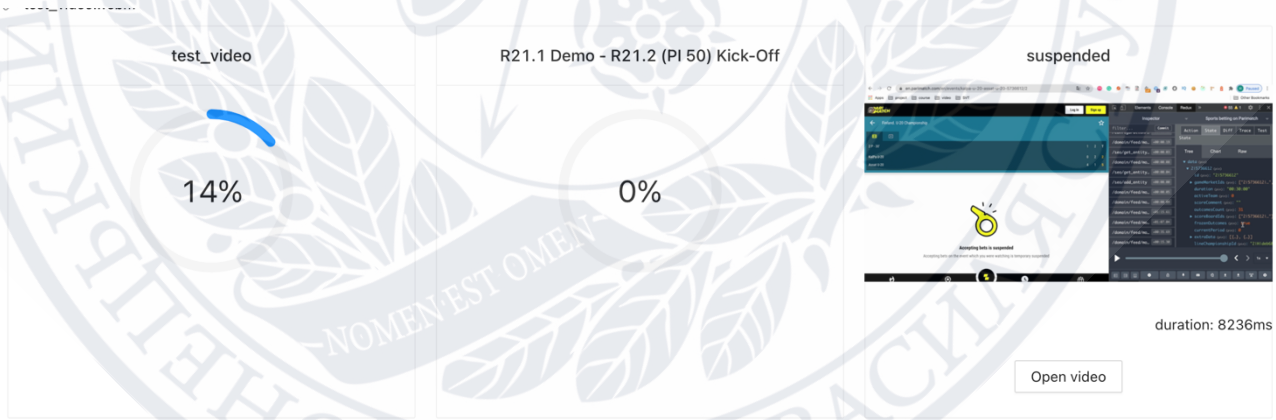


Рисунок 3.6 – сторінка з завантаженими відео, та відео які в процесі обробки

Також є можливість переглянути відео

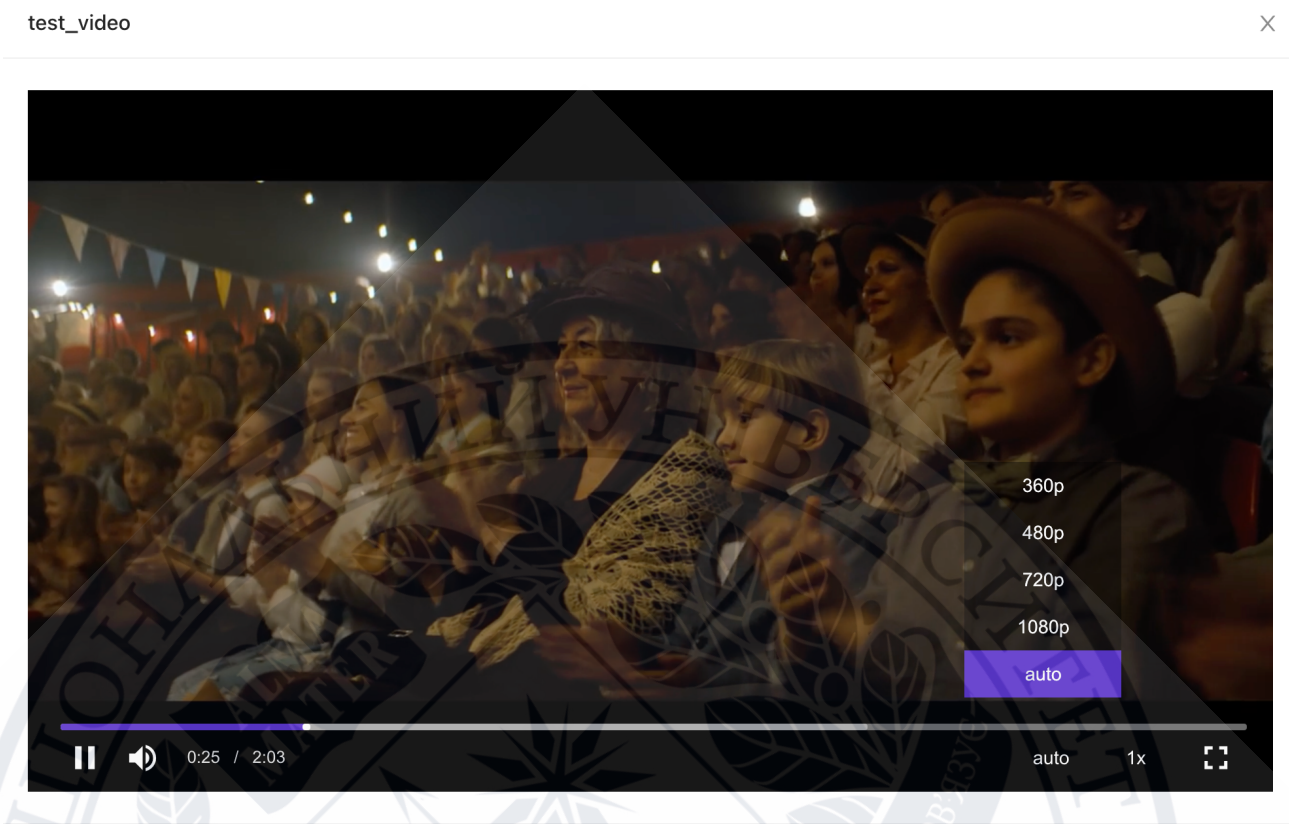


Рисунок 3.7 - відеоплеєр

Можна підсумувати, що поставлені цілі були виконані, у практичній частині вийшло створити швидкий та стабільний сервіс, який готовий до масштабування, як вертикального, так і горизонтального, завдяки використанню брокерів повідомлень, можна бути впевненим, що відео буде доставлено кінцевому користувачеві, і навіть якщо у процесі оброки виникли помилки, вони не залишаться без уваги.

ВИСНОВОК

У роботі представлені результати, котрі відповідають поставленій меті і є вирішенням задачі проектування сервісу для завантаження, обробки, аналізу та роздачі відео контенту кінцевим користувачам, з можливістю горизонтального масштабуванням, та з застосуванням мікросервісної архітектури.

При вирішенні завдання було проаналізовані основні гіганти сфери, проведене їхнє порівняння, та був поставлений вектор у якому рухатись, при написанні практичної частини.

Був оглянутий спектр технологій та програмних застосунків для вирішення задачі, були обрані оптимальні на суб'єктивну думку рішення. Далі було представлено основний інструментарій для виконання роботу. Були оглянуті основні особливості кожного застосунку, бібліотеки та фреймворку.

Для написання основого сервісу був використаний фреймворк NestJS, разом з використанням бібліотек для роботи з базою даних, брокером повідомлень, S3 сховищем, та інше. Для написання Worker компоненти була застосована мова програмування Go з використанням бібліотек для основого функціоналу роботи з чергами повідомлень, роботи з S3.

Також були задіяні наступні сторонні застосунки:

- RabbitMQ – взаємодія між сервісами, брокер повідомлень
- Redis спілкування між екземплярами додатку для синхронізації повідомлень, котрі відправляються через WebSocket
- PostegreSQL – як реляційна база даних

Основний функціонал для завантаження, аналізи та обробки відео був реалізований, також додатково було створено front-end частину з використанням бібліотеки ReactJS для відслідковування статусу відео та перегляду, відповідно.

Надалі планується допрацювати front-end частину, оптимізувати етап завантаження відео та імплементувати механізм для того, щоб можна було вбудовувати відео у будь який сайт чи мобільний додаток.

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТУСА
ALMA MATER
NOMEN EST OMEN
ІМ'Я ЗОБОВ'ЯЗУЄ

СПИСОК ЛІТЕАТУРИ

1. Apache Kafka [Електронний ресурс] – Режим доступу до ресурсу: https://kafka.apache.org/uses
2. RabbitMQ [Електронний ресурс] – Режим доступу до ресурсу: https://www.rabbitmq.com/documentation.html
3. NATS [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nats.io/
4. GoLang [Електронний ресурс] – Режим доступу до ресурсу: https://golang.org/doc/
5. What is PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/rds/postgresql/what-is-postgresql/
6. Apache Kafka [Електронний ресурс] – Режим доступу до ресурсу: https://kafka.apache.org/uses
7. Redis [Електронний ресурс] – Режим доступу до ресурсу: https://redis.io/topics/introduction
8. NestJS [Електронний ресурс] – Режим доступу до ресурсу: https://nestjs.com/
9. Why you should start using nest? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/coding-in-depth/why-you-should-start-using-nestjs-ffb3fd3b8451
10. TypeORM [Електронний ресурс] – Режим доступу до ресурсу: https://typeorm.io/
11. TypeORM + NestJS [Електронний ресурс] – Режим доступу до ресурсу: https://blog.theodo.com/2019/05/an-overview-of-nestjs-typeorm-release-your-first-application-in-less-than-30-minutes/
12. Ffmpeg – Режим доступу до ресурсу: https://ffmpeg.org/ffmpeg.html
13. First look at ffmpeg [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/swlh/a-first-look-at-ffmpeg-72537c65499e
14. Netflix [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Netflix

15. Youtube [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/YouTube
16. Twitch [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Twitch_(service)
17. TikTok [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/TikTok
18. Docker [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/
19. Docker-compose [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/compose/gettingstarted/
20. A Quick Guide To Understanding RabbitMQ & AMQP [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/swlh/a-quick-guide-to-understanding-rabbitmq-amqp-ba25fdfe421d
21. Building a microservices architecture with NATS [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/microservices-learning/building-a-microservices-architecture-with-nats-59fc8a4f331e
22. A complete guide to PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://prabhupant.medium.com/a-complete-guide-to-postgresql-e4d1cefb9866
23. Ant Design [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/
24. WebSockets in React, the component way! [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/practo-engineering/websockets-in-react-the-component-way-368730334eef
25. GoLang project structure [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/golang-standards/project-layout
26. 19 simple JavaScript coding standards to keep your code clean [Електронний ресурс] – Режим доступу до ресурсу: https://javascript.plainenglish.io/19-simple-javascript-coding-standards-to-keep-your-code-clean-7422d6f9bc0?gi=316be1d286e4

27. My TypeScript Best Practices [Електронний ресурс] – Режим доступу до ресурсу: https://non-traditional.dev/my-typescript-best-practices-845e196284aa?gi=5efc963d1fe9

РЕЦЕНЗІЯ

_____
Прізвище, ім'я, по-батькові

_____
Факультет

_____
Шифр і назва спеціальності

_____
Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «_____
_____»
є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних вправ, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____
дата підпис здобувача