НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“____” _____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Інженерія програмного забезпечення комп’ютерних систем”
спеціальності 121 “Інженерія програмного забезпечення”

на тему: *Веб застосунок для транскрибації, перекладу та створення дубляжів*

Виконав : студент __4__ курсу, групи _ІП-05
(шифр групи)
_____Коваль Максим Юрійович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____доцент, к.т.н. Порєв В.М._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____доцент, к.т.н. Волокита А.М.___
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____доцент, к.т.н Лісовиченко Олег Іванович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2024 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Коваля Максим Юрійовича

1. Тема проєкту *Веб застосунок для транскрибації, перекладу та створення дубляжів*
керівник проєкту Порєв Віктор Миколайович, доцент, к.т.н.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від

2. Термін здачі студентом закінченого проєкту 06.06.2024

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Аналіз предметної області.

Розділ 2. Вибір і обґрунтування засобів реалізації.

Розділ 3. Розробка системи.

Розділ 4. Демонстрація роботи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) діаграма класів (функціональна схема), діаграма зв'язку сутностей (структурна схема), діаграма прецедентів (принципова схема).

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Волокита А.М.		

7. Дата видачі завдання

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	12.10.2023-19.10.2023	
2.	*Вивчення та аналіз завдання*	20.10.2023-20.01.2024	
3.	*Розробка архітектури та загальної структури системи*	20.01.2024-30.01.2024	
4.	*Розробка структур окремих підсистем*	30.01.2024-10.02.2024	
5.	*Програмна реалізація системи*	8.02.2024-16.03.2024	
6.	*Оформлення пояснювальної записки*	14.04.2024-20.05.2024	
7.	*Захист програмного продукту*	03.06.2024-08.06.2024	
8.	*Передзахист*	03.06.2024-08.06.2024	
9.	*Захист*	19.06.2024	

Студент-дипломник _____ Коваль Максим
(підпис)

Керівник проєкту _____ Порєв Віктор

(підпис)

АНОТАЦІЯ

У дипломному проєкті було розроблено веб застосунок для траскрибування, перекладу та синтезу. Були сформовані конкретні вимоги та технічне завдання. Веб застосунок був розроблений на мові Java та бібліотеці Spring з використанням деяких сторонніх API. В проекті детально описано етапи формування підстав для розробки, огляд технологій розробки, процес самої розробки та тестування.

ANNOTATION

In graduation project, web application for transcribing, translation, and synthesis was developed. Specific requirements and terms of reference were formed. The web application was developed in Java and the Spring library using some third-party APIs. The project describes in detail the stages of forming the grounds for development, an overview of development technologies, the development process, and testing.

справки	*Формат*	*Значення*	*Найменування*	*Кіл. листів*	*№ екземпляра*	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Веб застосунок для	3		
			Траскрибування, перекладу			
			та синтезу			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Веб застосунок для	64		
			транскрибування, перекладу			
			та синтезу			
			Пояснювальна			
			записка			
	А4	***ІАЛЦ.467200.004 Д1***	Веб застосунок для	1		
			транскрибування, перекладу			
			та синтезу			
			Діаграма класів			
			(функціональна схема)			
	А4	***ІАЛЦ.4672008.005 Д2***	Веб застосунок для	1		
			траскрибування, перекладу			
			та синтезу			
			Діаграма зв’язку сутностей			
			(структурна схема)			
	А4	***ІАЛЦ.4672008.006 Д3***	Веб застосунок для	1		
			транскрибування, перекладу			
			та синтезу			
			Діаграма прецедентів			
			(принципова схема)			
	А4	***ІАЛЦ.4672008.007 Д4***	Веб застосунок для	38		
			транскрибування, перекладу			
			та синтезу			
			Текст програмного коду			

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	***ІАЛЦ.467200.001 ОА***		
Розроб.		*Коваль М. Ю.*			*Веб застосунок для траскрибації, перекладу та створення дубляжів* Опис альбому	*Літ.*	*Арк.* 1 / *Акрушів* 1
Перевір.		*Порєв В. М.*					
Н. Контр.						*КПІ ім.Ігоря Сікорського ФІОТ ІП-05*	
Затверд.							

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб застосунок для транскрибації, перекладу та створення дубляжів*»

Київ – 2024

ЗМІСТ

					ІАЛЦ.467200.002 ТЗ		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Коваль М.Ю.*			*Веб застосунок для траскрибації, перекладу та створення дубляжів* Технічне завдання	*Літ.*	*Арк.* 1 / *Акрушів* 4
Перевір.		*Порєв В.М.*					
						КПІ ім.Ігоря Сікорського ФІОТ ІП-05	
Н. Контр.							
Затверд.							

1.НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку веб застосунку для транскрибування, перекладу та синтезу. Це як нанесення підписів для відеороликів, так і створення повноцінних субтитрів з дублюванням на мові перекладу

Областю застосування цієї системи є медіа сфера. Цей застосунок буде корисним як для людей, які хочуть щоб їх могли зрозуміти як люди з інших частин світу, так і для людей з обмеженими здібностями.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3.МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробку веб застосунку для транскрибування, перекладу та синтезу (створення субтритрів та дубляжів на мові перекладу).

4.ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є публікації та статті,офіційні документації в інтернеті по цій темі, науково-технічна література, документація по мові програмування Java, документація по фреймворку Spring.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс API.
- Можливість вибору різних форматів субтитрів
- Підтримка основних мов

5.2. Вимоги до програмного забезпечення

Linux, Ffmpeg, INTELIJ IDEA IDE або аналоги.

5.3. Вимоги до апаратної частини

- Процесор Intel або AMD
- Оперативна пам'ять від 4 Гб
- Жорсткий диск від 10 Гб
- Канал Інтернет від 1мб/с

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	15.10.2023-20.10.2023
Вивчення та аналіз завдання	20.10.2023-20.01.2024
Розробка архітектури та загальної структури системи	20.01.2024-30.01.2024
Розробка структур окремих частин системи	30.01.2024-10.02.2024
Програмна реалізація системи	10.02.2024-20.03.2024
Виправлення помилок	20.03.2024-15.04.2024
Оформлення пояснювальної записки	15.04.2024-20.05.2024

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб застосунок для транскрибації, перекладу та створення дубляжів*»

Київ – 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

MVC (Model View Controller) Архітектурний шаблон

API (Application Programming Interface) Набір правил і протоколів для взажмодії між додатками

JPA (Java Persistence API) стандартизований інтерфейс

JVM (Java Virtual Machine) віртуальна машина для виконання байт-коду Java

JDK (Java Developer Kit) набір інструментів для розробки на java

SRT (SubRip Text) формат файлів з субтитрами

VTT / WebVTT (Web Video Text Tracks) формат файлів з субтитрами

SSA (Sub Station Alpha) формат файлів з субтитрами

SEO (Search Engine Optimization) оптимізація пошукового механізму

ADA (Americans with Disabilities Act) закон, щодо американців з обмеженими можливостями

NLP (Natural Language Processing) Технологія взаємодія людського мовлення з комп’ютером

CSS (Cascadian Style Sheets) Мова стилів для формування веб додатків

ВСТУП

Сучасний світ характеризується стрімким розвитком технологій, які впливають на всі аспекти нашого життя. Однією з таких інноваційних галузей є обробка мовлення, яка включає транскрибацію, переклад та синтез мови. Глобалізація сприяє збільшенню кількості міжнародних контактів. Люди з різних куточків світу все частіше взаємодіють у рамках бізнесу, освіти, науки та особистих зв'язків.

У свою чергу люди з обмеженими можливостями, такі як слабочуючі чи сліпі, можуть значно виграти від технологій транскрибації та синтезу мови. Застосунок може допомогти їм отримувати інформацію у зручній для них формі, покращуючи їхню інтеграцію у суспільство та доступ до інформації.

У сфері бізнесу, де комунікація з клієнтами та партнерами з різних країн є критично важливою, застосунок може значно підвищити ефективність роботи. Автоматичний переклад та транскрибація зустрічей, переговорів та ділових листів можуть зекономити час та ресурси, а також покращити точність переданої інформації.

Відеоконтент, подкасти та інші медіа все більше поширюються в інтернеті. Технології транскрибації можуть допомогти створювати субтитри для відео, перекладати контент на різні мови, а синтез мови може використовуватися для створення аудіоверсій текстового контенту, що робить його доступним для ширшої аудиторії.

Таким чином, розробка веб-застосунку для транскрибації, перекладу та синтезу мовлення є важливим кроком у напрямку створення більш зв'язкового, інклюзивного та ефективного глобального суспільства. Цей застосунок сприятиме покращенню комунікації, доступу до інформації та інтеграції різних соціальних груп, що в кінцевому результаті веде до загального прогресу та розвитку.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд ринку та актуальність

Ринковий контекст: Відео-контент стає все більш важливим у сучасному світі. Платформи, як Vimeo, YouTube, а також соціальні мережі на кшталт Facebook, Instagram і TikTok, активно просувають відео як основний формат контенту. Компанії та бренди все частіше використовують відео для маркетингових кампаній, рекламних роликів та презентацій, а освітні установи розширюють свої онлайн-курси та вебінари.

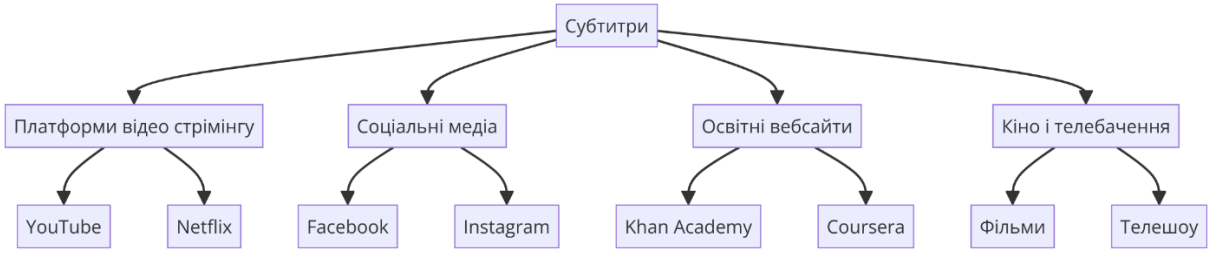


Рисунок 1.1 Основні сфери та місця використання субтитрів та дубляжів

Зростання попиту на інструменти для створення субтитрів: З ростом популярності відео збільшується і попит на інструменти для створення субтитрів та підписів (Рис 1.1). Це пов'язано з кількома ключовими факторами:

1. **Доступність для людей з вадами слуху**: Близько 5% світового населення страждає від різного ступеня втрати слуху. Для них субтитри є необхідними для сприйняття відео-контенту. Відсутність субтитрів може виключати значну частину потенційної аудиторії, особливо для освітніх та інформаційних відео.
2. **Розширення міжнародної аудиторії**: Багато відео створюються англійською мовою, яка не є рідною для більшості населення планети. Субтитри на різних мовах дозволяють розширити аудиторію, роблячи контент доступним для людей, які не розуміють мову оригіналу. Наприклад, субтитри китайською, іспанською або французькою можуть значно збільшити охоплення відео.

3. **Поліпшення SEO та залучення аудиторії**: Відео з субтитрами краще індексуються пошуковими системами, оскільки текст субтитрів допомагає пошуковим алгоритмам краще зрозуміти зміст відео. Це покращує позиції у пошукових результатах і, як наслідок, збільшує трафік. Крім того, субтитри можуть утримувати увагу глядачів довше, оскільки дозволяють сприймати інформацію не тільки через аудіо, але й через текст.
4. **Юридичні вимоги та стандарти**: У багатьох країнах існують законодавчі вимоги щодо доступності контенту для людей з обмеженими можливостями. Наприклад, у США Закон про американців з обмеженими можливостями (ADA) вимагає, щоб відео-контент був доступним для людей з вадами слуху, що включає наявність субтитрів.

Корпоративний та освітній контекст: У корпоративному середовищі субтитри стають важливим інструментом для навчання та комунікації. Наприклад, компанії, які створюють навчальні відео для співробітників, можуть використовувати субтитри для забезпечення розуміння матеріалу усіма працівниками, незалежно від їх рівня володіння мовою оригіналу. В освітніх установах субтитри допомагають студентам краще засвоювати матеріал, особливо коли мова йде про складні технічні або наукові теми (рис 1.2).

Приклади використання:

1. **YouTube та соціальні мережі**: Почнемо з того, що багато контент-креаторів використовують субтитри для підвищення залученості аудиторії та розширення її географії. Наприклад, відео-блогери, що створюють контент англійською мовою, можуть додавати субтитри іспанською або португальською для залучення глядачів з Латинської Америки чи деяких країн африки.
2. **Онлайн-курси та вебінари**: Платформи для онлайн-освіти, як Coursera, Udemy, edX, часто використовують субтитри для покращення доступності курсів. Це дозволяє студентам з різних країн брати участь у курсах, не маючи жодних мовних бар'єрів.

3. **Корпоративні тренінги**: Компанії, які проводять внутрішні тренінги для своїх співробітників, можуть використовувати субтитри для забезпечення однакового рівня розуміння матеріалу усіма працівниками, включаючи тих, для кого мова оригіналу не є рідною.

Висновок

Отже, зростання популярності відео-контенту та необхідність його доступності створюють сприятливі умови для розвитку веб-застосунків зі створення субтитрів та підписів, що відповідають потребам різних категорій користувачів юридичним та ринковим вимогам.

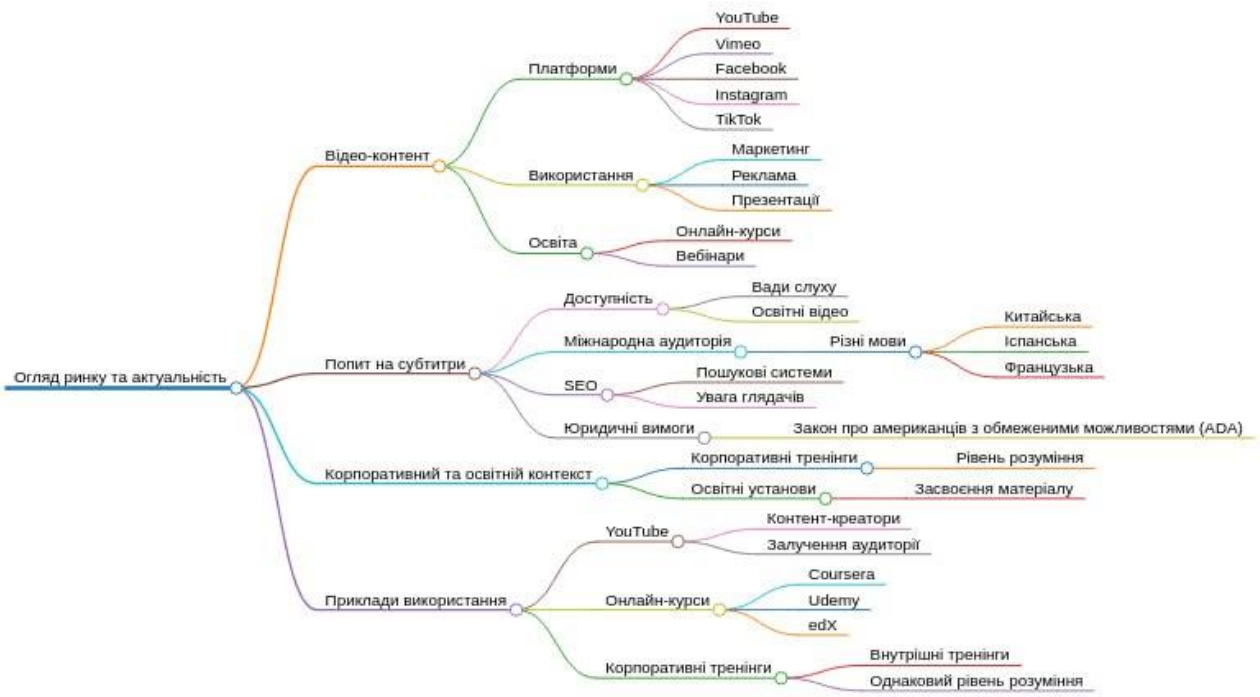


Рисунок 1.2 Огляд ринку та актуальності теми застосунку

1.2 Схожі застосунки

1. Amara

Опис: Amara – це онлайн-платформа для створення субтитрів та перекладу відео. Вона надає користувачам можливість створювати, редагувати та додавати субтитри до відео, які потім можна експортувати у різних форматах.

Основні функції:

- Інтерфейс для ручного додавання та редагування субтитрів.
- Автоматичне розпізнавання мови для створення субтитрів.
- Підтримка багатьох мов та переклад субтитрів.
- Інтеграція з платформами YouTube та Vimeo.
- Спільна робота над субтитрами з можливістю коментування та перегляду змін.

Переваги:

- Простий у використанні інтерфейс.
- Можливість працювати в команді.
- Безкоштовний для індивідуальних користувачів та неприбуткових організацій.

Недоліки:

- Обмежені можливості автоматизації у безкоштовній версії.
- Деякі функції доступні лише в платній версії.

2. Subtitle Horse

Опис: Subtitle Horse – це онлайн-інструмент для створення та редагування субтитрів у режимі реального часу. Платформа дозволяє працювати з субтитрами у браузері, без необхідності завантаження програмного забезпечення.

Основні функції:

- Таймлайн для синхронізації субтитрів з відео.
- Підтримка різних форматів субтитрів (SRT, VTT, SSA).

- Можливість імпорту та експорту субтитрів.
- Інструменти для ручного редагування та форматування тексту.

Переваги:

- Інтуїтивно зрозумілий інтерфейс.
- Швидка обробка субтитрів без завантаження додаткового ПО.
- Підтримка різних форматів субтитрів.

Недоліки:

- Відсутність автоматичного розпізнавання мови.
- Обмежені можливості інтеграції з іншими платформами.

3. а

Опис: Aegisub – це є безкоштовний програмний додаток для створення та редагування субтитрів. Він надає розширені інструменти для синхронізації та форматування субтитрів.

Основні функції:

- Підтримка популярних форматів субтитрів (SSA та SRT).
- Розширені можливості редагування та синхронізації субтитрів.
- Інструменти для автоматичної перевірки орфографії.
- Підтримка кількох мов для перекладу субтитрів.

Переваги:

- Потужні інструменти для точного редагування субтитрів.
- Підтримка великої кількості форматів субтитрів.
- Безкоштовний та відкритий код.

Недоліки:

- Складний для новачків через велику кількість функцій.

- Відсутність функцій автоматичного розпізнавання мови.

4. Rev

Опис: Rev – це онлайн-сервіс, який надає послуги транскрибування, створення субтитрів та перекладу. Rev використовує поєднання автоматичних інструментів, а також людської праці для забезпечення високої точності.

Основні функції:

- Автоматичне розпізнавання мови для створення субтитрів.
- Ручне редагування субтитрів професійними транскрибаторами.
- Підтримка різних форматів субтитрів (SRT, VTT).
- Інтеграція з платформами YouTube, Vimeo, та іншими.

Переваги:

- Висока точність завдяки поєднанню автоматизації та людської праці.
- Швидкий час обробки.
- Інтеграція з популярними відео-платформами.

Недоліки:

- Вартість послуг може бути високою для великих обсягів відео.
- Деякі функції доступні лише у платній версії.

5. Otter.ai

Опис: Otter.ai – це онлайн-платформа для автоматичного розпізнавання мови та створення транскриптів. Otter.ai також надає можливості для створення субтитрів для відео.

Основні функції:

- Автоматичне розпізнавання мови для створення транскриптів та субтитрів.
- Редагування тексту та синхронізація з відео.

- Підтримка кількох мов.
- Інтеграція з Zoom для автоматичного створення субтитрів під час відеоконференцій.

Переваги:

- Висока точність розпізнавання мови.
- Інтеграція з популярними інструментами для відеоконференцій.
- Зручний інтерфейс для редагування тексту.

Недоліки:

- Деякі функції доступні лише у платній версії.
- Обмежені можливості форматування субтитрів.

6. Kapwing

Опис: Kapwing – це онлайн-платформа для створення та редагування відео, яка також включає інструменти для автоматичного створення субтитрів.

Основні функції:

- Автоматичне розпізнавання мови та генерація субтитрів.
- Ручне редагування та форматування субтитрів.
- Підтримка різних форматів відео та субтитрів.
- Інтеграція з соціальними мережами для швидкого завантаження відео.

Переваги :

- Універсальний інструмент для редагування відео та субтитрів.
- Простий у використанні інтерфейс.
- Можливість швидкого завантаження відео на соціальні платформи.

Недоліки:

- Обмежені можливості для професійного редагування субтитрів.

- Деякі функції доступні лише у платній версії.

Ці інструменти надають широкий спектр можливостей для створення та редагування субтитрів, від автоматичного розпізнавання мови до ручного редагування та перекладу. Кожен з них має свої переваги та недоліки.

1.3 Ключові вимоги до застосунку

1.3.1 Підтримка основних мов

Підтримка популярних міжнародних мов є критично важливою для застосунку, з наступних причин:

1. **Розширення аудиторії**:
    - Забезпечуючи підтримку різних мов, застосунок може бути корисним для ширшого кола користувачів з різних країн і культур (рис 1.3). Це підвищує популярність та комерційний успіх продукту.
2. **Інклюзивність та доступність**:
    - Підтримка різних мов дозволяє людям з різними мовними навичками отримувати доступ до контенту, що сприяє інклюзивності. Наприклад, користувачі, які не володіють мовою оригіналу контенту, можуть використовувати субтитри для кращого розуміння.
3. **Поліпшення користувацького досвіду**:
    - Коли користувачі можуть переглядати контент на своїй рідній мові, це значно покращує їхній досвід. Вони легше сприймають інформацію і з більшою ймовірністю будуть задоволені сервісом.
4. **Освітні можливості**:
    - Застосунок, що підтримує різні мови, може бути використаний для вивчення мов. Люди можуть дивитися контент з субтитрами на іноземній мові, що сприяє їхній мовній практиці та розвитку.
5. **Підтримка багатомовних спільнот**:
    - В багатьох країнах та регіонах є спільноти, де розмовляють різними мовами. Підтримка цих мов в субтитрах допомагає забезпечити доступність контенту для всіх членів спільноти.

6. **Конкурентна перевага**:
    - Багатомовна підтримка може стати важливою конкурентною перевагою на ринку. Застосунок, що пропонує субтитри на багатьох мовах, більш привабливий для користувачів порівняно з тими, що мають обмежену мовну підтримку.
7. **Легалізація та відповідність регіональним вимогам**:
    - У деяких країнах є законодавчі вимоги щодо доступності контенту на офіційних мовах. Підтримка цих мов в субтитрах може допомогти дотримуватися місцевих законів та регуляцій.

Загалом, підтримка популярних міжнародних мов у застосунку для генерування субтитрів сприяє розширенню аудиторії, підвищенню задоволеності користувачів, інклюзивності та відповідності регіональним вимогам, що є ключовими факторами успіху на сучасному глобалізованому ринку.

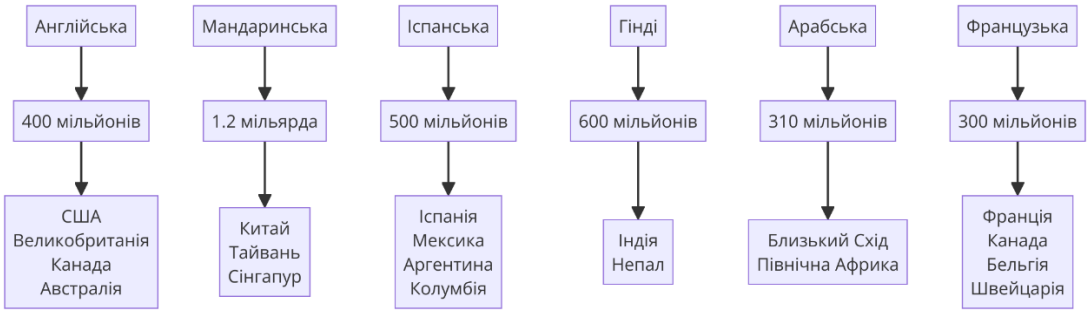


Рисунок 1.3 Найпопуляніші мови світу

1.3.2 Підтримка основних форматів субтритрів

Одними з основних форматів субтитрів є VTT та SRT

Приклад VTT

WEBVTT

00:00:00.500 --> 00:00:02.000

The Web is always changing

00:00:02.500 --> 00:00:04.300

and the way we access it is changing

Приклад SRT

1

00:00:00,500 --> 00:00:02,000

The Web is always changing

2

00:00:02,500 --> 00:00:04,300

and the way we access it is changing

1. **Сумісність з різними платформами та плеєрами**:
    - SRT та VTT є найбільш поширеними форматами субтитрів, які підтримуються більшістю медіаплеєрів, відеоплатформ та потокових сервісів, таких як YouTube, Vimeo, VLC, і т.д. Забезпечення

підтримки цих форматів дозволяє користувачам без проблем використовувати субтитри на різних пристроях та платформах.

2. **Простота та стандартизація**:
    - Формати SRT та VTT мають просту і зрозумілу структуру, що робить їх легкими для редагування та налаштування. Це також спрощує інтеграцію субтитрів у відео, оскільки вони є стандартом в індустрії.
3. **Широка підтримка інструментами та програмним забезпеченням**:
    - Існує багато інструментів та програм для створення, редагування та перевірки субтитрів у форматах SRT та VTT. Підтримка цих форматів дозволяє користувачам легко переходити від одного інструменту до іншого, не втрачаючи даних та форматування.
4. **SEO-оптимізація для веб-контенту**:
    - Формат VTT особливо важливий для веб-контенту, оскільки він підтримується HTML5 і дозволяє вбудовувати субтитри безпосередньо в веб-сторінки. Це сприяє поліпшенню пошукової оптимізації (SEO) відео, роблячи його доступнішим для пошукових систем.
5. **Адаптація до різних потреб користувачів**:
    - Різні формати субтитрів мають свої переваги. Наприклад, VTT підтримує додаткові функції, такі як стилізація тексту та позиціонування, що може бути важливим для деяких користувачів та додатків. Наявність підтримки обох форматів дозволяє задовольнити різні потреби та уподобання користувачів.
6. **Міжнародні стандарти та доступність**:
    - Формати SRT та VTT широко використовуються в міжнародних стандартах доступності. Це дозволяє забезпечити відповідність застосунку регуляторним вимогам різних країн щодо доступності відеоконтенту для людей з обмеженими можливостями.
7. **Гнучкість та масштабованість**:

- Підтримка різних форматів субтитрів робить застосунок більш гнучким і масштабованим, дозволяючи легко додавати нові формати в майбутньому, якщо це буде необхідно для задоволення змінних вимог ринку та користувачів.

Основними відмінностями цих форматів є

- **Формат заголовків:**
  - **SRT**: Не має спеціального заголовка. Вміст файлу починається безпосередньо з номеру субтитру.
  - **VTT**: Починається зі слова "WEBVTT", що вказує на формат файлу.
- **Формат часу:**
  - **SRT**: Використовує кому для відокремлення мілісекунд, наприклад `00:00:00,500`.
  - **VTT**: Використовує крапку для відокремлення мілісекунд, наприклад `00:00:00.500`.
- **Стилізація тексту:**
  - **SRT**: Не підтримує вбудовану стилізацію. Стилізація може бути додана лише за допомогою зовнішніх стилів чи інших засобів.
  - **VTT**: Підтримує CSS-подібну стилізацію всередині файлу, що дозволяє задавати стилі безпосередньо в файлі субтитрів.
- **Також ще деякі функції:**
  - **SRT**: Фокусується виключно на текстових субтитрах без додаткових можливостей.
  - **VTT**: Може містити метадані, стилі, а також спеціальні розмітки для більш гнучкого відображення субтитрів (наприклад, положення на екрані).

Висновки до розділу 1

Загалом, підтримка основних форматів субтитрів, таких як SRT та VTT, забезпечує сумісність, простоту використання, широку підтримку інструментами, поліпшення SEO, адаптацію до різних потреб користувачів, відповідність міжнародним стандартам та гнучкість застосунку, що є ключовими факторами для його успішного функціонування та популярності.

Ринок технологій обробки мови стрімко зростає і стає дедалі важливішим у різних галузях. Використання технологій автоматичного транскрибування, перекладу та синтезу мови стало невід'ємною частиною комунікаційних процесів у бізнесі, освіті, медицині та інших сферах. Попит на ці технології значно зріс завдяки глобалізації, розвитку віддаленої роботи та необхідності обслуговування багатомовної аудиторії.

Згідно з даними досліджень, світовий ринок обробки природної мови (NLP) очікується, що зросте до кількох мільярдів доларів у найближчі роки. Це зумовлено як зростаючою потребою в автоматизації рутинних завдань, так і підвищенням якості технологій машинного навчання, що використовуються для обробки мови. Основні гравці ринку, такі як Google, Microsoft та Amazon, активно інвестують у розвиток своїх NLP-технологій, що свідчить про їхнє стратегічне значення.

Отже, враховуючи динаміку розвитку ринку та потребу в ефективних рішеннях для обробки мови, веб застосунок для транкрибування, перекладу та синтезу є надзвичайно перспективним і своєчасним кроком.

РОЗДІЛ 2. ВИБІР І ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Java

Рисунок 2.1 Логотип java

Існує кілька причин, чому java є чудовим вибором для розробки застосунку

1. **Платформонезалежність**:
    - Java є мовою, що працює за принципом "write once, run anywhere" (WORA), тобто код, написаний на Java, може виконуватися на будь-якій платформі, яка підтримує JVM. Це дозволяє створити універсальний застосунок, який може працювати на різних операційних системах без необхідності внесення змін до коду.
2. **Масштабованість**:
    - Java добре підходить для створення масштабованих застосунків, які можуть обробляти великі обсяги даних та забезпечувати високий рівень продуктивності. Це важливо для застосунків, що генерують субтитри, оскільки вони можуть обробляти великі відеофайли та генерувати велику кількість текстових даних.
3. **Багатий набір бібліотек та фреймворків**:
    - Java має величезну екосистему бібліотек та фреймворків, які можуть значно спростити розробку застосунку. Наприклад, для роботи з текстом можна використовувати бібліотеки Apache Commons. Це дозволяє швидко та ефективно розробляти функціонал для генерування субтитрів.

4. **Безпека**:
    - Java має вбудовані механізми безпеки, такі як менеджер безпеки та система контролю доступу на основі політик. Це дозволяє розробникам створювати безпечні застосунки, що є особливо важливим для захисту даних користувачів та запобігання несанкціонованому доступу.
5. **Підтримка багатопотоковості**:
    - Java підтримує багатопотоковість, що дозволяє застосунку виконувати кілька завдань одночасно. Це може бути корисно для прискорення процесу генерування субтитрів, оскільки можна паралельно обробляти різні частини відео.
6. **Спільнота та підтримка**:
    - Java має одну з найбільших та найактивніших спільнот розробників. Це означає, що завжди можна знайти допомогу, документацію та приклади коду для вирішення будь-яких проблем, що можуть виникнути під час розробки застосунку.
7. **Інтеграція з іншими технологіями**:
    - Java легко інтегрується з іншими технологіями та мовами програмування. Це дозволяє створювати гібридні рішення, де частини застосунку можуть бути реалізовані на інших мовах, якщо це необхідно для досягнення певних функціональних чи продуктивних цілей.
8. **Довговічність та стабільність**:
    - Java є однією з найбільш стабільних та перевірених мов програмування, що використовується вже більше двох десятиліть. Це забезпечує довготривалу підтримку та стабільність для розроблених на ній застосунків.

Враховуючи ці переваги, використання Java для створення застосунку, що генерує субтитри, забезпечує надійність, масштабованість, безпеку та ефективність, що є ключовими факторами для успішного функціонування такого застосунку.

2.2 Spring framework

Рисунок 2.2 Spring логотип

Використання spring важливе через наступний ряж причин

1. **Розширюваність та модульність**:
   - Spring Framework побудований на принципах інверсії контролю (IoC) та ін'єкції залежностей (DI), що сприяє модульності та розширюваності застосунку. Це дозволяє легко додавати нові функції та компоненти без порушення існуючої архітектури (рис 2.3).

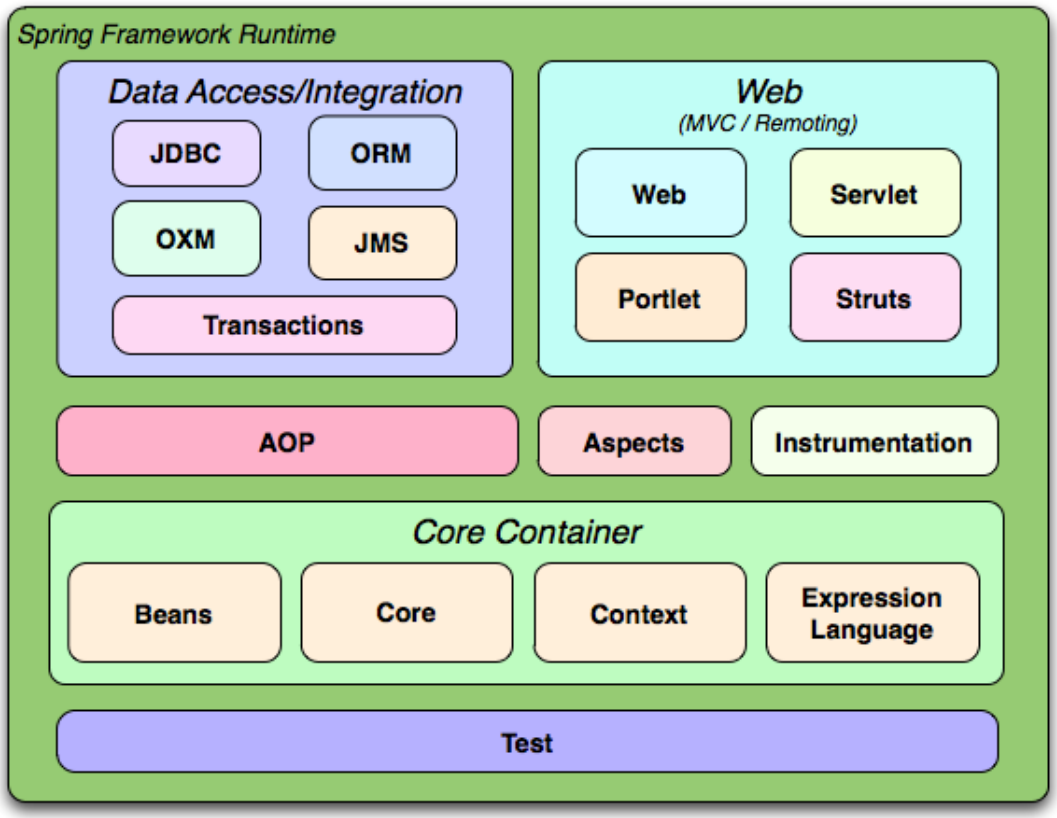


Рисунок 2.3 Структура Spring Framework

2. **Підтримка MVC архітектури**:
    - Spring MVC є потужним веб-фреймворком, який полегшує розробку веб-застосунків, дотримуючись патерну Model-View-Controller (MVC). Це сприяє розділенню логіки, представлення та обробки даних, що робить код чистим і легким для підтримки.
3. **Безпека**:
    - Spring Security забезпечує потужні механізми для аутентифікації та авторизації користувачів, що є критично важливим для захисту даних користувачів та забезпечення безпеки веб-застосунку. Це включає підтримку різних протоколів безпеки, таких як OAuth2, JWT, та інші.
4. **Легка інтеграція з базами даних**:
    - Spring Data JPA та інші модулі Spring Data значно спрощують роботу з базами даних, забезпечуючи зручні інтерфейси для виконання CRUD-операцій та складних запитів. Це допомагає швидко та ефективно взаємодіяти з базами даних, зберігаючи та отримуючи інформацію про субтитри.
5. **Підтримка RESTful веб-сервісів**:
    - Spring Boot та Spring MVC роблять створення RESTful веб-сервісів швидким і зручним. Це дозволяє легко створювати API для взаємодії з фронтендом або іншими сервісами, забезпечуючи гнучкість і масштабованість веб-застосунку.
6. **Масштабованість та продуктивність**:
    - Spring Framework підтримує побудову масштабованих та високопродуктивних застосунків, що може бути особливо важливо для обробки великих обсягів даних та користувацьких запитів в реальному часі. Spring Boot також спрощує конфігурацію та розгортання застосунків, що сприяє швидкому масштабуванню.
7. **Велика спільнота та підтримка**:
    - Spring має одну з найбільших та найактивніших спільнот розробників. Це означає, що завжди можна знайти допомогу, документацію,

приклади коду та інші ресурси, які допоможуть у розробці та підтримці застосунку.

8. **Мікросервісна архітектура**:
   - Spring Cloud надає набір інструментів для розробки мікросервісних архітектур, що дозволяє створювати масштабовані та розподілені системи. Це може бути корисно для застосунку, що генерує субтитри, оскільки різні компоненти (наприклад, обробка відео, генерація субтитрів, зберігання даних) можуть бути реалізовані як окремі сервіси.
9. **Тестування та якість коду**:
   - Spring Framework забезпечує потужні інструменти для тестування, такі як Spring Test, що дозволяють писати юніт-тести, інтеграційні тести та енд-то-енд тести. Це сприяє підтриманню високої якості коду та швидкому виявленню помилок.

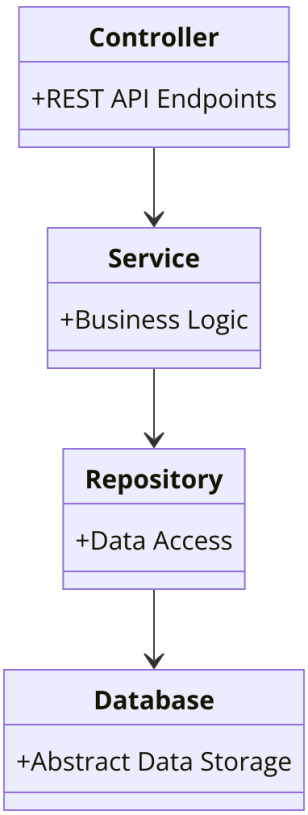


Рисунок 2.4 Базова структура spring rest-застосунку

Отже, враховуючи ці переваги, використання Spring Framework для веб-

застосунку, забезпечує масштабованість, безпеку, продуктивність та легкість інтеграції, що є ключовими факторами для успішного функціонування такого застосунку.

2.3 Apache commons

Ключові причина використовувати бібліотеку Apache commons. Структуру бібліотеки commons lang можна побачити на рисунку (2.5)

1. **Зручність та прискорення розробки**:
    - Apache Commons містить велику кількість готових до використання утиліт і функцій, що дозволяють вирішувати різноманітні завдання без необхідності писати власний код. Це прискорює розробку і дозволяє зосередитися на основній функціональності застосунку.
2. **Обробка тексту та даних**:
    - Commons Lang та Commons Text містять багатий набір утиліт для роботи з рядками, обробки тексту та виконання різних маніпуляцій з даними. Це особливо корисно для генерування субтитрів, де необхідно часто обробляти текстові дані.
3. **Обробка файлів**:
    - Commons IO надає зручні засоби для роботи з файлами, включаючи читання, запис, копіювання та інші операції. Це спрощує процес збереження, завантаження та управління файлами субтитрів у форматах SRT та VTT.
4. **Конфігурація та налаштування**:
    - Commons Configuration дозволяє легко управляти конфігураційними налаштуваннями застосунку. Це може бути корисно для збереження різних параметрів, таких як налаштування бази даних, шляхи до файлів та інші конфігурації, що спрощує підтримку та адміністрування застосунку.
5. **Обробка чисел і дат**:

- Commons Math і Commons Lang надають інструменти для роботи з числами, математичними операціями та датами. Це може бути корисно для точного обрахунку часових інтервалів у субтитрах, зокрема для синхронізації тексту з відео.

6. **Управління колекціями**:
   - Commons Collections надає розширені можливості для роботи з колекціями, що може бути корисно для ефективного зберігання та обробки великих обсягів даних субтитрів.
7. **Робота з мережею**:
   - Commons Net містить утиліти для роботи з різними мережевими протоколами, що може бути корисно для завантаження або передачі файлів субтитрів через мережу.
8. **Обробка JSON**:
   - Commons Lang і інші утиліти можуть допомогти у простій обробці JSON-даних, що може бути корисно для обміну даними між клієнтом і сервером або зберігання налаштувань у форматі JSON.

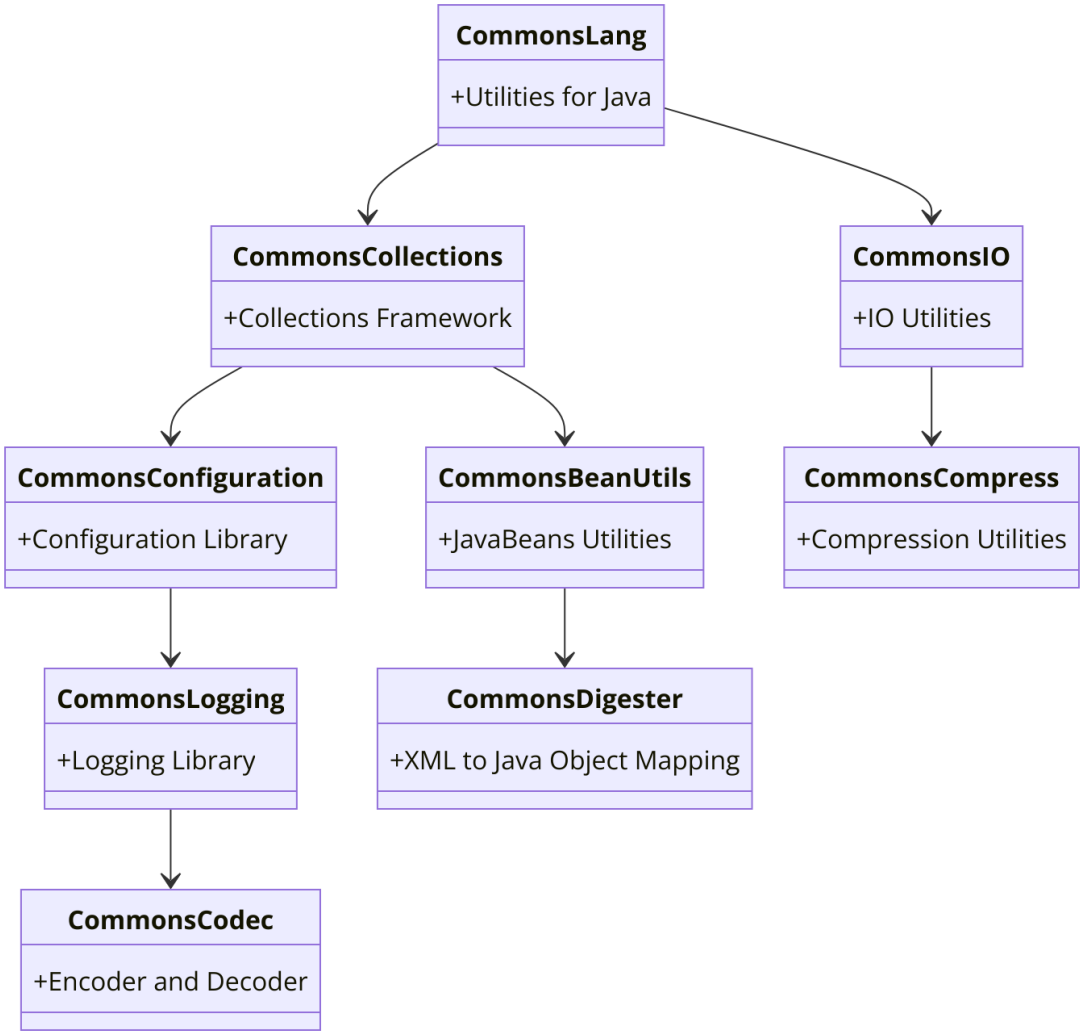


Рисунок 2.5 Структура компонентів commons lang

2.4 Сторонні спецілазовані api

Deepgram

Розпізнавання мови

- **Точність**: Deepgram забезпечує високоточне розпізнавання мовлення, що дозволяє створювати точні субтитри для відео та аудіо.
- **Швидкість**: Це API може швидко обробляти великі обсяги аудіо даних, що є важливим для стрімінгових сервісів.
- **Підтримка багатьох мов**: Підтримує різні мови та акценти, що робить його універсальним інструментом для глобальних платформ.

DeepL

Переклад тексту

- **Висока якість перекладу**: DeepL відомий своєю високою якістю перекладу, що забезпечує точність та природність перекладених субтитрів.
- **Швидкість**: Миттєвий переклад тексту дозволяє оперативно створювати субтитри на різних мовах для різних аудиторій.
- **Підтримка багатьох мов**: Підтримує широкий спектр мов, що робить його корисним для мультимовних застосунків.

ElevenLabs

Генерація голосу

- **Натуральність**: ElevenLabs використовує передові алгоритми синтезу мови, що дозволяє створювати натуральні голоси для озвучення субтитрів.
- **Налаштовуваність**: Можливість налаштовувати голоси під різні стилі та емоції, що може бути корисним для різних типів контенту.
- **Інтеграція з іншими сервісами**: Легка інтеграція з іншими API, такими як Deepgram та DeepL, для створення комплексних мультимедійних рішень.

Загальна важливість

- **Покращення користувацького досвіду**: Точні субтитри та якісний переклад покращують доступність та розуміння контенту для користувачів з різних країн.
- **Підвищення ефективності**: Автоматизація процесів розпізнавання мовлення, перекладу та генерації голосу зменшує час і ресурси, необхідні для створення субтитрів.
- **Масштабованість**: Використання сторонніх API дозволяє легко масштабувати сервіс без значних витрат на інфраструктуру.

2.5 Ffmpeg

FFmpeg є надзвичайно корисним інструментом з кількох причин:

1. Багатофункціональність

FFmpeg підтримує велику кількість форматів відео та аудіо. Він може обробляти практично будь-який медіа-файл, перетворюючи його в інші формати, що робить його універсальним інструментом для роботи з мультимедіа.

2. Конвертація формату

Один з основних застосунків FFmpeg – конвертація медіа-файлів з одного формату в інший. Це дозволяє легко підготувати відео або аудіо для різних пристроїв та платформ.

3. Редагування відео та аудіо

FFmpeg дозволяє здійснювати базові та просунуті операції редагування, такі як обрізка, об'єднання, розділення, додавання фільтрів та ефектів до відео та аудіо.

4. Стиснення відео

З його допомогою можна ефективно стиснути відео без значної втрати якості, що є корисним для зменшення розміру файлів та підвищення ефективності зберігання та передавання медіа.

5. Автоматизація процесів

FFmpeg підтримує командний рядок, що дозволяє автоматизувати багато процесів обробки медіа. Це особливо корисно для серверів або скриптів, які потребують обробки великої кількості файлів без участі користувача.

6. Підтримка потокової передачі

FFmpeg може кодувати та декодувати потоки відео та аудіо в реальному часі, що робить його відмінним вибором для організації потокових трансляцій і запису з них.

7. Платформна незалежність

FFmpeg є кросплатформним інструментом, що означає, що він може працювати на різних операційних системах, включаючи Windows, macOS та Linux.

8. Велика спільнота та підтримка

Завдяки великій спільноті розробників та користувачів, FFmpeg має гарну підтримку, включаючи детальну документацію, приклади використання та активні форуми.

9. Вільне програмне забезпечення

FFmpeg є програмою з відкритим вихідним кодом, що означає, що його можна використовувати, змінювати та поширювати безкоштовно.

Ці переваги роблять FFmpeg потужним інструментом для будь-яких завдань, пов'язаних з обробкою мультимедійних файлів.

Висновки до розділу 2

Отже, використання технологій Java, Spring та Apache Commons для розробки веб-застосунку забезпечує значні переваги, які сприяють успіху і ефективності проекту.

1. **Java**:
    - **Платформонезалежність**: Java дозволяє створювати застосунки, які можуть працювати на будь-якій платформі, що підтримує JVM, забезпечуючи широке охоплення користувачів.
    - **Масштабованість**: Висока продуктивність і можливість обробки великих обсягів даних роблять Java ідеальною для побудови масштабованих систем.
    - **Безпека**: Вбудовані механізми безпеки Java допомагають захищати дані користувачів і забезпечують надійну роботу застосунку.
    - **Багатий набір бібліотек**: Екосистема Java пропонує широкий вибір бібліотек, які прискорюють розробку і підвищують функціональність застосунку.
2. **Spring Framework**:
    - **Розширюваність та модульність**: Інверсія контролю та ін'єкція залежностей сприяють гнучкості та простоті розширення застосунку.
    - **Підтримка MVC**: Полегшує розділення логіки, представлення і обробки даних, роблячи код чистим і легким для підтримки.
    - **Безпека**: Spring Security забезпечує потужні засоби для аутентифікації та авторизації, що критично важливо для захисту даних.
    - **Легка інтеграція з базами даних**: Spring Data спрощує роботу з базами даних, забезпечуючи зручні інтерфейси для виконання операцій.
3. **Apache Commons**:

- **Зручність та прискорення розробки**: Надає великий набір утиліт і функцій для вирішення різних завдань, що знижує потребу в написанні власного коду.
- **Обробка тексту та даних**: Полегшує маніпуляції з текстовими даними, що особливо важливо для генерування субтитрів.
- **Обробка файлів**: Зручні засоби для роботи з файлами спрощують процес збереження та управління субтитрами.
- **Надійність та перевіреність**: Бібліотеки Apache Commons широко використовуються і підтримуються спільнотою, що забезпечує стабільність і надійність.

4. **Сторонні спеціалізовані арі**

- **Покращення користувацького досвіду**: Точні субтитри та якісний переклад покращують доступність та розуміння контенту для користувачів з різних країн.
- **Підвищення ефективності**: Автоматизація процесів розпізнавання мовлення, перекладу та генерації голосу зменшує час і ресурси, необхідні для створення субтитрів.
- **Масштабованість**: Використання сторонніх API дозволяє легко масштабувати сервіс без значних витрат на інфраструктуру.

5. **Ffmpeg**

Що ж стосується ffmpeg. То це є надзвичайно потужним і універсальним інструментом для роботи з мультимедійними файлами. Його можливості включають конвертацію форматів, редагування, стиснення, автоматизацію процесів, підтримку потокової передачі та кросплатформну сумісність. Завдяки своїй багатофункціональності, відкритому коду та великій спільноті, FFmpeg стає незамінним для професіоналів та ентузіастів, які працюють з відео та аудіо.

Отже, комбінація Java, Spring та Apache Commons та інших сервісів надає потужний набір інструментів для створення надійного, масштабованого та безпечного веб-застосунку. Це дозволяє розробникам швидко і ефективно створювати функціональні рішення, які можуть задовольнити різноманітні потреби користувачів, забезпечуючи при цьому високу якість та продуктивність.

Використання цих технологій допомагає створити застосунок, який легко підтримувати, розширювати та інтегрувати з іншими системами, що є важливими факторами для довгострокового успіху та розвитку проекту.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Встановлення ffmpeg

Важливим компонетом застосунку є ffmpeg.

Рисунок 3.1 Логотип ffmpeg

FFmpeg — це вільний набір інструментів для роботи з мультимедіа-даними, який включає бібліотеки та програми для запису, перетворення, стрімінгу та відтворення аудіо- і відеофайлів. Це один з найбільш потужних і широко використовуваних інструментів у сфері обробки мультимедіа.

У проекті він необхідний для зміни звуку у відео, вшиванню субтитрів і синтезованого перекладу.

Система на за якою я буду працювати – Arch Linux. Тому для встановлення ffmpeg мені необхідно

```
SUDO PACMAN -S FFMPEG
```

Для розробки я буду використовувати версію 6.0.

3.2 Отримання ключів сторонніх API

У застосунку я збираюсь використовувати 3 стороніх сервіса. Кожен з них має як платні, так і безкошстовні тарифні плани. У дипломному проекті я використовую безкоштовні. Нижче я привожу інструкції для кожного з сервісів

Deepgram

1. **Реєстрація:**
    - Переходимо на сайт Deepgram.
    - Натискаємо на кнопку "Get Started for Free" або "Sign Up".
    - Заповнюємо реєстраційну форму, надавши свою електронну адресу та пароль, або зареєструйтесь через соціальні мережі (Google, GitHub).
    - Підтверджуємо свою електронну адресу, якщо це необхідно.
2. **Отримання API ключа:**
    - Після входу в обліковий запис, переходимо на вкладку "API Keys" або "Dashboard".
    - Натискаємо кнопку "Create API Key" або "Generate New API Key".

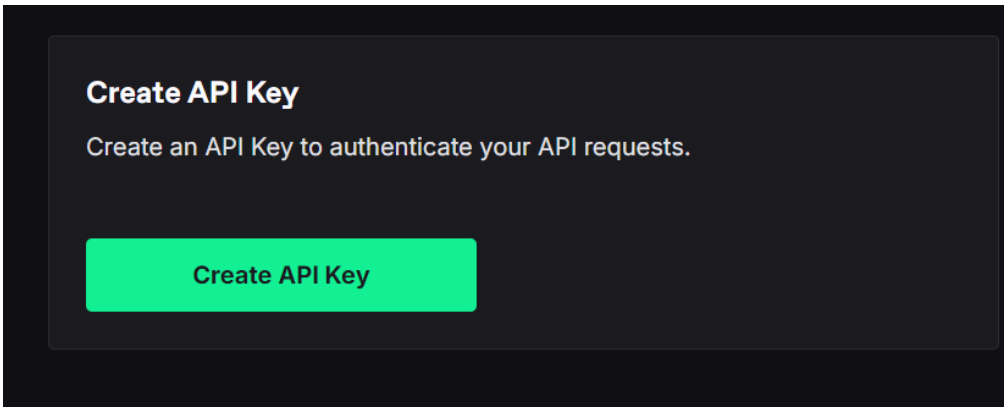


Рисунок 3.2 Створення ключа для deepgram

    - Даємо ключу ім'я для зручності (наприклад, "My First Key") і натисніть "Create".
    - Копіюєм згенерований API ключ.

DeepL

1. **Реєстрація:**
    - Переходимо на сайт DeepL.
    - Натискаємо на кнопку "Sign Up".
    - Заповнюємо реєстраційну форму, надавши свою електронну адресу та пароль.
    - Підтверджуємо свою електронну адресу, якщо це необхідно.
2. **Отримання API ключа:**
    - Після входу в обліковий запис, переходимо на вкладку "Account" або "API".

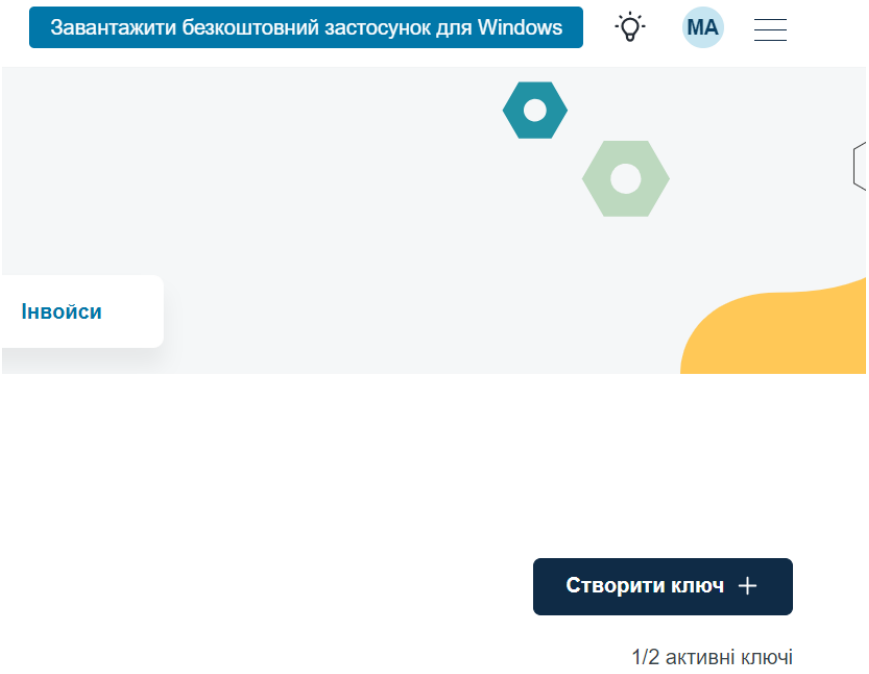


Рисунок 3.3 Створення ключа для deepL

- На сторінці API знайдемо розділ, де вказано API ключ.
- Якщо ключ ще не згенеровано, натиснемо кнопку "Generate API Key".
- Копіюємо згенерований API ключ.

ElevenLabs

1. **Реєстрація:**
    - Переходимо на сайт ElevenLabs.
    - Натискаємо на кнопку "Get Started"
    - Заповнюємо реєстраційну форму, надавши свою електронну адресу та пароль, або зареєструйтесь через соціальні мережі.
    - Підтверджуємо свою електронну адресу.
2. **Отримання API ключа:**
    - Після входу в обліковий запис, переходимо на вкладку "API" або "Dashboard".

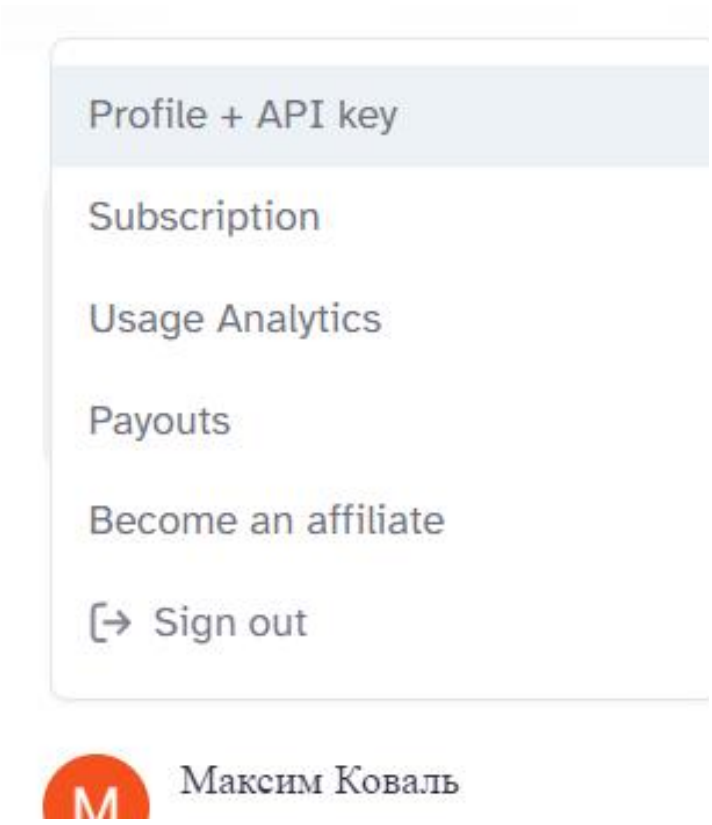


Рисунок 3.4 Створення ключа для elevenlabs

- Натискаємо кнопку "Profile + API key

- Копіюємо згенерований API ключ

Важливим аспектом є безпека ключів. Просто захардкодити їх у проект – не можна. Тому я буду використовувати їх як змінні середовища. Для того, щоб це зробити я використаю системну команду export

Приклад використання

EXPORT MY_VARIABLE="MY_VALUE"

У моємоу випадку це буде

EXPORT DEEPGRAM ="<DEEPGRAM_KEY> "

EXPORT DEEPL ="DEEPL_KEY "

EXPORT ELEVENLABS ="ELEVENLABS_KEY "

3.3 Розробка додатку

3.3.1 Конфігурації

Почнемо з файлу pom.xml, який описую проекту
Цей файл є частиною проекту і використовується системою управління залежностями Maven. Ось що я додав до цього файлі:

1. **xmlns і xsd**: Визначає простір імен та схему XML для Maven.
2. **modelVersion**: Версія моделі POM, яка використовується у цьому файлі.

3. **parent**: Вказує на батьківський POM файл, в даному випадку, це POM файл для Spring Boot starter parent версії 3.2.4. Це дозволяє успадковувати налаштування та залежності від батьківського проекту.
4. **groupId, artifactId, version, name, description**: Базові відомості про проект, такі як ідентифікатор групи, артефакт, версія, назва та опис.
5. **properties**: Визначення властивостей проекту, таких як версія Java.
6. **dependencies**: Перелік залежностей проекту. Наприклад, у нашому випадку є залежності від бібліотек httpclient, commons-lang3, spring-boot-starter, json, spring-boot-starter-test, spring-web, commons-io, spring-boot-starter-data-jpa, h2, spring-boot-starter-web, spring-data-jpa, lombok. Кожна з цих залежностей вказує на конкретну бібліотеку або фреймворк, яка використовується в проекті.

Пройдемося біль детально по кожній з залежностей

- **org.apache.httpcomponents:httpclient:4.5.13**: HttpClient - це потужна бібліотека для створення HTTP клієнтів та виконання запитів до серверів. Вона дозволяє легко взаємодіяти з веб-серверами, виконуючи HTTP запити та обробляючи відповіді.
- **org.apache.commons:commons-lang3:3.14.0**: Commons Lang - це набір утиліт для роботи з базовими операціями над рядками, масивами, числами і так далі. Вона містить багато корисних функцій, які полегшують роботу з Java.
- **org.springframework.boot**Spring Boot Starter - це пакет, який містить базові залежності для розробки додатків на Spring Boot. Він автоматизує конфігурацію та залежності Spring, що дозволяє швидко розпочати розробку.
- **org.json:json:20210307**: JSON - це простий, легкий формат обміну даними, який широко використовується для передачі структурованих даних через мережу. Ця бібліотека надає зручні інструменти для роботи з JSON в Java.

- **org.springframework.boot**Spring Boot Starter Test - це набір залежностей для тестування додатків на Spring Boot. Вона містить інструменти для написання та виконання різних видів тестів, включаючи модульні тести, інтеграційні тести та тестування з використанням фреймворку JUnit.
- **org.springframework** Spring Web - це частина Spring Framework, яка надає інструменти для розробки веб-додатків на Java. Вона допомагає взаємодіяти з HTTP запитами та відповідями, обробляти введення користувача та керувати життєвим циклом веб-додатків.
- **commons-io:commons-io:2.16.1**: Commons IO - це набір утиліт для роботи з вводом/виводом в Java. Вона містить різноманітні функції для роботи з файлами, потоками, URL і так далі, що полегшує роботу з обробкою даних вводу/виводу.

- **org.springframework.boot**
    - Spring Boot Starter Data JPA - це набір залежностей для роботи з Java Persistence API (JPA) у Spring Boot додатках. Вона дозволяє легко підключати та використовувати ORM функціональність для роботи з базами даних.

- **org.springframework.boot**
    - Spring Boot Starter Web - це пакет для розробки веб-додатків на Spring Boot. Він містить залежності та конфігурацію, необхідну для реалізації веб-сервера з використанням Spring MVC.

- **org.springframework.data**
    - Spring Data JPA - це частина Spring Framework, яка надає абстракцію відносно роботи з базами даних через JPA. Вона допомагає спростити роботу з базами даних у Spring додатках.

- **org.projectlombok:lombok:1.18.20**: Lombok - це бібліотека для роботи з анотаціями в Java, яка автоматизує створення методів доступу, конструкторів та інших шаблонних методів. Вона полегшує розробку, зменшуючи кількість шаблонного коду.

- **com.h2database**

H2 - це вбудована база даних, яка може використовуватися для розробки та тестування додатків. Вона підтримує різні режими роботи та може бути легко інтегрована в Java додатки. Для розробки я буду використовувати саме цю вбудовану базу данних, для цього зробимо правильну конфігурацію

```
spring.h2.console.enabled=true
spring.h2.console.path=/h2-ui

spring.datasource.url=jdbc:h2:file:./testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto= update
logging.level.org.springframework.web= DEBUG
logging.level.org.hibernate= ERROR
```

Рисунок 3.5 application.yaml файл

Пройдемося більш детально по конфігурації (рис. 3.5)

- **spring.h2.console.enabled=true**:
- Ця властивість вмикає консоль H2, яка надає графічний інтерфейс для взаємодії з базою даних H2 через браузер.
- **spring.h2.console.path=/h2-ui**:
- Вказує шлях, за яким можна звертатися до консолі H2 у додатку.
- **spring.datasource.url=jdbc:h2:file:./testdb**:
- Вказує URL-адресу для підключення до бази даних H2. У цьому випадку використовується вбудована база даних H2, яка зберігається в файлі testdb.
- **spring.datasource.driverClassName=org.h2.Driver**:
- Вказує назву класу драйвера JDBC для підключення до бази даних H2.
- **spring.datasource.username=sa**:
- Вказує користувача бази даних H2.

- **spring.datasource.password=**:
- Вказує пароль користувача бази даних H2. У цьому випадку пароль порожній, тобто немає пароля.
- **spring.jpa.show-sql=true**:
- Вмикає вивід SQL-запитів, які виконуються в додатку.
- **spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect**:
- Вказує діалект Hibernate для взаємодії з базою даних H2.
- **spring.jpa.hibernate.ddl-auto=update**:
- Вказує режим автоматичного оновлення схеми бази даних. У цьому випадку схема буде автоматично оновлюватися, якщо потрібно, але існуючі дані не будуть втрачені.
- **logging.level.org.springframework.web=DEBUG**:
- Вказує рівень журналювання для логування повідомлень веб-компонентів Spring у режимі DEBUG.
- **logging.level.org.hibernate=ERROR**:
- Вказує рівень журналювання для Hibernate на рівні ERROR, щоб уникнути надмірного журналювання.

3.3.2 Ініціалізація відео проекту

Почати слід з того, що потрібно завантажити проект з яким ми будемо працювати. Для цього нам необхідно створити entity клас, яким буде зберігати дані про проект. Зокрема його id, відео, транкрипт, переклад, синтез, тощо.

Давайте зробимо цей клас(рис 3.7).

Його структура буде виглядати наступним чином (рис 3.6)

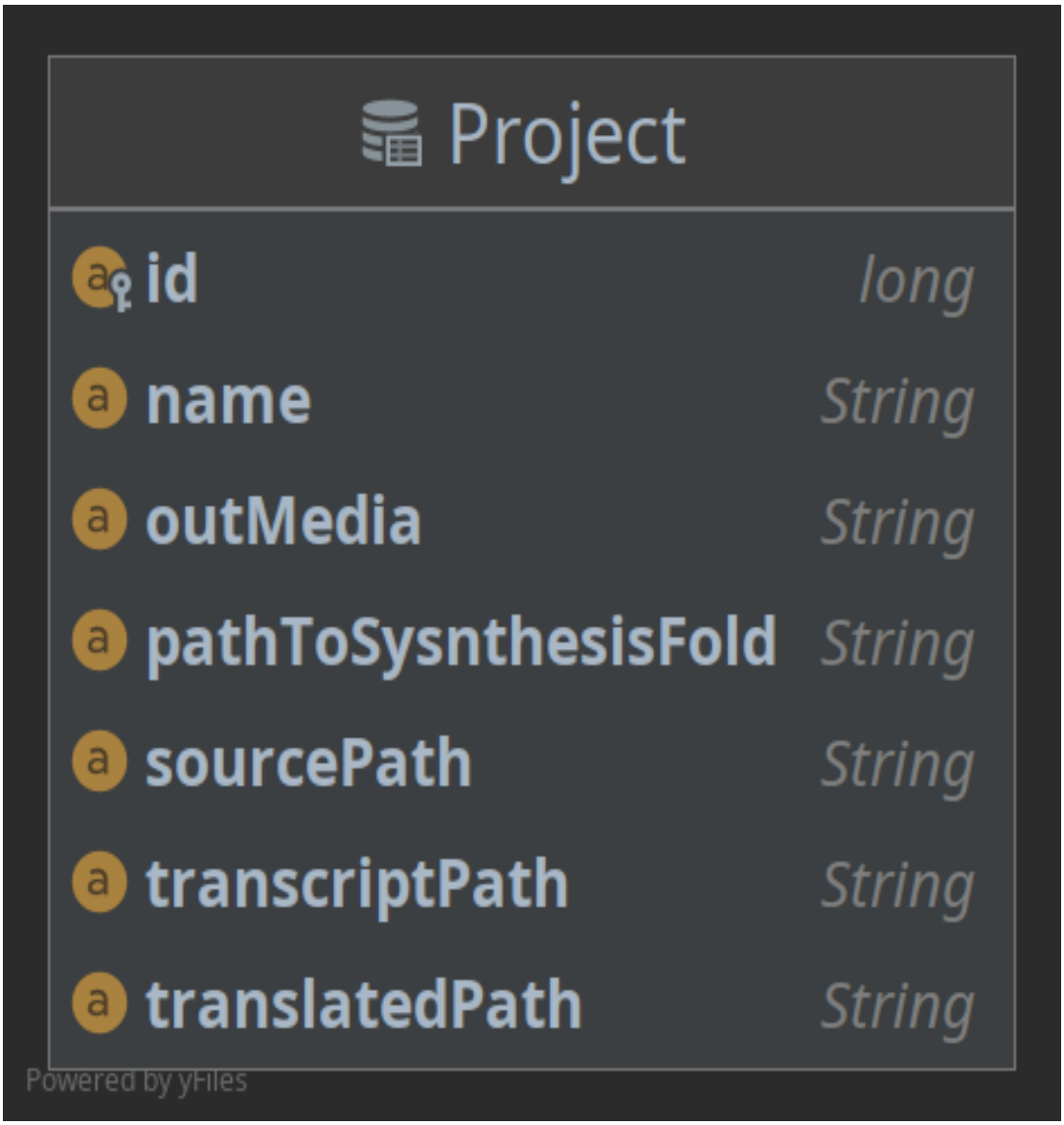


Рисунок 3.6 Структура сутності Project

Сам клас виглядатиме так:

```java
@Entity
public class Project {

    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    public long id;

    @Column(name = "name")
    public String name;

    @Column(name = "sourcepath")
    public String sourcePath;

    @Column(name = "transcriptpath")
    public String transcriptPath;

    @Column(name = "translatedPath")
    public String translatedPath;

    @Column(name = "pathToSysnthesisFold")
    public String pathToSysnthesisFold;

    @Column(name = "outmedia")
    public String outMedia;


    public Project (Long id, String name, String sourcePath) {
        this.id = id;
        this.name = name;
        this.sourcePath = sourcePath;
    }

    public Project () {

    }
```

Рисунок 3.7 Клас Project

Кожне з полів, будуть застосовані далі, при розробці застосунку у майбутніх сервісах.

Також важливо імплементувати метод (рис 3.8), який буде зберігати заватажену вхідну медіа, щоб у майбутньому з нею взаємодіяти

```
    public Path putMedia(InputStream stream, String fileName, String dir) {
        Path filePath = Paths.get(dir, fileName);
        try (OutputStream fileStream = Files.newOutputStream(filePath)) {
            IOUtils.copyLarge(stream, fileStream);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return filePath;
    }
```

Рисунок 3.8 Метод putMedia

Важливо, щоб ця медіа була прив’язана до ID проекту.

Тепер тут достатньо інструментів, для створення першого RestController. В застосунку його завданням будуть створення проекту та завантаження відео.

```
@RestController
@RequestMapping("project")
public class VideoController {

    @Autowired
    MediaService mediaService;

    @Autowired
    ProjectService projectService;

    @PutMapping("project")
    public ResponseEntity<Long> createProject() {
        Project project = new Project();
        projectService.putProject(project);
        return ResponseEntity.ok(project.id);
    }

    @PutMapping("{id}/video")
    public void putMedia(
            @PathVariable("id")
                    String id,
                    InputStream stream) {
        Project project = projectService.getProject(id);
        project.sourcePath = mediaService.putMedia(stream, id, MediaService.DEFAULT_DIR).toString();
        System.out.println(project.sourcePath);
        projectService.putProject(project);
    }
}
```

Рисунок 3.9 Відео контроллер

Зараз зупинимось на ньому більш детально (рис 3.9)

Отже, данний клас є першим котролером веб-застосунку. В цьому класі є декілька анотацій, на яких варто зупинитись.

➓ **@RestController**:

Ця анотація позначає клас як контролер, який обробляє HTTP запити та повертає дані в форматі JSON . Всі методи в цьому класі, які повертають дані, будуть автоматично серіалізовані відповідно до встановленого контенту типу.

- **@RequestMapping("project")**:

  Ця анотація вказує базовий шлях для всіх методів у цьому контролері. Всі HTTP запити, які мають шлях "/project", будуть оброблятися цим контролером.

- **@Autowired**:

  Ці анотації позначають поля, які повинні бути автоматично внедрені Spring контейнером залежностей. В даному випадку mediaService і projectService будуть автоматично ініціалізовані Spring бінами, які реалізують відповідні інтерфейси.

- **@PutMapping("project")**:

  Цей метод обробляє HTTP PUT запити до шляху "/project". Він створює новий проект, викликаючи метод putProject() сервісу projectService, і повертає статус успішного виконання разом з ідентифікатором нового проекту.

- **@PutMapping("{id}/video")**:

Цей метод обробляє HTTP PUT запити до шляху "/{id}/video", де {id} є змінною частиною шляху. Він приймає id проекту та вхідний потік (InputStream) відео. Потік тут представляє вхідні дані відеофайлу. Метод витягує проект з вказаним id, зберігає вхідне відеофайл до відповідного медіа-сховища через mediaService та оновлює шлях до відеофайлу в об'єкті проекту перед збереженням за допомогою projectService.

3.3.3 Транскрибування

Наступним контроллером в застосунку буде TranscriptController

Його завданням буде створити транскрипт до завантаженого відео

Для цього необхідно три сервіси.

- TranscribeService
- VttConvetor

Почнемо з TranscribeService

Цей сервій має у собі основний метод transcribeAudio(String path, String language) (рис. 3.10)

Цей метод призначений для транскрибації аудіофайлу за допомогою сервісу Deepgram.

transcribeAudio(String path, String language) :

- Цей метод отримує два параметри: path, який є шляхом до аудіофайлу, який потрібно транскрибувати, та language, який вказує мову аудіофайлу.
- Викликається внутрішнім методом sendRequest(path, URL, language), який відправляє запит на транскрибацію аудіофайлу до сервісу Deepgram і повертає відповідь у вигляді об'єкту AsrResponse.
- Повертає транскрибований текст, отриманий з відповіді Deepgram, в форматі String.

```
@SneakyThrows
public String transcribeAudio (String path, String language) {
    AsrResponse asrResponse = sendRequest(path, URL, language);
    return getResponse(asrResponse);
}
```

Рисунок 3.10 Метод Transcribe Audio

```
public static String convertToSRT(List<Word> words) {
    StringBuilder srtBuilder = new StringBuilder();
    int index = 1;
    int wordIndex = 0;
    StringBuilder sentenceBuilder = new StringBuilder();
    boolean newChunk = true;
    String chunkStartTime = null;
    while (wordIndex < words.size()) {
        String wordStr = words.get(wordIndex).getPunctuatedWord();
        double startTime = words.get(wordIndex).getStart();
        double endTime = words.get(wordIndex).getEnd();
        if (newChunk) {
            chunkStartTime = formatTime(startTime);

            newChunk = false;
        }
        sentenceBuilder.append(wordStr).append(" ");
        if (wordStr.endsWith(".") || wordStr.endsWith("?") || wordStr.endsWith("!") || sentenceBuilder.toString().length() > 150)
        {
            srtBuilder.append(index++).append("\n");
            srtBuilder.append(chunkStartTime).append(" --> ").append(formatTime(endTime)).append("\n");
            srtBuilder.append(sentenceBuilder.toString().trim()).append("\n\n");
            sentenceBuilder = new StringBuilder();
            newChunk = true;
        }
        wordIndex++;
    }

    return srtBuilder.toString();
}
```

Рисунок 3.11 Конвертація у SRT формат

Основна мета цього методу - забезпечити можливість транскрибувати аудіофайли з використанням сервісу Deepgram та отримувати результат у форматі SRT для подальшого використання.

VttConvertor

Ще одним аспектом застомунку є підтримка vtt формату

Підтримка формату WebVTT (VTT) може бути корисною з декількох причин:

1. **Субтитри для відео**: Формат VTT використовується для створення субтитрів для відео. Він дозволяє вставляти текстові блоки (субтитри) у відеофайли, що дозволяє глядачам зручно читати текстову інформацію разом із відео.
2. **Підтримка асинхронного тексту**: VTT може бути використаний для відображення тексту, який відображається асинхронно з відео. Наприклад, це може бути корисно для відображення субтитрів, які з'являються на екрані у відповідь на певні події або дії відео.
3. **Синхронізація з аудіо та відео**: Формат VTT може містити часові мітки, які вказують час початку та закінчення кожного субтитру. Це дозволяє

синхронізувати субтитри з аудіо та відео контентом, що полегшує користувачам відтворення відео з субтитрами.

4. **Веб-сторінки та HTML5 відеопрогравачі**: Формат VTT широко підтримується сучасними веб-браузерами та HTML5 відеопрогравачами. Він може бути вбудований безпосередньо у HTML сторінки за допомогою відповідного тегу <track>.
5. **Міжнародна підтримка**: Формат VTT підтримує різні мови та символи, що дозволяє створювати субтитри для відео на різних мовах і з різними алфавітами.

Для цього є клас, яким вміє конвертувати srt формат в vtt(рис 3.12)

Він виглядає наступним чином

```
public static String srtToVtt(String srtContent) {
    StringBuilder vttContent = new StringBuilder();
    vttContent.append("WEBVTT").append("\n\n");
    String[] lines = srtContent.split( regex: "\n");
    for (String line : lines) {
        if (line.matches( regex: "^\\d+$")) {
            continue;
        } else if (line.matches( regex: "^\\d{2}:\\d{2}:\\d{2},\\d{3}\\s-->\\s\\d{2}:\\d{2}:\\d{2},\\d{3}$")) {
            vttContent.append(line.replace( target: ",", replacement: ".")).append("\n");
        } else if (!line.isEmpty()) {
            vttContent.append(line).append("\n\n");
        }
    }
    return vttContent.toString();
}
```

Рис 3.12 Конвертація SRT до TT

Отже, в цілому контроллер матиме наступний вигляд (рис.3.13)

```java
@SneakyThrows
@PutMapping(path = "/{id}")
public void transcribe(
        @PathVariable("id")
                String id,
        @RequestParam("lang")
                String language) {
    Project project = projectService.getProject(id);
    project.transcriptPath = MediaService.DEFAULT_DIR + "/" + id + "transcript.srt";
    String srt = transcribeService.transcribeAudio(project.sourcePath, language);
    FileUtils.write(new File(project.transcriptPath), srt);
    projectService.putProject(project);
}


@SneakyThrows
@PutMapping("{id}/srt")
public void putSrt(
        @PathVariable("id")
                String id,
        String srt) {
    FileUtils.write(new File(projectService.getProject(id).transcriptPath), srt);
}

@SneakyThrows
@GetMapping("{id}/srt")
public ResponseEntity<String> getSrt(
        @PathVariable("id")
                String id) {
    String transcriptPath = projectService.getProject(id).transcriptPath;
    return ResponseEntity.ok(FileUtils.readFileToString(new File(transcriptPath)));
}


@SneakyThrows
@GetMapping("{id}/vtt")
public ResponseEntity<String> getVtt(
        @PathVariable("id")
                String id) {
    String transcriptPath = projectService.getProject(id).transcriptPath;
    String srt = FileUtils.readFileToString(new File(transcriptPath));
    String vtt = VttConvertor.srtToVtt(srt);
    return ResponseEntity.ok(vtt);
```

Рисунок 3.13 Методи Transcribe контроллера

3.3.4 Переклад

Наступним контроллером в застосунку буде TranslationController

Він міститиме в собі TranslationService та SrtService

Його призначенням буде взаємодія з перекладом. Переклад буде відбуватися для кожного SRT/VTT сегмента окремо, для збереженням часових поміток, які були отриманні від TranscribeService. TranscribeService базуватиметься на DeepL api. Для збереження контексту, також до api буде надсилатися попередній сегмент. Вигляд це матиме такий

```
public String translate(String text, String from, String to) throws Exception {
    Request request = new Request(from, to, Collections.singletonList(text), context);
    context = text;
    try {
        HttpClient client = HttpClients.createDefault();
        HttpPost httpPost = createPostRequest(request);
        HttpResponse response = client.execute(httpPost);
        return getTranslations(response).getTexts().get(0);
    }
    catch (IOException | URISyntaxException e) {
        throw new Exception(e.getMessage());
    }
}
```

Рисунок 3.15 Метод перекладу

Для роботи з Srt будуть застосовані наступні методи.

Вони дозволять проводити десереалзацію srt тексту до сегментів (у вигляді java об’єктів)

```java
public static List<Segment> parseSrt(String srtContent) {
    List<Segment> captions = new ArrayList<>();
    String[] lines = srtContent.split( regex: "\n");

    int i = 0;
    while (i < lines.length) {
        String indexLine = lines[i].trim();
        i++;

        if (i < lines.length && !indexLine.isEmpty() && indexLine.matches( regex: "\\d+")) {
            String timeLine = lines[i].trim();
            i++;

            if (timeLine.contains("-->")) {
                String[] times = timeLine.split( regex: "-->");
                Duration startTime = parseDuration(times[0].trim());
                Duration endTime = parseDuration(times[1].trim());
                StringBuilder textBuilder = new StringBuilder();

                while (i < lines.length && !lines[i].trim().isEmpty()) {
                    textBuilder.append(lines[i].trim()).append("\n");
                    i++;
                }

                String text = textBuilder.toString().trim();
                Segment segment = new Segment();
                segment.startTime = startTime;
                segment.endTime = endTime;
                segment.text = text;
                captions.add(segment);
            }
        }

        while (i < lines.length && lines[i].trim().isEmpty()) {
            i++;
        }
    }

    return captions;
}
```

Рисунок 3.16 Конвертація з SRT у сегменти

І навпаки

```java
public static String convertSegmentsToSrt(List<Segment> segments) {
    StringBuilder srtContent = new StringBuilder();

    for (int i = 0; i < segments.size(); i++) {
        Segment segment = segments.get(i);
        srtContent.append(i + 1).append("\n");
        srtContent.append(formatDuration(segment.startTime)).append(" --> ").append(formatDuration(segment.endTime)).append("\n");
        srtContent.append(segment.text).append("\n\n");
    }

    return srtContent.toString().trim();
}
```

Рисунок 3.17 Конвертація сегментів у SRT

Основний метод контроллеру матиме такий вигляд

```
@Autowired
TranslateService translateService;

@Autowired
ProjectService projectService;


@PutMapping(path = "/{id}")
public void translate(
        @PathVariable("id")
                String id,
        @RequestParam("from")
                String fromLanguage,
        @RequestParam("to")
                String toLanguage) throws Exception {
    Project project = projectService.getProject(id);
    project.translatedPath = MediaService.DEFAULT_DIR + "/" + id + "translated.srt";
    String sourceSrt = FileUtils.readFileToString(new File(project.transcriptPath));
    List<Segment> sourceSegments = parseSrt(sourceSrt);
    for (int i = 0 ; i < sourceSegments.size() ; i++) {
        sourceSegments.get(i).text = translateService.translate(sourceSegments.get(i).text, fromLanguage, toLanguage);
    }
    String srt = convertSegmentsToSrt(sourceSegments);
    FileUtils.write(new File(project.translatedPath), srt);
    projectService.putProject(project);
}
```

Рисунок 3.18 Контроллер перекладу

Отже, в цьому методі відбуваються наступні дії

- **Отримання проекту за ідентифікатором**:
- Метод отримує ідентифікатор проекту як шляхову змінну {id} з URL. Він викликає сервіс projectService.getProject(id) для отримання даних проекту за вказаним ідентифікатором.
- **Створення перекладених субтитрів**:
- Спочатку метод формує шлях до файлу перекладених субтитрів для даного проекту, який зберігається в полі translatedPath проекту. Цей шлях формується шляхом конкатенації шляху за замовчуванням та ідентифікатора проекту, а також додавання суфіксу "translated.srt".
- **Читання початкових субтитрів**:
- Метод читає вміст початкових субтитрів з файлу, вказаного у полі transcriptPath проекту, за допомогою об'єкта FileUtils.

- **Парсинг субтитрів**:
- Текст початкових субтитрів парситься для отримання списку сегментів за допомогою методу parseSrt(). Кожен сегмент представляє собою текст та часові мітки початку і закінчення субтитру.
- **Переклад субтитрів**:
- Для кожного сегмента тексту викликається метод translate() сервісу translateService, який виконує переклад тексту з мови fromLanguage на мову toLanguage.
- **Створення перекладених субтитрів**:
- Після перекладу тексту всі сегменти знову конвертуються у формат SRT за допомогою методу convertSegmentsToSrt() та записуються у файл перекладених субтитрів за допомогою FileUtils.
- **Оновлення проекту**:
- Зміни у проекті, такі як шлях до перекладених субтитрів, зберігаються та оновлюються за допомогою сервісу projectService.putProject(project).

3.3.5 Синтез та отримання результату

Для отримання фінального результату потрібно вставити згенеровані субтитри у медіа файл, а також згенерувати синтезовані дубляжі.

Для вшивання субтитрів буде використанно Ffmpeg.

За допомогою команди subtitles

Її ми завернемо в java. У клас FfmpegWrapper.

Настпий етапом є зменшення звуку у вхідному файлі. Це необхідно для того, щоб згенеровані дубляжі було краще чути.

Це ми так само зробимо за допомогою ffmpeg

А саме завдяки команді

-filter:a

Після цього все готове для генерації дубляжівДля цього буде використано Elevenlabs. Формат для результатів буде **PCM 44_100**.

Також для генерації аудіо буде використано параметер previous_text.

Це дозволить використати для генерації аудіо контекст з попереднім текстом

Після отримання результатів від Elevenlabs, потрібно буде їх закодувати у **WAV** формат для подальшого використання.

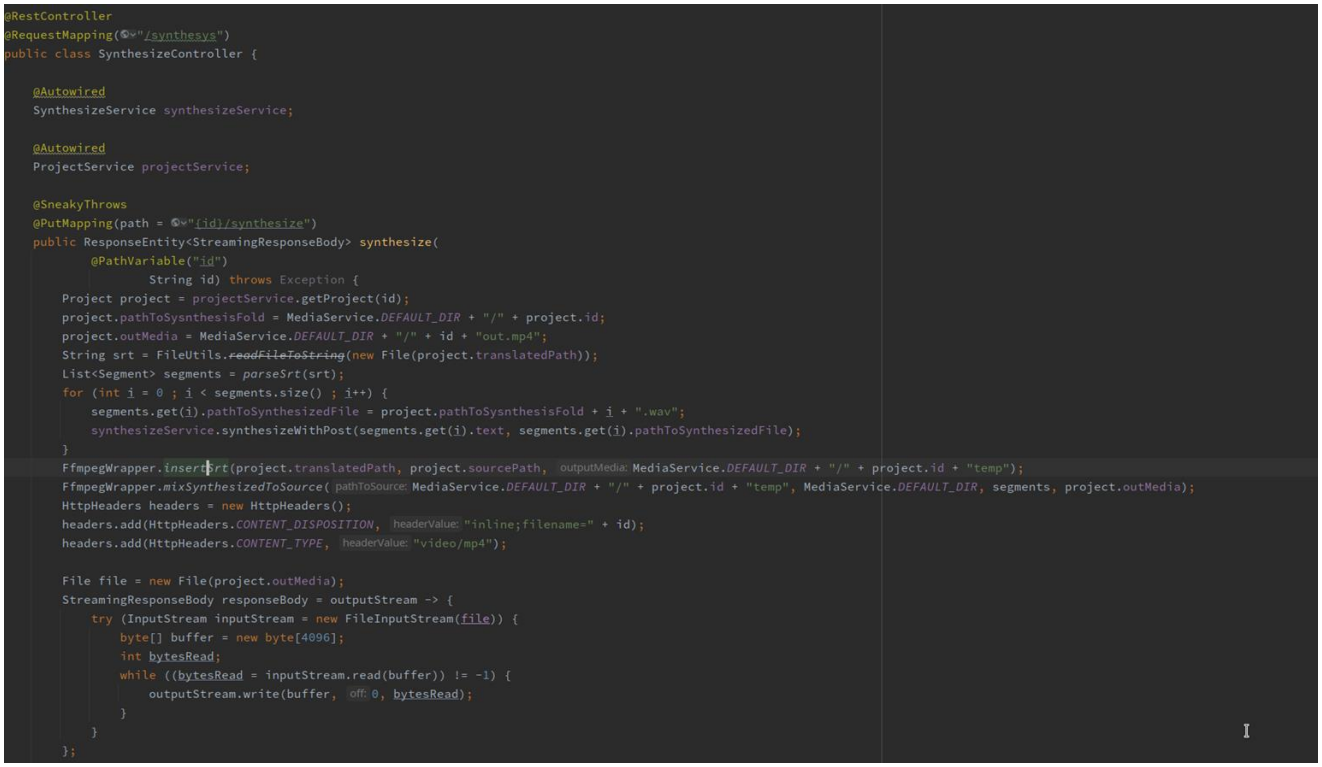
Останьою дією буде міксування згенерованого аудіо з вхідним відео.

```
StringBuilder command = new StringBuilder();
command.append("ffmpeg -y").append(" ");
command.append("-i").append(" ");
command.append(srcMedia).append(" ");

for (Segment tgt : segments) {
    command.append("-i").append(" ");
    command.append(tgt.pathToSynthesizedFile).append(" ");
}
StringBuilder filterComplex = new StringBuilder();
for (int i = 0; i < segments.size(); i++) {
    long delayMillis = segments.get(i).startTime.toMillis();
    filterComplex.append("[").append(i + 1).append("]adelay=").append(delayMillis).append("|").append(delayMillis).append("[s").append(i + 1)
}
filterComplex.append("[0]");
for (int i = 0; i < segments.size(); i++) {
    filterComplex.append("[s").append(i + 1).append("]");
}
filterComplex.append("amix=inputs=").append(segments.size() + 1).append("[a]");
command.append("-filter_complex").append(" ");
command.append(filterComplex.toString()).append(" ");
command.append("-map").append(" ");
command.append("0:v").append(" ");
command.append("-map").append(" ");
command.append("[a]").append(" ");
command.append("-c:v").append(" ");
command.append("copy").append(" ");
command.append("-c:a").append(" ");
command.append("aac").append(" ");
command.append(outMedia).append(" ");
```

Рис 3.19 Команда для міксування відео з синтезованими аудіо

Сам контроллер матиме наступний вигляд (рис. 3.20)



```
@RestController
@RequestMapping("/synthesys")
public class SynthesizeController {

    @Autowired
    SynthesizeService synthesizeService;

    @Autowired
    ProjectService projectService;

    @SneakyThrows
    @PutMapping(path = "{id}/synthesize")
    public ResponseEntity<StreamingResponseBody> synthesize(
            @PathVariable("id")
                    String id) throws Exception {
        Project project = projectService.getProject(id);
        project.pathToSysnthesisFold = MediaService.DEFAULT_DIR + "/" + project.id;
        project.outMedia = MediaService.DEFAULT_DIR + "/" + id + "out.mp4";
        String srt = FileUtils.readFileToString(new File(project.translatedPath));
        List<Segment> segments = parseSrt(srt);
        for (int i = 0 ; i < segments.size() ; i++) {
            segments.get(i).pathToSynthesizedFile = project.pathToSysnthesisFold + i + ".wav";
            synthesizeService.synthesizeWithPost(segments.get(i).text, segments.get(i).pathToSynthesizedFile);
        }
        FfmpegWrapper.insertSrt(project.translatedPath, project.sourcePath,  outputMedia: MediaService.DEFAULT_DIR + "/" + project.id + "temp");
        FfmpegWrapper.mixSynthesizedToSource( pathToSource: MediaService.DEFAULT_DIR + "/" + project.id + "temp", MediaService.DEFAULT_DIR, segments, project.outMedia);
        HttpHeaders headers = new HttpHeaders();
        headers.add(HttpHeaders.CONTENT_DISPOSITION,  headerValue: "inline;filename=" + id);
        headers.add(HttpHeaders.CONTENT_TYPE,  headerValue: "video/mp4");

        File file = new File(project.outMedia);
        StreamingResponseBody responseBody = outputStream -> {
            try (InputStream inputStream = new FileInputStream(file)) {
                byte[] buffer = new byte[4096];
                int bytesRead;
                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer,  off: 0, bytesRead);
                }
            }
        };
```

Рисунок 3.20 Контроллер синтезу

Висновки до розділу 3

Отже, для розробки застосунку перш за все розібралися з технологією ffmpeg, яку використали для вшиття субтитрів, зміни звуку, міксування генерованих аудіо з відео.

Також створили наступні контроллери

- VideoController
- TranscriptController
- TranslateController
- SynthesizeController

Кожен з яких відповіє за певні дії

Імплементували наступні сервіси

- VideoService
- ProjectService
- FfmpegWrapper
- TranscribeService
- TranslateService
- SynthesizeService
- SrtService
- VttConvertor

Окрім цього, було реалізовано Project entity.

Сутність, яка є основною імплементацією створюваного проекту.

Реалізували взаємодії між ними завдяки можливостям Spring.

Внаслідок цих дій застосунок повністю готови до перевірок та тестування на предмет задоволення вимог.

РОЗДІЛ 4. ПЕРЕВІРКА ФУНКЦІОНАЛЬНОСТІ СИСТЕМИ

Необхідно перевірити на коректність та повноту функцій кожен компонент розробленої системи. Перевірка буде здійснена поетапно для кожного типу користувача.

4.1 Перевірка працездатності системи

Для початку створемо проект

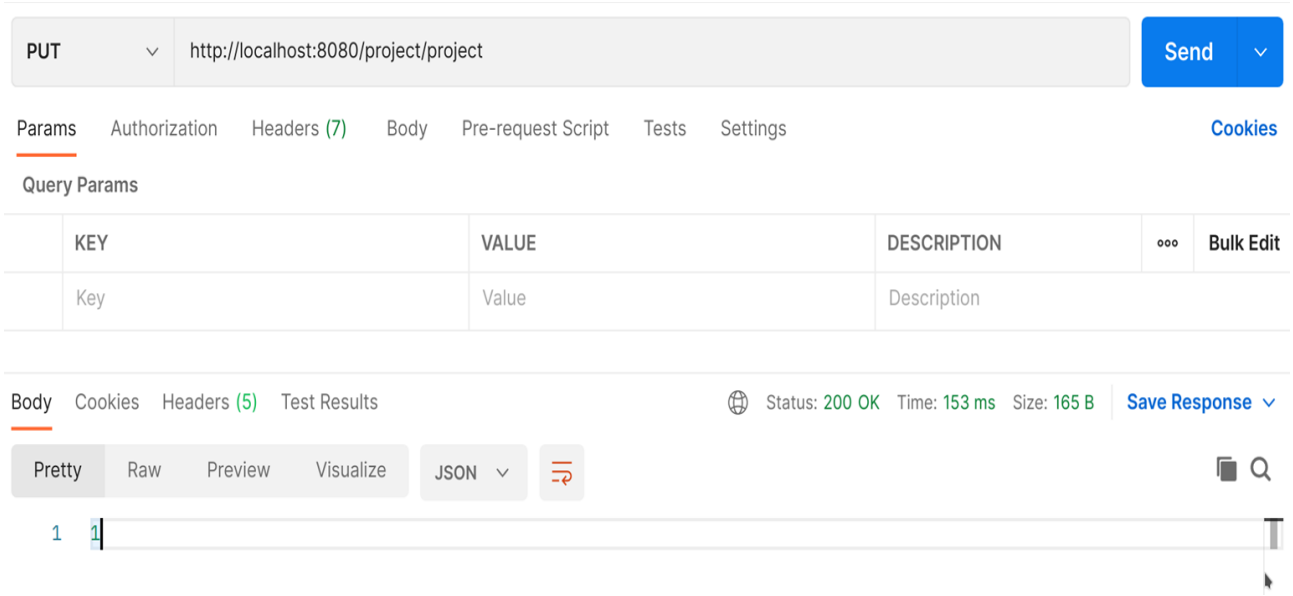


Рисунок 4.1 Створення проекту

Як відповідь нам надійшло його id

Після цього завантажимо файл з відео

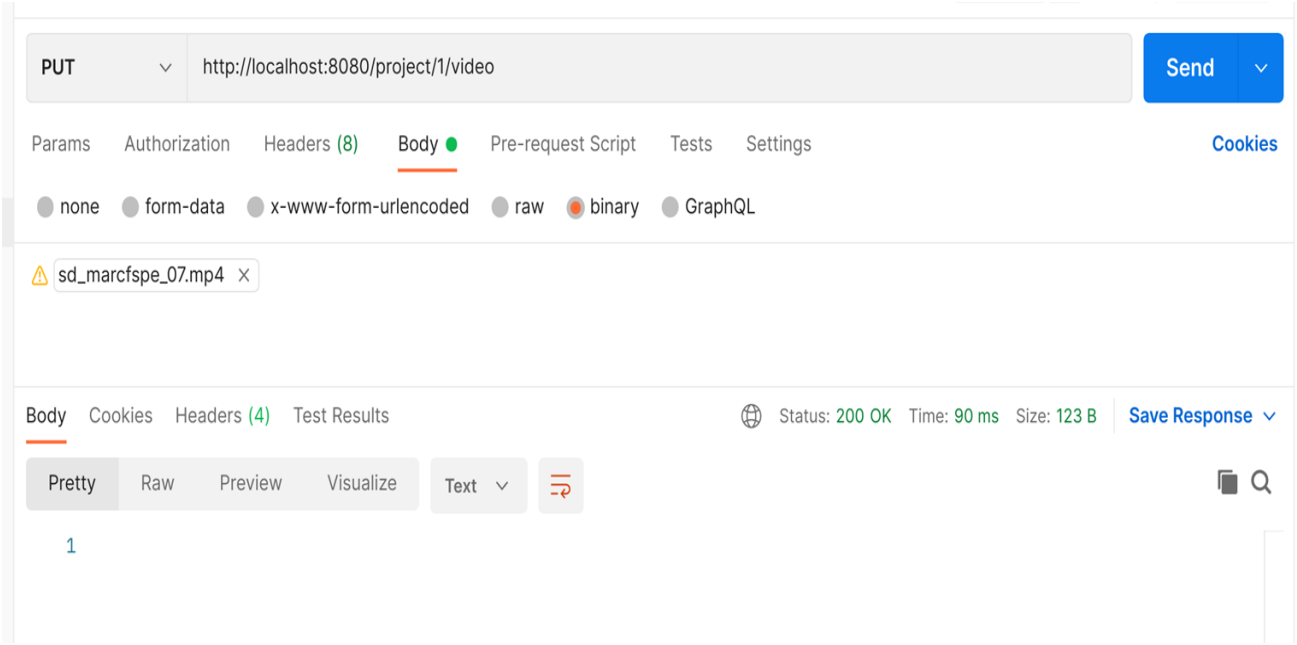


Рисунок 4.2 Завантаження відео

Далі генеруємо транскрипт для завантежонго відео

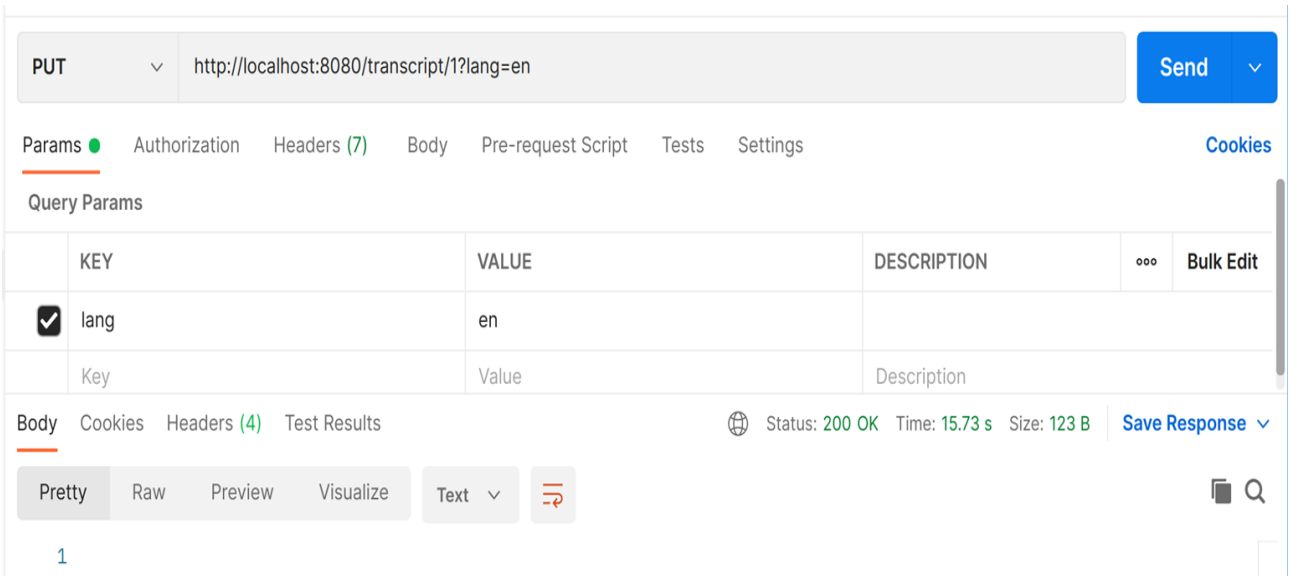


Рисунок 4.3 Генерація транскрипту

Тепер спробуємо отримати згенеровані підписи у форматі SRT

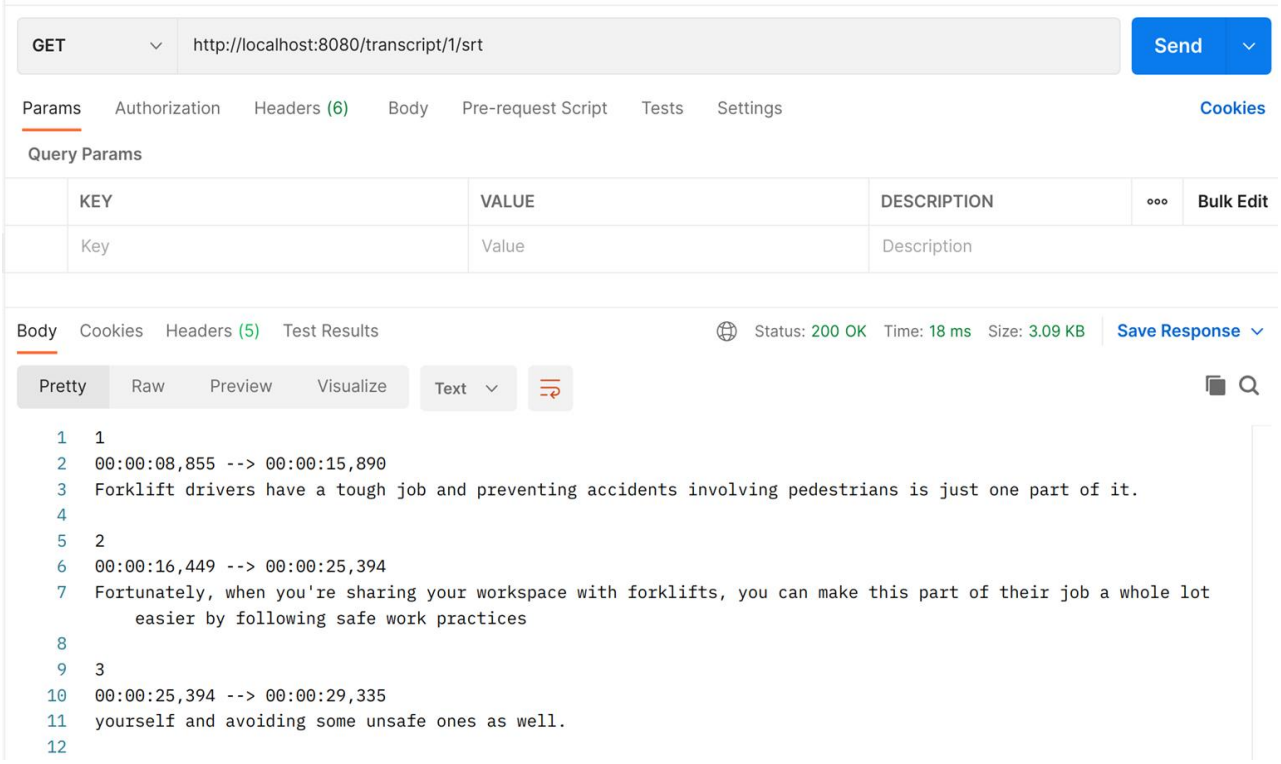


Рисунок 4.4 Отримання SRT транскрипту

та VTT

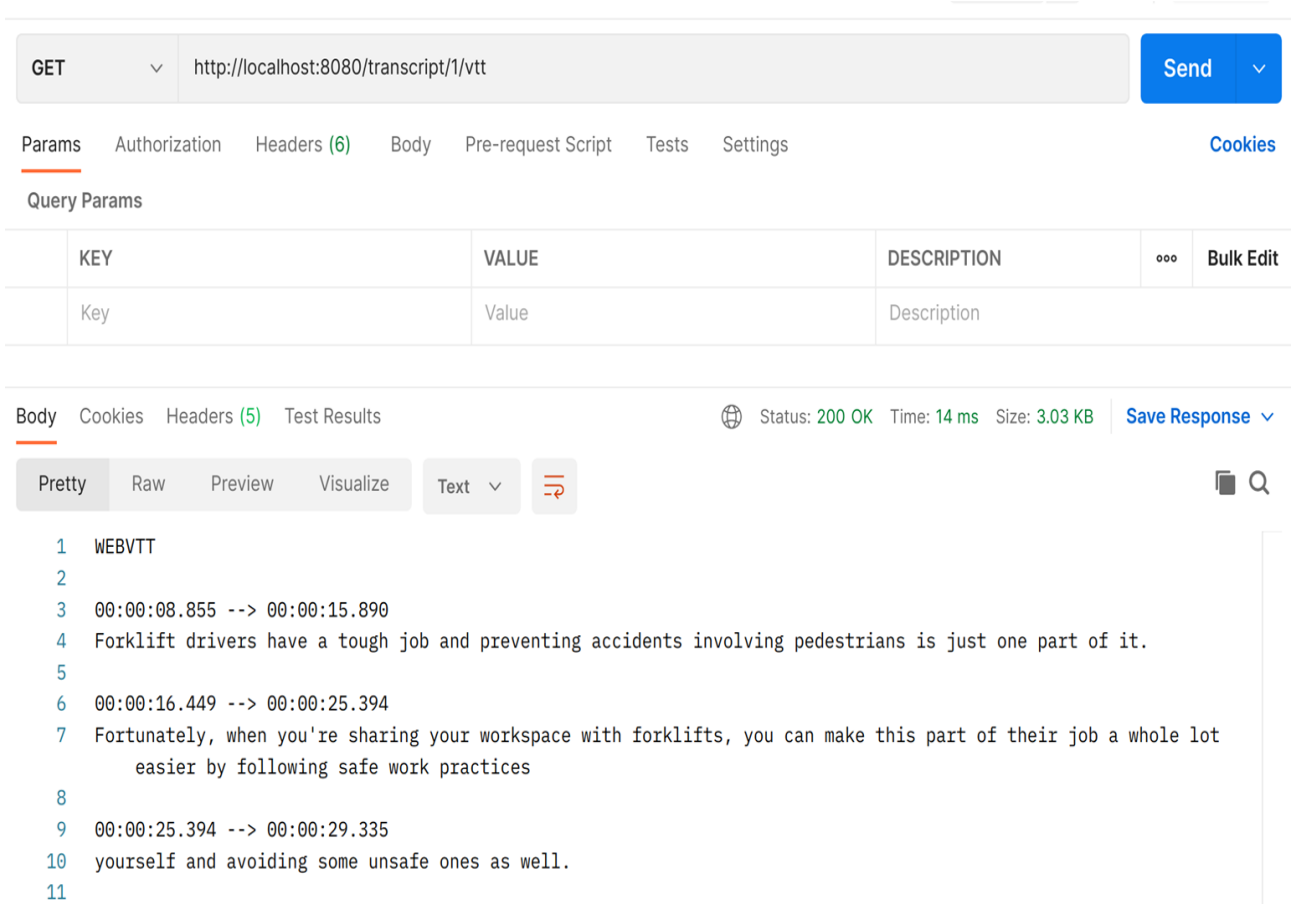


Рисунок 4.5 Отримання VTT транскрипту

Тепер перекладемо на укрїнську мову

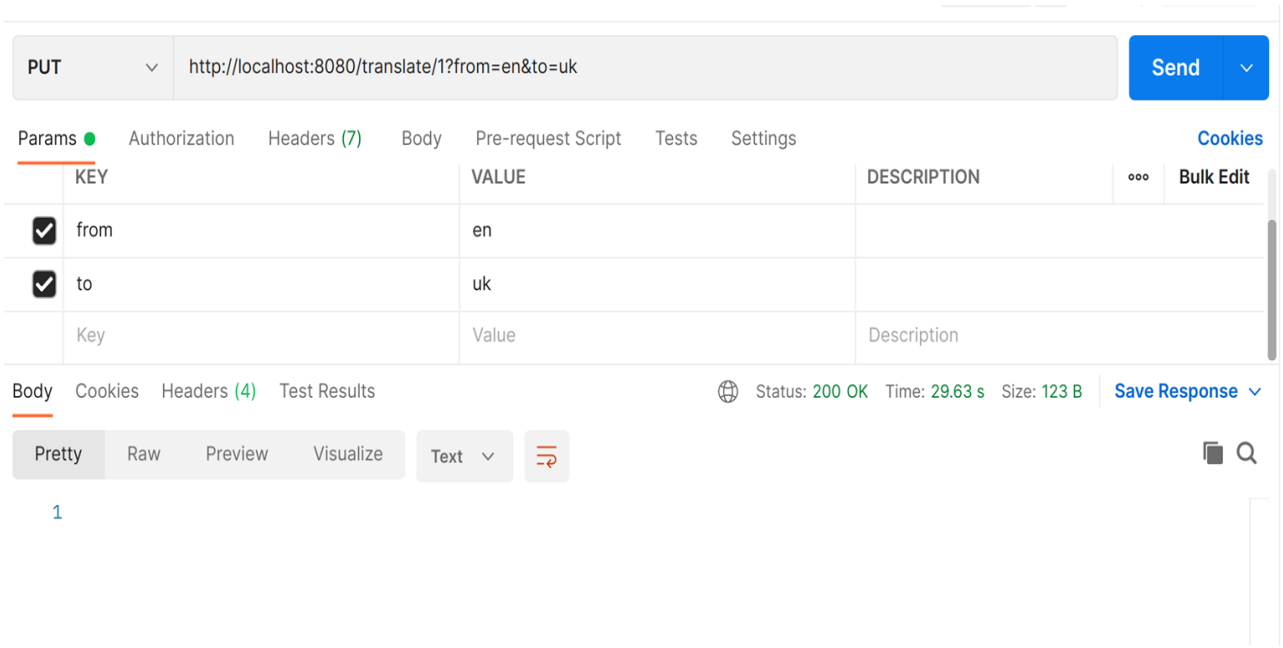


Рисунок 4.6 Переклад

Переглянемо результати у SRT та VTT форматах

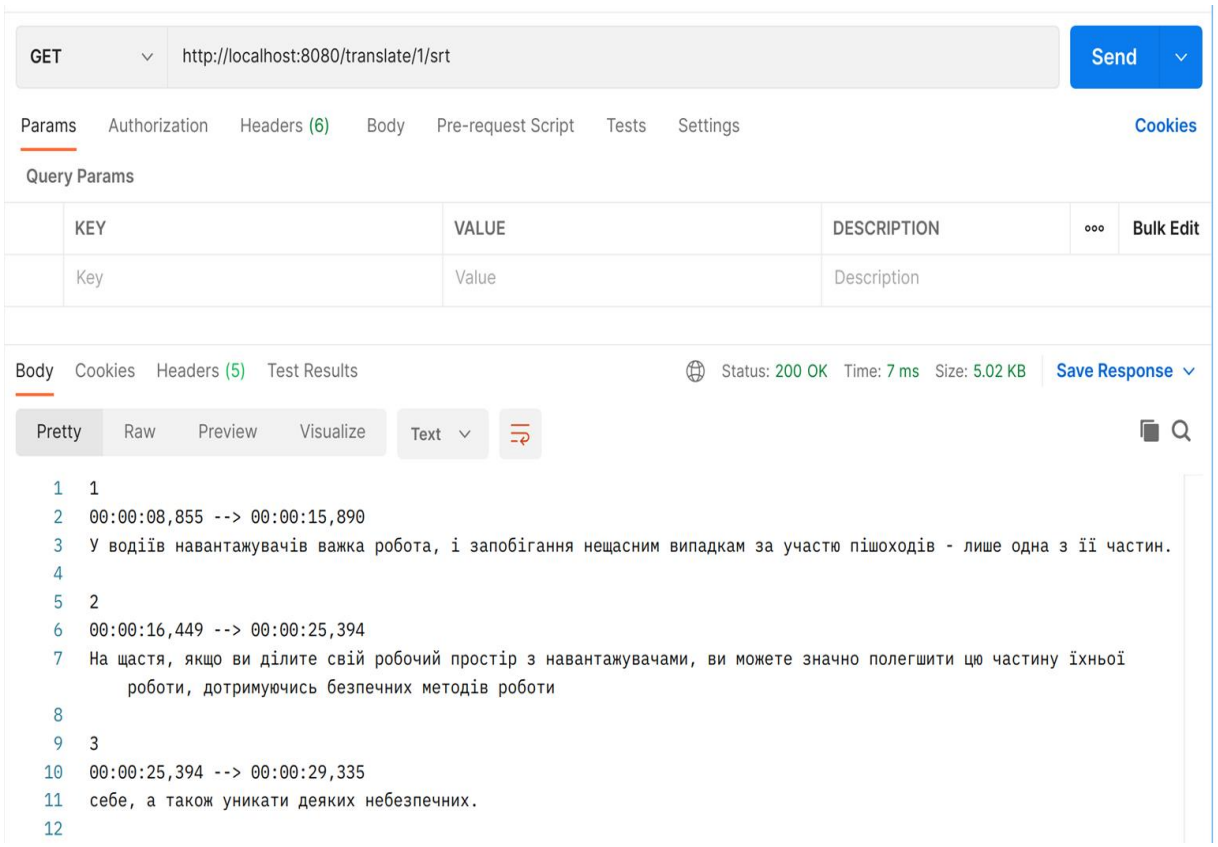


Рисунок 4.7 Переклад у SRT форматі

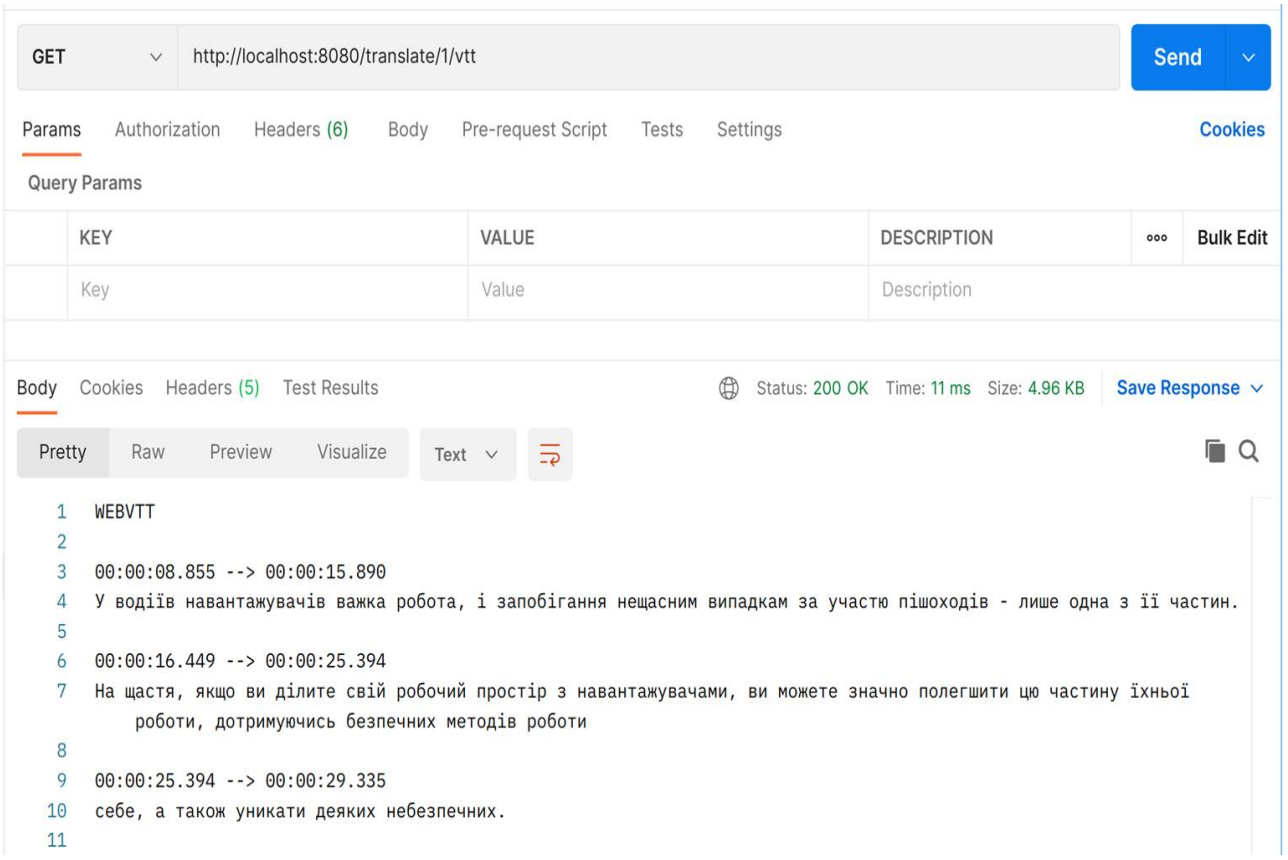


Рисунок 4.8 Переклад у VTT форматі

І спробуємо згенерувати готову медіа з вшитими субтитрами та дубляжами

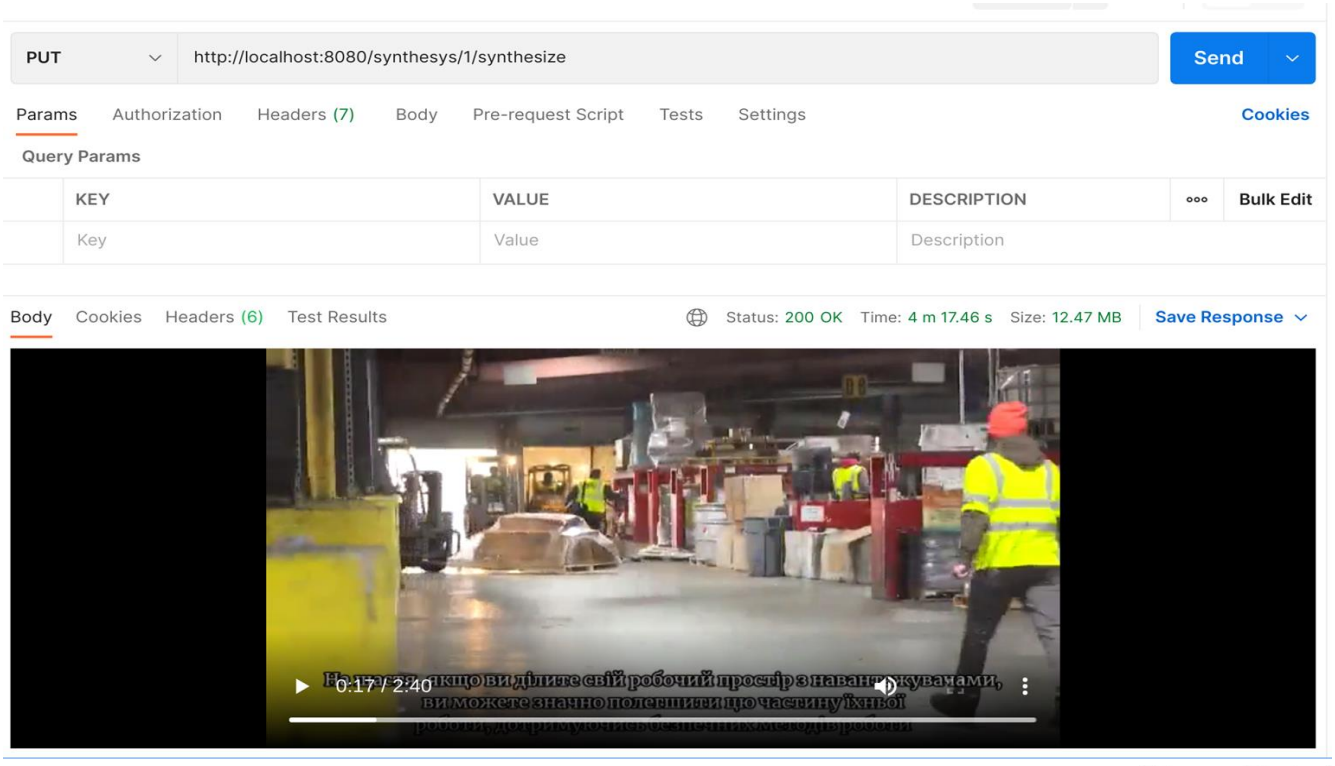


Рисунок 4.9 Синтез та отримання результату

Висновки до розділу 4

Було проведено перевірку всього застосунку, а саме логіку створення проекту, завантаження медіа файлу, траскрибацію, переклад, генерацію субтитрів у VTT та SRT форматах, синтез мовлення та вшивання субтитрів. Веб застосунок продемонстрував стабільну та очікувану роботу. Всі компоненти пройшли необхідні тестування та відповідають вимогам до функціональності та якості. Всі вимоги вважаються задоволеними. Результатом успішного завершення проекту є розроблений та готовий до використання продукт. Система функціонує стабільно, є зручною та надійною у використанні. Успішне завершення проекту підтверджує правильність проведеного аналізу, постановки задач, вимог, підходу до розробки та вибраних технологій, що дозволило створити ефективний продукт.

Висновки

Отже, розробка веб застосунку для транскрибування, перекладу та синтезу тексту є важливим кроком у напрямку підвищення доступності інформації для людей з обмеженими можливостями та подолання мовних бар'єрів між різними народами. Цей проект є важливим внеском у соціальну інтеграцію та комунікаційні можливості в сучасному глобалізованому світі. Проект спрямований на розв'язання важливих соціальних проблем. Люди з обмеженими можливостями тепер можуть легше отримувати інформацію з різних джерел, а автоматичний переклад сприяє культурному обміну та взаєморозумінню між народами. Таким чином, цей додаток сприяє інклюзивності та взаємоповазі у глобальному суспільств.

Також, варто віддати належне технологіям, які були використанні при розробці застосунку. Зокрема Java. Мові програмування, яка має багату екосистему бібліотек і фреймворків, які значно спростили розробку веб додатку. Наприклад, використання Spring фреймворку дозволило легко управляти залежностями, налаштовувати додаток і забезпечувати його надійність та безпеку.

Варто додати, що застосунок задовольнив умови підтримки основних мов. Можливість працювати з багатьма мовами є критично важливою для додатку, який призначений для подолання мовних бар'єрів та забезпечення доступності інформації. Підтримка багатомовності дозволяє користувачам з різних країн та культур легко взаємодіяти з додатком. Це включає не лише транскрибування та переклад тексту, але й синтез мовлення різними мовами, що є особливо корисним для міжнародної аудиторії.

Ще однією умовою була підтримка форматів SRT та VTT. Нагадаю, що підтримка форматів субтитрів SRT та VTT є важливою для забезпечення сумісності додатку з широким спектром медіаплеєрів та відеоплатформ. SRT та VTT є найбільш популярними форматами субтитрів, які підтримуються багатьма відеосервісами, включаючи YouTube, Vimeo, та HTML5 відеоплеєри. Це

забезпечує користувачам можливість легко експортувати та використовувати створені субтитри для своїх відео на будь-якій платформі. З цією умовою проекту впорався також успішно.

Розроблений додаток демонструє, як сучасні технології можуть бути ефективно використані для вирішення важливих соціальних проблем. Використання Java, Spring, FFmpeg, SRT та VTT забезпечило створення потужного, гнучкого та функціонального інструменту, що може значно покращити якість життя багатьох людей. Цей проект є яскравим прикладом того, як технології можуть об'єднувати людей, роблячи світ більш доступним та взаємопов'язаним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Java [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/
2. Документація JDK [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/javase/17/
3. Документація JVM [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/javase/17/docs/specs/
4. Документація Spring [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring/docs/current/spring-framework-reference/
5. Документація Spring Boot [Елекстронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
6. Документація Spring Data JPA [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-data/jpa/reference/jpa.html
7. Документація H2 Database [Електронний ресурс] – Режим доступу до ресурсу: https://h2database.com/html/main.html
8. Документація FFmpeg [Електронний ресурс] – Режим доступу до ресурсу: https://ffmpeg.org/documentation.html
9. Документація SRT [Електронний ресурс] – Режим доступу до ресурсу: https://www.srtalliance.org/technology
10. Документація VTT [Електронний ресурс] – Режим доступу до ресурсу: https://w3c.github.io/webvtt/
11. Документація Deepgram [Електронний ресурс] – Режим доступу до ресурсу: https://developers.deepgram.com/docs/
12. Документація Deepl [Електронний ресурс] – Режим доступу до ресурсу: https://www.deepl.com/docs-api
13. Документація Elevenlabs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.elevenlabs.io/
14. Документація Arch Linux [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.archlinux.org/
15. Baeldung [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/

ДОДАТОК 1

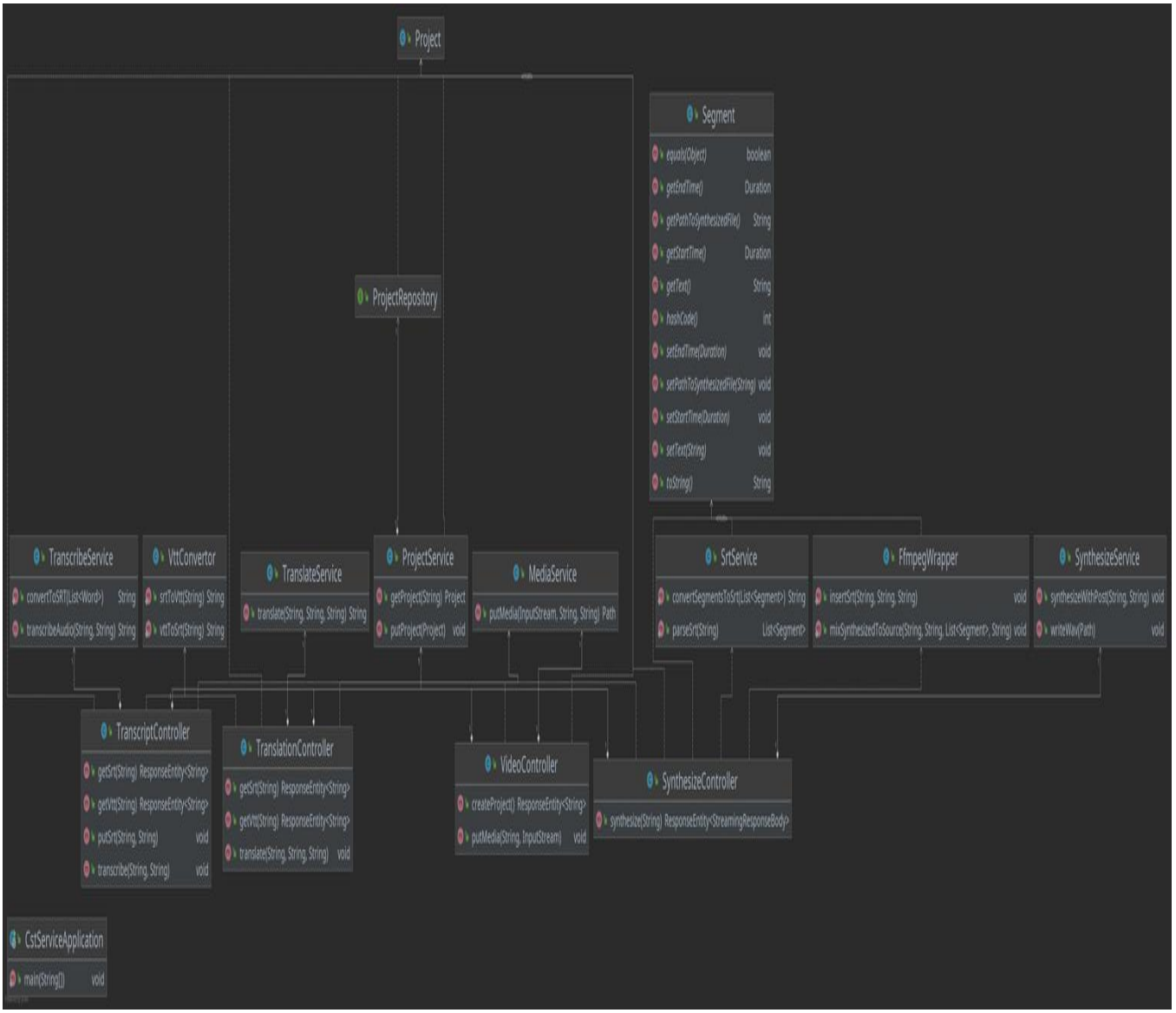
Веб застосунок для транскрибації, перекладу та створення дубляжів

Діаграма класів (функціональна схема)

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2024 р.



					ІАЛЦ.467200.004 Д1		
Змн.	***Арк.***	***№ докум.***	***Підпис***	***Дата***			
Розроб.		*Коваль М.Ю.*			*Веб застосунок для траскрибації, перекладу та створення дубляжів* Компоненти системи (структурна схема)	***Літ.***	***Арк.*** / ***Акрушів***
Перевір.		*Порєв В.М.*					*1* / *1*
						КПІ ім.Ігоря Сікорського ФІОТ ІП-05	
Н. Контр.							
Затверд.							

ДОДАТОК 2

Веб застосунок для транскрибації, перекладу та створення дубляжів

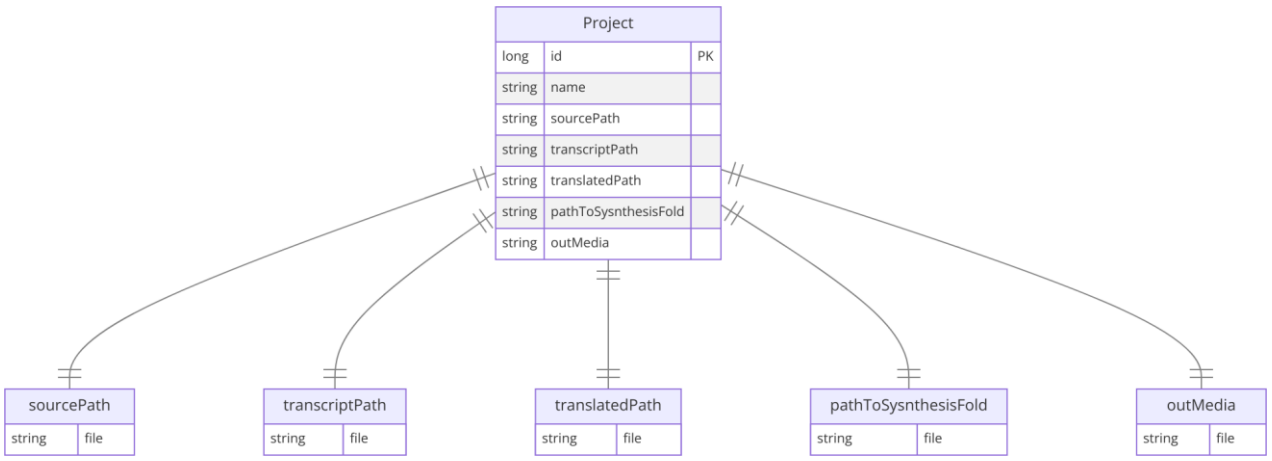
Діаграма зв’язку сутностей

(структурна схема)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2024 р



					ІАЛЦ.467200.005 Д2		
Змн.	**Арк.**	**№ докум.**	**Підпис**	**Дата**			
Розроб.		*Коваль М.Ю*			*Веб застосунок для траскрибації, перекладу та створення дубляжів* Діаграма класів (функціональна схема)	**Літ.**	**Арк.** 1 / **Акрушів** 1
Перевір.		*Порєв В.М.*					
						КПІ ім.Ігоря Сікорського ФІОТ ІП-05	
Н. Контр.							
Затверд.							

ДОДАТОК З

Веб застосунок для транскрибації, перекладу та створення дубляжів

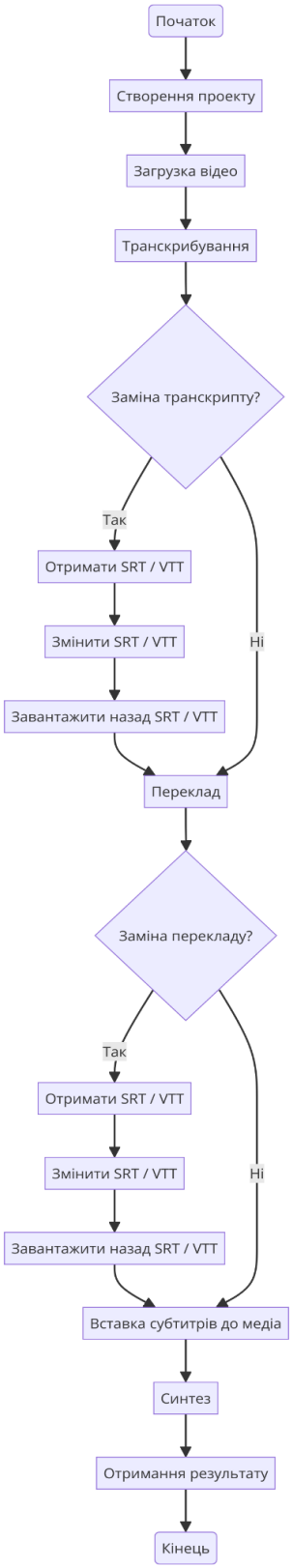
Діаграма прецедентів

(принципова схема)

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2024 р



					ІАЛЦ.467200.006 Д3		
Змн.	***Арк.***	***№ докум.***	***Підпис***	***Дата***			
Розроб.		*Коваль М.Ю.*			*Веб застосунок для траскрибації, перекладу та створення дубляжів* Алгоритм дій програмногозабезпечення (принципова схема)	***Літ.***	***Арк.*** / ***Акрушів***
Перевір.		*Порєв В.М.*					*1* / *1*
						КПІ ім.Ігоря Сікорського ФІОТ ІП-05	
Н. Контр.							
Затверд.							

ДОДАТОК 4

Веб застосунок для транскрибації, перекладу та створення дубляжів

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 38

Київ 2024 р

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CstServiceApplication {

    public static void main (String[] args) {
        SpringApplication.run(CstServiceApplication.class, args);
    }

}

package com.example.cstservice.services;

import com.example.cstservice.entities.Project;
import com.example.cstservice.repositories.ProjectRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class ProjectService {


    @Autowired
    ProjectRepository processRepository;

    public void putProject (Project process) {
        processRepository.save(process);
    }

    public Project getProject (long id) {
        return processRepository.findByIdId(id).get();
    }
}
```

```
package com.example.cstservice.repositories;

import com.example.cstservice.entities.Project;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface ProjectRepository extends JpaRepository<Project, Long> {

}

package com.example.cstservice.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.RequiredArgsConstructor;

import java.time.Duration;

@Data
@AllArgsConstructor
@RequiredArgsConstructor
public class Segment {

    public Duration startTime;
    public Duration endTime;
```

```
    public String text;
    public String pathToSynthesizedFile;
}

package com.example.cstservice.entities;

import jakarta.persistence.*;

@Entity
public class Project {

    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    public long id;

    @Column(name = "name")
    public String name;

    @Column(name = "sourcepath")
    public String sourcePath;

    @Column(name = "transcriptpath")
    public String transcriptPath;

    @Column(name = "translatedPath")
    public String translatedPath;

    @Column(name = "pathToSysnthesisFold")
    public String pathToSysnthesisFold;

    @Column(name = "outmedia")
```

```
    public String outMedia;



    public Project (Long id, String name, String sourcePath) {
        this.id = id;
        this.name = name;
        this.sourcePath = sourcePath;
    }
    public Project () {

    }
}

package com.example.cstservice.controllers;

import com.example.cstservice.entities.Project;
import com.example.cstservice.entities.Segment;
import com.example.cstservice.services.synthesize.SynthesizeService;
import com.example.cstservice.services.media.FfmpegWrapper;
import com.example.cstservice.services.media.MediaService;
import com.example.cstservice.services.ProjectService;
import lombok.SneakyThrows;
import org.apache.commons.io.FileUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```

```
import
org.springframework.web.servlet.mvc.method.annotation.StreamingResponseBody;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.List;

import static com.example.cstservice.services.subtitle.SrtService.parseSrt;

@RestController
@RequestMapping("/synthesys")
public class SynthesizeController {

   @Autowired
   SynthesizeService synthesizeService;

   @Autowired
   ProjectService projectService;

   @SneakyThrows
   @PutMapping(path = "{id}/synthesize")
   public ResponseEntity<StreamingResponseBody> synthesize(
         @PathVariable("id")
               long id) throws Exception {
      Project project = projectService.getProject(id);
      project.pathToSysnthesisFold = MediaService.DEFAULT_DIR + "/" +
project.id;
      project.outMedia = MediaService.DEFAULT_DIR + "/" + id + "out.mp4";
      String srt = FileUtils.readFileToString(new File(project.translatedPath));
      List<Segment> segments = parseSrt(srt);
```

```
        for (int i = 0 ; i < segments.size() ; i++) {
            segments.get(i).pathToSynthesizedFile = project.pathToSysnthesisFold + i +
".wav";
            synthesizeService.synthesizeWithPost(segments.get(i).text,
segments.get(i).pathToSynthesizedFile);
        }
        FfmpegWrapper.insertSrt(project.translatedPath, project.sourcePath,
MediaService.DEFAULT_DIR + "/" + project.id + "temp");
        FfmpegWrapper.mixSynthesizedToSource(MediaService.DEFAULT_DIR + "/"
+ project.id + "temp", MediaService.DEFAULT_DIR, segments, project.outMedia);
        HttpHeaders headers = new HttpHeaders();
        headers.add(HttpHeaders.CONTENT_DISPOSITION, "inline;filename=" +
id);
        headers.add(HttpHeaders.CONTENT_TYPE, "video/mp4");

        File file = new File(project.outMedia);
        StreamingResponseBody responseBody = outputStream -> {
            try (InputStream inputStream = new FileInputStream(file)) {
                byte[] buffer = new byte[4096];
                int bytesRead;
                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }
            }
        };
        return ResponseEntity.ok()
                .headers(headers)
                .contentLength(file.length())
                .body(responseBody);
    }
}
```

```
package com.example.cstservice.controllers;

import com.example.cstservice.entities.Project;
import com.example.cstservice.services.media.MediaService;
import com.example.cstservice.services.ProjectService;
import com.example.cstservice.services.subtitle.VttConvertor;
import com.example.cstservice.services.transcribe.TranscribeService;
import lombok.SneakyThrows;
import org.apache.commons.io.FileUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.io.File;

@RestController
@RequestMapping("/transcript")
public class TranscriptController {

    @Autowired
    ProjectService projectService;

    @Autowired
    TranscribeService transcribeService;

    @SneakyThrows
    @PutMapping(path = "/{id}")
    public void transcribe(
            @PathVariable("id")
                    Long id,
            @RequestParam("lang")
```

```
            String language) {
        Project project = projectService.getProject(id);
        project.transcriptPath = MediaService.DEFAULT_DIR + "/" + id +
"transcript.srt";
        String srt = transcribeService.transcribeAudio(project.sourcePath, language);
        FileUtils.write(new File(project.transcriptPath), srt);
        projectService.putProject(project);
    }


    @SneakyThrows
    @PutMapping("{id}/srt")
    public void putSrt(
            @PathVariable("id")
                Long id,
            String srt) {
        FileUtils.write(new File(projectService.getProject(id).transcriptPath), srt);
    }

    @SneakyThrows
    @GetMapping("{id}/srt")
    public ResponseEntity<String> getSrt(
            @PathVariable("id")
                Long id) {
        String transcriptPath = projectService.getProject(id).transcriptPath;
        return ResponseEntity.ok(FileUtils.readFileToString(new File(transcriptPath)));
    }


    @SneakyThrows
    @GetMapping("{id}/vtt")
```

```
    public ResponseEntity<String> getVtt(
            @PathVariable("id")
                    Long id) {
        String transcriptPath = projectService.getProject(id).transcriptPath;
        String srt = FileUtils.readFileToString(new File(transcriptPath));
        String vtt = VttConvertor.srtToVtt(srt);
        return ResponseEntity.ok(vtt);
    }



}
package com.example.cstservice.controllers;

import com.example.cstservice.entities.Project;
import com.example.cstservice.entities.Segment;
import com.example.cstservice.services.media.MediaService;
import com.example.cstservice.services.ProjectService;
import com.example.cstservice.services.subtitle.VttConvertor;
import com.example.cstservice.services.translate.TranslateService;
import lombok.SneakyThrows;
import org.apache.commons.io.FileUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.io.File;
import java.util.List;

import static
com.example.cstservice.services.subtitle.SrtService.convertSegmentsToSrt;
```

```
import static com.example.cstservice.services.subtitle.SrtService.parseSrt;

@RestController
@RequestMapping("/translate")
public class TranslationController {

   @Autowired
   TranslateService translateService;

   @Autowired
   ProjectService projectService;


   @PutMapping(path = "/{id}")
   public void translate(
         @PathVariable("id")
               Long id,
         @RequestParam("from")
               String fromLanguage,
         @RequestParam("to")
               String toLanguage) throws Exception {
      Project project = projectService.getProject(id);
      project.translatedPath = MediaService.DEFAULT_DIR + "/" + id +
"translated.srt";
      String sourceSrt = FileUtils.readFileToString(new File(project.transcriptPath));
      List<Segment> sourceSegments = parseSrt(sourceSrt);
      for (int i = 0 ; i < sourceSegments.size() ; i++) {
         sourceSegments.get(i).text =
translateService.translate(sourceSegments.get(i).text, fromLanguage, toLanguage);
      }
      String srt = convertSegmentsToSrt(sourceSegments);
```

```
        FileUtils.write(new File(project.translatedPath), srt);
        projectService.putProject(project);
    }


    @SneakyThrows
    @GetMapping("{id}/srt")
    public ResponseEntity<String> getSrt(
            @PathVariable("id")
                    Long id) {
        String transcriptPath = projectService.getProject(id).translatedPath;
        return ResponseEntity.ok(FileUtils.readFileToString(new File(transcriptPath)));
    }


    @SneakyThrows
    @GetMapping("{id}/vtt")
    public ResponseEntity<String> getVtt(
            @PathVariable("id")
                    Long id) {
        String transcriptPath = projectService.getProject(id).translatedPath;
        String srt = FileUtils.readFileToString(new File(transcriptPath));
        String vtt = VttConvertor.srtToVtt(srt);
        return ResponseEntity.ok(vtt);
    }
}



package com.example.cstservice.controllers;
```

```
import com.example.cstservice.entities.Project;
import com.example.cstservice.services.media.MediaService;
import com.example.cstservice.services.ProjectService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.io.InputStream;

@RestController
@RequestMapping("project")
public class VideoController {

    @Autowired
    MediaService mediaService;

    @Autowired
    ProjectService projectService;

    @PutMapping("project")
    public ResponseEntity<Long> createProject() {
        Project project = new Project();
        projectService.putProject(project);
        return ResponseEntity.ok(project.id);
    }

    @PutMapping("{id}/video")
    public void putMedia(
            @PathVariable("id")
                    long id,
                    InputStream stream) {
```

```
        Project project = projectService.getProject(id);
        project.sourcePath = mediaService.putMedia(stream, String.valueOf(id),
MediaService.DEFAULT_DIR).toString();
        projectService.putProject(project);
    }
}

package com.example.cstservice.services.media;

import com.example.cstservice.entities.Segment;
import org.apache.commons.io.FileUtils;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.List;

public class FfmpegWrapper {



    public FfmpegWrapper () {
    }

    public static void insertSrt (String srt, String sourceMedia, String outputMedia) {
        String insertCommand = "${executable} -y -i ${src-media} -preset veryfast -crf
23 -vf subtitles='filename=${src-srt}':fontsdir='${caption-fonts-
dir}':force_style='FontName=${caption-font-name}' -f ${media-format} -y ${output-
media}";
        String command = insertCommand
                .replace("${executable}", "ffmpeg")
```

```
            .replace("${src-media}", sourceMedia)
            .replace("${src-srt}", srt)
            .replace("${caption-fonts-dir}", "/home/maxim/Projects/Maksym/CST-service/src/main/resources")
            .replace("${caption-font-name}", "Merriweather")
            .replace("${media-format}", "mp4")
            .replace("${output-media}", outputMedia);

        try {
            System.out.println(command);
            Process process = Runtime.getRuntime().exec(command);
            BufferedReader reader =
                    new BufferedReader(new InputStreamReader(process.getInputStream()));

            String line;
            while ((line = reader.readLine()) != null) {

            }

            process.waitFor();

            int exitValue = process.exitValue();
            System.out.println("Process exited with value: " + exitValue);

        } catch ( InterruptedException | IOException e) {
            System.err.println(e.toString());
        }
    }
```

```
   public static void mixSynthesizedToSource(String pathToSource, String
pathToSynthesizedFolder, List<Segment> segments, String outPath) throws
IOException, InterruptedException {
      for (Segment tgt : segments) {
         String increaseVolume = "ffmpeg -y -i " + tgt.pathToSynthesizedFile + " -
filter:a volume=2 " + tgt.pathToSynthesizedFile + "incr.wav";
         Process process = Runtime.getRuntime().exec(increaseVolume);
         process.waitFor();
         tgt.pathToSynthesizedFile = tgt.pathToSynthesizedFile + "incr.wav";
      }
      final String decreaseVolumeCommand = "ffmpeg -y -i " + pathToSource + " -
filter:a volume=0.2 " + pathToSource + "_low_volume" + ".mp4";
      try {

         Process process1 = Runtime.getRuntime().exec(decreaseVolumeCommand);
         BufferedReader reader =
               new BufferedReader(new
InputStreamReader(process1.getInputStream()));

         String line;
         while ((line = reader.readLine()) != null) {
            System.out.println(line);
         }
         process1.waitFor();

         int exitValue = process1.exitValue();
         System.out.println("Process exited with value: " + exitValue);
      } catch (IOException | InterruptedException e) {
         e.printStackTrace();
      }
```

```
        String srcMedia = pathToSource + "_low_volume" + ".mp4";
        String outMedia = outPath;
        StringBuilder command = new StringBuilder();
        command.append("ffmpeg -y").append(" ");
        command.append("-i").append(" ");
        command.append(srcMedia).append(" ");

        for (Segment tgt : segments) {
            command.append("-i").append(" ");
            command.append(tgt.pathToSynthesizedFile).append(" ");
        }
        StringBuilder filterComplex = new StringBuilder();
        for (int i = 0; i < segments.size(); i++) {
            long delayMillis = segments.get(i).startTime.toMillis();
            filterComplex.append("[").append(i +
1).append("]adelay=").append(delayMillis).append("|").append(delayMillis).append
("[s").append(i + 1).append("];");
        }
        filterComplex.append("[0]");
        for (int i = 0; i < segments.size(); i++) {
            filterComplex.append("[s").append(i + 1).append("]");
        }
        filterComplex.append("amix=inputs=").append(segments.size() +
1).append("[a]");
        command.append("-filter_complex").append(" ");
        command.append(filterComplex.toString()).append(" ");
        command.append("-map").append(" ");
        command.append("0:v").append(" ");
        command.append("-map").append(" ");
        command.append("[a]").append(" ");
        command.append("-c:v").append(" ");
```

```
        command.append("copy").append(" ");
        command.append("-c:a").append(" ");
        command.append("aac").append(" ");
        command.append(outMedia).append(" ");

        try {
            Process process = Runtime.getRuntime().exec(command.toString());
            BufferedReader reader =
                    new BufferedReader(new
InputStreamReader(process.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            process.waitFor();
            int exitValue = process.exitValue();
            System.out.println("Process exited with value: " + exitValue);
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}

package com.example.cstservice.services.media;

import org.apache.commons.io.IOUtils;
import org.springframework.stereotype.Service;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

@Service
public class MediaService {

    public static String DEFAULT_DIR = "/home/maxim/Downloads";

    public Path putMedia(InputStream stream, String fileName, String dir) {
        Path filePath = Paths.get(dir, fileName);
        try (OutputStream fileStream = Files.newOutputStream(filePath)) {
            IOUtils.copyLarge(stream, fileStream);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return filePath;
    }
}

package com.example.cstservice.services.subtitle;

import com.example.cstservice.entities.Segment;

import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

public class SrtService {

    public static List<Segment> parseSrt(String srtContent) {
```

```
List<Segment> captions = new ArrayList<>();
String[] lines = srtContent.split("\n");

int i = 0;
while (i < lines.length) {
   String indexLine = lines[i].trim();
   i++;

   if (i < lines.length && !indexLine.isEmpty() && indexLine.matches("\\d+")) {
      String timeLine = lines[i].trim();
      i++;

      if (timeLine.contains("-->")) {
         String[] times = timeLine.split("-->");
         Duration startTime = parseDuration(times[0].trim());
         Duration endTime = parseDuration(times[1].trim());
         StringBuilder textBuilder = new StringBuilder();

         while (i < lines.length && !lines[i].trim().isEmpty()) {
            textBuilder.append(lines[i].trim()).append("\n");
            i++;
         }

         String text = textBuilder.toString().trim();
         Segment segment = new Segment();
         segment.startTime = startTime;
         segment.endTime = endTime;
         segment.text = text;
         captions.add(segment);
      }
   }
```

```
            while (i < lines.length && lines[i].trim().isEmpty()) {
                i++;
            }
        }

        return captions;
    }

    private static Duration parseDuration(String timeString) {
        String[] timeParts = timeString.split(":");
        int hours = Integer.parseInt(timeParts[0]);
        int minutes = Integer.parseInt(timeParts[1]);
        String[] secondsAndMillis = timeParts[2].split(",");
        int seconds = Integer.parseInt(secondsAndMillis[0]);
        int millis = Integer.parseInt(secondsAndMillis[1]);

        return Duration.ofHours(hours)
                .plusMinutes(minutes)
                .plusSeconds(seconds)
                .plusMillis(millis);
    }

    public static String convertSegmentsToSrt(List<Segment> segments) {
        StringBuilder srtContent = new StringBuilder();

        for (int i = 0; i < segments.size(); i++) {
            Segment segment = segments.get(i);
            srtContent.append(i + 1).append("\n");
            srtContent.append(formatDuration(segment.startTime)).append(" --> 
").append(formatDuration(segment.endTime)).append("\n");
```

```
            srtContent.append(segment.text).append("\n\n");
        }

        return srtContent.toString().trim();
    }

    private static String formatDuration(Duration duration) {
        long hours = duration.toHours();
        long minutes = duration.toMinutes() % 60;
        long seconds = duration.getSeconds() % 60;
        long millis = duration.toMillis() % 1000;

        return String.format("%02d:%02d:%02d,%03d", hours, minutes, seconds,
millis);
    }
}

import java.io.*;
import java.net.URI;
import java.nio.file.Files;
import java.nio.file.Path;
import java.time.Duration;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import org.json.JSONObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
```

```
import java.net.http.HttpResponse;

@Component
public class SynthesizeService {

   String subscriptionKey;
   static final String OUTPUT_FORMAT = "pcm_44100";

   static final String SUBSCRIPTION_KEY_HEADER = "xi-api-key";
   static final String POST_ENDPOINT_URI = "https://api.elevenlabs.io/v1/text-to-
speech/"
         + "21m00Tcm4TlvDq8ikWAM"
         + "?output_format=pcm_44100"
         + "&optimize_streaming_latency=0";
   final double stability= 0.3;
   final double similarityBoost = 0.5;
   final double style = 0.0;
   final boolean speakerBoost = true;
   final String modelId = "eleven_multilingual_v2";
   String previousTest = null;

   int channelCount = 1;
   int sampleRate = 44100;

   float[] samples;

   @Autowired
   public SynthesizeService (@Value("${ELEVENLABS}") String authToken) {
      this.subscriptionKey = authToken;
   }
```

```
    public void synthesizeWithPost (String text, String path) throws Exception {
        byte[] response = new byte[0];

        URI endpoint = URI.create(POST_ENDPOINT_URI);
        String body = previousTest != null ? getRequestBodyWithContext(text) :
getRequestBody(text);

        Map<String, String> headers = new HashMap<>();
        headers.put("Content-type", "application/json");
        headers.put("Accept", "audio/wav");
        headers.put(SUBSCRIPTION_KEY_HEADER, subscriptionKey);

        HttpClient httpClient = HttpClient.newBuilder()
                .connectTimeout(Duration.ofSeconds(2))
                .build();

        HttpRequest request = HttpRequest.newBuilder()
                .uri(endpoint)
                .header("Content-type", "application/json")
                .header("Accept", "audio/wav")
                .header(SUBSCRIPTION_KEY_HEADER, subscriptionKey)
                .POST(HttpRequest.BodyPublishers.ofString(body))
                .build();

        try {
            HttpResponse<byte[]> httpResponse = httpClient.send(request,
HttpResponse.BodyHandlers.ofByteArray());
            response = httpResponse.body();
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
```

```
    }
    previousTest = text;
    try {
        readStream(new BinaryInputStream(new ByteArrayInputStream(response))
                , response.length / channelCount / 2
                , false
                , 2);
        createWav(Path.of(path));
    }
    catch ( IOException e) {
        throw new AssertionError(e);
    }
    samples = null;

}


private String getRequestBody (String text) {
    return new JSONObject(Map.of
            ("model_id", modelId
                    , "text", text
                    , "voice_settings", Map.of
                            ( "stability", stability
                                    , "similarity_boost", similarityBoost
                                    , "style", style
                                    , "speaker_boost", speakerBoost)))
            .toString();
}

private String getRequestBodyWithContext (String text) {
    return new JSONObject(Map.of
```

```
            ("model_id", modelId
                , "text", text
                ,"previous_text", previousTest
                , "voice_settings", Map.of
                    ( "stability", stability
                        , "similarity_boost", similarityBoost
                        , "style", style
                        , "speaker_boost", speakerBoost)))
            .toString();
    }
package com.example.cstservice.services.transcribe;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.AccessLevel;
import lombok.Data;
import lombok.SneakyThrows;
import lombok.experimental.FieldDefaults;
import lombok.extern.slf4j.Slf4j;
import org.apache.http.client.utils.URIBuilder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Path;
import java.util.Collections;
```

```
import java.util.List;

@Slf4j
@Component
public class TranscribeService {

   @Autowired
   TranscribeService (@Value("${DEEPGRAM}") String authToken) {
      this.API_KEY = authToken;
   }

   private final String API_KEY;
   private static final String URL =
"https://api.deepgram.com/v1/listen?punctuate=true&model=nova-
2&diarize=true&smart_format=true";
   private static final ObjectMapper MAPPER = new ObjectMapper();

   @SneakyThrows
   public String transcribeAudio (String path, String language) {
      AsrResponse asrResponse = sendRequest(path, URL, language);
      return getResponse(asrResponse);
   }

   private String getResponse (AsrResponse asrResponse)  {
      List<Channel> channelList = asrResponse.getResults().getChannels();
      Channel channel = channelList.get(0);
      List<Alternative> alternativeList = channel.getAlternatives();
      Alternative alternative = alternativeList.get(0);
      List<Word> wordList = alternative.getWords();
      convertToSRT(wordList);
      return convertToSRT(wordList);
```

```
    }


    public static String convertToSRT(List<Word> words) {
        StringBuilder srtBuilder = new StringBuilder();
        int index = 1;
        int wordIndex = 0;
        StringBuilder sentenceBuilder = new StringBuilder();
        boolean newChunk = true;
        String chunkStartTime = null;
        while (wordIndex < words.size()) {
            String wordStr = words.get(wordIndex).getPunctuatedWord();
            double startTime = words.get(wordIndex).getStart();
            double endTime = words.get(wordIndex).getEnd();
            if (newChunk) {
                chunkStartTime = formatTime(startTime);

                newChunk = false;
            }
            sentenceBuilder.append(wordStr).append(" ");
            if (wordStr.endsWith(".") || wordStr.endsWith("?") || wordStr.endsWith("!")
|| sentenceBuilder.toString().length() > 150)
            {
                srtBuilder.append(index++).append("\n");
                srtBuilder.append(chunkStartTime).append(" -->
").append(formatTime(endTime)).append("\n");
                srtBuilder.append(sentenceBuilder.toString().trim()).append("\n\n");
                sentenceBuilder = new StringBuilder();
                newChunk = true;
            }
            wordIndex++;
```

```
        }

        return srtBuilder.toString();
    }

    private static String formatTime(double time) {
        int hours = (int) (time / 3600);
        int minutes = (int) ((time % 3600) / 60);
        int seconds = (int) (time % 60);
        int milliseconds = (int) ((time - ((int) time)) * 1000);

        return String.format("%02d:%02d:%02d,%03d", hours, minutes, seconds,
milliseconds);
    }

    protected AsrResponse sendRequest (String path, String url, String language)
throws URISyntaxException, IOException, InterruptedException {
        HttpClient client = HttpClient.newHttpClient();
        URI paramsUrl = new URIBuilder(url)
                .addParameter("language", language)
                .build();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(paramsUrl)
                .header("Authorization", "Token " + API_KEY)
                .header("Content-Type", "audio/x-flac; rate=16000")
                .POST(HttpRequest.BodyPublishers.ofFile(Path.of(path)))
                .build();
        HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
        AsrResponse asrResponse = MAPPER.readValue(response.body(),
AsrResponse.class);
```

```
    return asrResponse;
}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class AsrResponse {
    Metadata metadata;
    Results results;

}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Metadata {

    String transaction_key;
    String request_id;
    String sha256;
    String created;
    Double duration;
    Integer channels;
    List<String> models;
}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Results {
    List<Channel> channels;
```

```
}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Channel {
   List<Search> search = Collections.emptyList();
   List<Alternative> alternatives;
}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Alternative {
   String transcript;
   Integer confidence;
   List<Word> words;
   Paragraphs paragraphs = new Paragraphs();
}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Paragraphs {

   String transcript = "";
   List<Paragraph> paragraphs = Collections.emptyList();
}



@Data
```

```
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Hit {
   Integer confidence;
   Integer start;
   Integer end;
   String snippet;
}

@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Search {
   String query = "";
   List<Hit> hits = Collections.emptyList();
}



@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
@JsonIgnoreProperties(ignoreUnknown = true)
public static class Paragraph {

   List<Sentence> sentences = Collections.emptyList();
   @JsonProperty("num_words")
   Integer numWords = 0;
   Double start = 0.0;
   Double end = 0.0;
}

@Data
```

```
    @FieldDefaults(level = AccessLevel.PRIVATE)
    @JsonIgnoreProperties(ignoreUnknown = true)
    public static class Sentence {

        String text = "";
        Double start = 0.0;
        Double end = 0.0;
    }

    @Data
    @FieldDefaults(level = AccessLevel.PRIVATE)
    @JsonIgnoreProperties(ignoreUnknown = true)
    public static class Word {
        String word;
        Double start;
        Double end;
        Double confidence;
        String speaker;
        @JsonProperty("punctuated_word")
        String punctuatedWord;
        @JsonProperty("speaker_confidence")
        Double speakerConfidence;
    }


}


package com.example.cstservice.services.translate;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.*;
```

```
import lombok.experimental.FieldDefaults;
import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpHeaders;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicHeader;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import java.io.*;
import java.net.URISyntaxException;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.stream.Collectors;


@Component
public class TranslateService {
   private static final Header[] HEADERS = new Header[]{
         new BasicHeader(HttpHeaders.HOST, "api.deepl.com"),
         new BasicHeader(HttpHeaders.CONTENT_TYPE, "application/x-www-
form-urlencoded"),
         new BasicHeader(HttpHeaders.ACCEPT, "*/*")
   };
   private static final String ENDPOINT = "https://api.deepl.com/v2/translate";
```

```
    private final String authKey;
    String context = "";

    public TranslateService (@Value("${DEEPL}") String authToken) {
        authKey = authToken;
    }

    public String translate(String text, String from, String to) throws Exception {
        Request request = new Request(from, to, Collections.singletonList(text),
context);
        context = text;
        try {
            HttpClient client = HttpClients.createDefault();
            HttpPost httpPost = createPostRequest(request);
            HttpResponse response = client.execute(httpPost);
            return getTranslations(response).getTexts().get(0);
        }
        catch (IOException | URISyntaxException e) {
            throw new Exception(e.getMessage());
        }
    }



    private HttpPost createPostRequest (Request request) throws Exception {
        HttpEntity bodyEntity = createStringEntity(request, request.getTexts());
        HttpPost post = createDefaultPost();
        post.setEntity(bodyEntity);
        return post;
    }
```

```
    private StringEntity createStringEntity(Request request, List<String> texts)
throws Exception {
        StringBuilder entityStringBuilder = new StringBuilder();
        entityStringBuilder
                .append("&target_lang=")
                .append(request.targetLang);
        if (request.sourceLang != null) {
            entityStringBuilder
                    .append("&source_lang=")
                    .append(request.sourceLang);
        }
        for (String text : texts) {
            try {
                String encodedText = URLEncoder.encode(text,
String.valueOf(StandardCharsets.UTF_8));
                entityStringBuilder
                        .append("&text=")
                        .append(encodedText);
            } catch (UnsupportedEncodingException e) {
                throw new Exception(e);
            }
        }
        entityStringBuilder.append("&auth_key=").append(authKey);
        return new StringEntity(entityStringBuilder.toString(),
ContentType.APPLICATION_FORM_URLENCODED);
    }

    private HttpPost createDefaultPost() {
        HttpPost post = new HttpPost(ENDPOINT);
        post.setHeaders(HEADERS);
        return post;
```

```
}


private Response getTranslations(HttpResponse response) throws Exception {
    try{
        BufferedReader rd = new BufferedReader(new InputStreamReader(
                response.getEntity().getContent()
        ));
        String responseText = rd.readLine();
        DeeplResponseObject responseObject = new
ObjectMapper().readValue(responseText, DeeplResponseObject.class);
        List<String> translations = responseObject.translations.stream()
                .map(transObj -> transObj.text)
                .collect(Collectors.toList());
        return new Response(translations);

    } catch (IOException e) {
        throw new Exception(e);
    }
}

@Data
@FieldDefaults(makeFinal = true, level = AccessLevel.PRIVATE)
private static class Request {
    String sourceLang;
    String targetLang;
    final List<String> texts;
    String context;
}

@Data
```

```
    @FieldDefaults(makeFinal = true, level = AccessLevel.PRIVATE)
    private static class Response {
        List<String> texts;
    }

    private static class DeeplResponseObject {
        public List<TransObj> translations = new ArrayList<>();

        public static class TransObj {
            public String detected_source_language;
            public String text;
        }
    }
}

package com.example.cstservice.services.subtitle;

public class VttConvertor {

    public static String vttToSrt(String vttContent) {
        StringBuilder srtContent = new StringBuilder();
        String[] lines = vttContent.split("\n");
        int count = 1;
        for (String line : lines) {
            if (line.matches("^\\d{2}:\\d{2}:\\d{2}\\.\\d{3}\\s--
>\\s\\d{2}:\\d{2}:\\d{2}\\.\\d{3}$")) {
                srtContent.append(count++).append("\n");
                srtContent.append(line.replace(".", ",")).append("\n");
            } else if (!line.isEmpty()) {
                srtContent.append(line).append("\n");
            }
```

```
        }
        return srtContent.toString();
    }

    public static String srtToVtt(String srtContent) {
        StringBuilder vttContent = new StringBuilder();
        vttContent.append("WEBVTT").append("\n\n");
        String[] lines = srtContent.split("\n");
        for (String line : lines) {
            if (line.matches("^\\d+$")) {
                continue;
            } else if (line.matches("^\\d{2}:\\d{2}:\\d{2},\\d{3}\\s--
>\\s\\d{2}:\\d{2}:\\d{2},\\d{3}$")) {
                vttContent.append(line.replace(",", ".")).append("\n");
            } else if (!line.isEmpty()) {
                vttContent.append(line).append("\n\n");
            }
        }
        return vttContent.toString();
    }
}
```