МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки
на тему:

ПЛАТФОРМА ДЛЯ СТВОРЕННЯ І СИНХРОННОГО ЧИТАННЯ БАГАТОМОВНИХ ТЕКСТІВ

Виконав студент 2 курсу магістратури
Дарчин Петро Ростиставович _____
(підпис)

Науковий керівник:
доцент
Панченко Тарас Володимирович _____
(підпис)

Засвідчую, що в цій курсовій роботі
немає запозичень з праць інших авторів
без відповідних посилань
Студент Дарчин П.Р _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри Математичної інформатики
«____»_____2021 р., протокол
Завідувач кафедри Терещенко В.М _____
(підпис)

Київ-2021

Реферат

Обсяг роботи 58 сторінок, робота містить 12 ілюстрацій і 19 джерел посиланнь.

СИНХРОННИЙ ТЕКСТ, КНИГА, ВЕБ-ДОДАТОК, МІКРОСЕРВІСИ, МОБІЛЬНІ ДОДАТКИ, СЕРВЕР, IOS, SWIFT, KOTLIN, JAVA, MYSQL, ГРАФ, ОБРОБКА НАТУРАЛЬНОЇ МОВИ, ТОКЕНІЗАЦІЯ, ВКЛАДЕННЯ СЛІВ

Об’єктом роботи є комплекс технологічних рішень для можливості створення та поширення синхронних текстів.

Мета роботи — розробити технологію та всі необхідні супутні інструменти, які б дозволили з'єднювати різномовні переклади одного і того ж тексту, з метою надання користувачу можливості читати текст однією мовою і мати змогу отримати переклад будь-якого його речення одним натисканням на нього. Реалізувати та впровадити мобільні додатки для використання кінцевим споживачем.

Інструменти розроблення: Java, Spring, Spring Boot, Swift, Kotlin, MySQL, Google Cloud Platform, Debian, HTML, CSS, BootStrap, JavaScript, Udpipe, Word2Vec.

Результати кваліфікаційної роботи: було розроблено веб-додаток та мікросервіси, що забезпечую можливість створення синхронних текстів та надають API для мобільних додатків. Було розроблено та впроваджено мібільні додатки на платформах iOS та Android. Було розроблено технологію з’єднання різномовних текстів. Було пройдено всі етапи розробки, від постановки задачі, планування, вибору стеку технологій, до реалізації, тестування, впровадження та практичного використання.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API — Application Programming Interface

DNS — Domain Name System

IP — Internet Protocol

SSL — Secure Sockets Layer

HTTP — Hypertext Transfer Protocol

HTTPS — Hypertext Transfer Protocol Secure

HP — Hewlett-Packard

CPU — Central Processing Unit

GB — Gigabit

RAM — Random-access Memory

SSD — Solid-state Drive

SSH — Secure Shell Protocol

JRE — Java Runtime Environment

UI — User Interface

ВСТУП

Оцінка сучасного стану об’єкта розробки. В контексті покращення навичок володіння іноземними мовами, синхронне читання текстів - метод для практики та поглиблення знань тієї чи іншої мови, при якому, читаючи текст іноземною мовою, користувач має доступ до перекладу кожної частини тексту (речення чи параграфу) на відому йому мову. Такий спосіб поглиблення знань на практиці довів свою ефективність. Існує чимало програмних реалізацій які дають користувачу можливість синхронного читання текстів, наприклад Smart Book, Parallel Texts Reader, Beelinguapp, Libera та інші. В рамках цієї роботи було створено ще один програмний продукт під назвою duoBooks, метою якого є надання користувачу зручного середовища для покращення навичок володіння іноземними мовами за допомогою читання текстів. Хоча синхронні тексти і є центральним інструментом поглиблення знань який пропонує програма, користувачу також надається набір інших необхідних мовних інструментів, наприклад читання будь-яких текстових файлів з можливістю перекладу будь-якого окремого слова, персональний словник з детальним поясненням значення кожного слова, його орфоепія(вимова) та синоніми. Набір таких інструментів з часом буде тільки розширюватись надаючи користувачам все більше і більше можливостей. Що ж до синхронних текстів, окрему увагу слід приділити методу їх генерації. Для цієї задачі було створено веб-додаток за допомогою якого адміністратор може створювати паралельні тексти та керувати ними. Ще однією майбутньою ціллю, до якої прагне продукт, є повна автоматизація даного процесу з метою надання користувачу можливості злиття текстів просто загрузивши відповідні файли з користувацького інтерфесу мобільного додатку. У ході виконання роботи були розроблені мобільні додатки на платформи iOs та Android,

веб-застосунок для адміністрування синхронних текстів, методологія злиття текстів, що в сукупності утворює повноцінну та готову до функціонування систему з можливістю її подальшого розвитку.

Актуальність роботи та підстави для її виконання. У сучасному суспільстві створюються все нові й нові засоби що дозволяють людям ефективно навчатись, зокрема вивчати іноземні мови. Можливості смартфонів надають користувачам чимало нових інструментів. Потрібно також зазначити, що в сучасному суспільстві зростає роль знання іноземних мов. Багато хто любить читати і в той же час хоче покращити свої навички в тій чи іншій іноземній мові. Синхронні тексти надають можливість вирішити два завдання одночасно та поєднати приємне з корисним - вивчати мови, читаючи книги. Практикування іноземної мови в такий спосіб є досить ефективним. По-перше, книга дає “відчуття мови”, того, як мовні конструкції і граматика працюють на практиці.[1] По друге, коли нові слова зустрічаються в контексті книги, їх набагато простіше запам’ятати і вивчити, ніж повторюючи окремо. Крім того, читання художньої літератури нас не обтяжує, ми можемо читати впродовж довгого періоду часу, в такий спосіб ми можемо вивчати мови будь-де, їдучи в метро, перебуваючи у черзі, чи просто у своєму ліжку, готуючись до сну. Кількість часу, яку користувачі зможуть витрачати на вивчення мови істотно збільшиться, при цьому кількість зусиль, витрачених на це, фактично не зміниться.

Мета й завдання роботи. Мета роботи — проектування платформи(системи) для практикування навичок володіння іноземними мовами, що базується на читанні іншомовних текстів. Розробка основних елементів системи: веб-додатку, клієнтських iOs/Android додатків та всіх супутніх елементів нижчих рівнів. Створення та застосування у веб-додатку технології злиття різномовних текстів. Впровадження

системи: домашній сервер, хостинг веб-додатку, публікація додатків на дистрибутивних платформах App Store та Play Market.

Об’єкт, методи й засоби розроблення. Об’єктом роботи є сукупність пов’язаних між собою елементів системи для створення та читання синхронних текстів та супутній набір інструментів для спрощення процесу вивчення та практикування мов.

Веб-проект розроблявся на мові програмування Java, за допомогою фреймворку Spring на основі бази даних MySQL та з використаннях певних NLP-моделей машинного навчання для роботи з текстом. В якості веб-сервера було використано домашній сервер HP ProLiant DL360p Gen8 на основі двох процесорів Intel Xeon E5-2630 v3. Він служить сервером як для бази даних, так і для самого додатку. IOs додаток розроблявся на мові програмування SWIFT. Android додаток розроблявся на мовах програмування Kotlin/Java.

РОЗДІЛ 1
ПРОЕКТУВАННЯ

1.1 Дослідження та формування завдань проекту

1.1.1 Задача проекту. Аналоги

Задача проекту полягає в наданню користувачу зручних інстурументів для практикування навичок володіння іноземними мовами, зокрема таких, що базуються на читанні іншомовних текстів. Взаємодія з користувачем відбувається через мобільний додаток. В контексті додатку можна розрізнити два типи текстів: синхронні та прості. Синхронні тексти забезпечують користувачу доступ до перекладу певної складової частини тексту (речення, декілька речень, або параграф). Під простими текстами мається на увазі будь-які текстові файли з файлової системи телефона відкриті користувачем всередині програми.

На диструбутивній платформі App Store серед додатків зі схожою функціональністю можна виділити такі: Smart Book, Parallel Texts Reader, Beelinguapp. Перевагами програми що розроблялась є: наявність багатофункціонального словника, в перспективі можливість відкривати будь-які файли що знаходяться на телефоні, зручний та зрозумілий дизайн. Також поза рамками цієї магістрської роботи — можливість створення синхронних текстів з програми, додатковий розширений інструменарій для вивчення іноземних мов.

1.1.2 Формування завдань

Серед завдань проекту можна виділити наступні:

- Генерація синхронних текстів. Завдання полягає у створенні всіх необхідних компонентів для зручного та швидкого з'єднання різномовних текстів. Ці компоненти повинні

включати в себе: набір необхідних алгоритмів для злиття текстів, процес що забезпечує життєвий цикл кожної окремої задачі злиття, а також графічний інтерфейс для забезпечення цього життєвого циклу.

- Система для адміністрування контенту. Адміністрування включає в себе управління всіма аспектами щодо зміни будь-яких даних пов'язних з книгами чи будь-яких інших параметрів пов'язаних з будь-якими іншими сутностями що використовуються для вирішення завдання.
- WEB-API для клієнт-серверної комунікації.
- Хостинг всіх необхідних компонентів.
- Створення клієнтських додатків.(Android/iOS).
- Розміщення клієнтських додатків на дистрибутивних системах, їх підтримка.

1.2 Структурні елементи проекту

1.2.1 Веб-додаток. Задачі, архітектурний підхід та компотенти

Задача веб-додатку полягає в розв’язанні наступних завдань: генерація синхронних текстів, система для адміністрування контенту, WEB-API для клієнт-серверної комунікації.

1.2.1.1 Архітектурний підхід

Для вирішення цих завдань, веб-додаток було вирішено розділити на сукупність невеликих, самодостатніх, незалежних, не тісно зв’язаних сервісів, що спілкуються між собою. Така архітектура вибрана для забезпечення гнучкої розробки та можливості простішого масштабування проекту в майбутньому. Мікросервіси представляють архітектурний стиль,

в якому складні додатки створені як сукупність маленьких, легких, самодостатніх, незалежних, нетісно зв’язаних сервісів, кожен з яких відповідальний за конкретний процес. Такий стиль протиставляється монолітному стилю, згідно з яким додатки будуються як єдине ціле.[2]

Мікросервіси співпрацюють один з одним на основі потреби виконання певної дії. Вони «спілкуються» через API, для яких не має значення мова програмування.

1.2.1.2 Мікросервіси

При плануванні, для визначення необхідної кількості мікросервісів було виділено окремі зони відповідальності для кожного з них. В результаті було прийнято рішення розділити всю функціональність між трьома мікросервісами. Крім них, необхідно було створити ще два допоміжних серевера для комунікації та роутингу між ними. Кожен з мікросервісів має власну зону відповідальності:

- користувацький інтерфейс, функціональність та WEB-API пов’язані з генерацією і адмініструванням контенту.
- функціональність та WEB-API для словника (того що всередині мобільних додатків).
- WEB-API для будь-якої іншої взаємодії клієнтських додатків з веб-сервером, окрім словника (отримання контенту, обмін інформацією про користувачів, тд.) .
- допоміжний сервіс на якому “реєструються” всі мікросервіси, та який знає всі порти та IP-адреси інших мікросервісів.
- допоміжний Zuul-сервіс (запезпечує єдину вхідну точку для всіх запитів та динамічну маршрутизацію між мікросервісами)

Цими мікросервісами є: Constructor service, Dictionary service, Mobile service, Registration service, Web service.

1.2.2 Мобільні додатки

1.2.2.1 Вимоги

Для того, щоб розробити продукт з орієнтацією на користувача, контрибютором проекту, Олівко Дмитром, було проведено інтерв'ю аудиторії потенційних користувачів. За допомогою технології дизайн підходу, було виділено основні проблеми використання нинішніх рішень на ринку. Це дає змогу не робити зайву роботу, яка потім не потрібна користувачу. Для того, щоб розробити продукт, який наближений до потреб користувача, потрібно виділити ці потреби. Було проведено 7 глибинних інтерв'ю з можливими майбутніми користувачами. Методика проведення глибинних інтерв'ю полягає у розумінні глибинних мотивацій людини до занять тією чи іншою діяльністю. В нашому випадку, це читання книг на телефоні та іноземною мовою.

Таким чином, було виділено основні вимоги до додатку та проблеми, що виникають при читанні книг іноземною мовою. Опитувані користувачі це студенти, які читають книги в оригіналі. В основному, всі студенти обирають англійську як іноземну. Саме тому більшість книг в проекті мають саме таку мову оригіналу. Підсумовуючи вимоги опитаних користувачів та провівши дослідження того, що вже є на ринку мобільних платформ для читання, сформовано такі основні вимоги для продукту.

Користувач повинен мати змогу:

- Бачити які книги є в наявності для читання
- Читати обрану книги в бажаному мовному варіанті
- Читатаючи книгу, дивитись переклад кожного речення окремо

- Підлаштовувати середовище читання під свої вимоги (розмір тексту, шрифт, відступи, колір тексту та фону)
- Переглянути авторів та обрати книгу за автором
- Розуміти жанр та мати попереднє представлення про книгу перед її прочитанням
- Знати розмір книги (або приблизну кількість часу потрібну на її прочитання)
- Мати змогу вести словник. Додавати, видаляти з нього слова. Мати вичерпну інформацію про кожне слово та його значення
- Мати змогу отримати переклад кожного окремого слова при читанні книги

1.2.2.2 Планування розробки. Вибір технологій

Продукт розроблявся під дві найпопулярніші операційні системи смартфонів та планшетів (iOS та Android). Планування інтерфейсу проводилось сумісно з дизайнером Дмитром Олівко. В процесі було створено макети всіх необхідних екранів додатку. Для цього було використано інструмент для розробки графічних інтерфейсів під назвою Figma. На момент планування та створення дизайнів вже була готова версія Android додатку, розроблена раніше на мові програмування Java. Вона забезпечувала тільки базову можливість читання синхронних текстів. Було прийнято рішення взяти її ядро за основу, провести рефакторинг на мову Kotlin та додати всю іншу необхідну функціональність не застосовуючи на даному етапі нових дизайнів, натомість сконцентруватись на впровадженні нових дизайнів та повноти функціональності в iOS-додатку.

Рішення про міграцію Android додатку на мову Kotlin було прийняте з оглядкою на простоту підтримки в майбутньому, оскільки дана технологія значною мірою розвивається та набирає популярності,

поступово витісняючи свого попередника Java. Перевагами мови Kotlin є: швидше розгортання та компіляція, менший розмір кінцевої програми, будь-яка частина коду, написана на Kotlin, набагато менша порівняно з Java, оскільки вона менш детальна і менше коду означає менше помилок.[3] Kotlin компілює код у байт-код, який може бути виконаний у JVM.

Застосунок для IOs-платформи було прийнято рішення розробляти на мові Swift. Перевагами такого вибору є: open source (за роки свого існування Swift притягнув до себе велику спільноту, що підтримує технологію та створює велику кількість сторонніх інструментів. Безпечний код. Його синтаксис заохочує писати чіткий і послідовний код. Swift забезпечує захист для запобігання помилкам та покращення читабельності. Швидкість. Swift був побудований з урахуванням продуктивності. Як зазначено на сайті Apple, Swift в 2,6 рази швидше, ніж Objective-C, і в 8,4 рази швидше, ніж Python. Попит. Swift безумовна найпопулярніша мова розробки IOs додатків що активно розвивається та підтримується компанією Apple.

РОЗДІЛ 2
СЕРВЕРНА ЧАСТИНА. РЕАЛІЗАЦІЯ

2.1 Вибір технологій

Об’єктом розробки є веб-додаток та мікросервіси задачами яких є генерація синхронних текстів з супутнім інтерфейсом, система для адміністрування контенту та WEB-API для клієнт-серверної комунікації. Проект розроблявся на мові програмування Java, за допомогою його фреймворків Spring та Spring Boot.

При розробці було прийнято рішення використовувати модель програмування MVC — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача. Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

2.1.1 Java Spring Framework

Spring Framework — це програмний каркас з відкритим кодом та контейнерами з підтримкою інверсії управління для платформи Java. Spring Framework не нав'язує якоїсь конкретної моделі програмування, він

дозволяє легко створювати корпоративні програми Java, надає все, що потрібно для того, щоб реалізувати мову Java в корпоративному середовищі, підтримуючи Groovy та Kotlin як альтернативні мови на JVM, також Spring надає гнучкість для створення багатьох видів архітектур залежно від потреб програми.[4]

Spring Boot є фреймворком з відкритим вихідним кодом на основі Java, що використовується для створення мікросервісів. Він розроблений компанією Pivotal Team і використовується для побудови автономних і готових до використання у виробництві додатків.

При розробці інтерфейсної частини було використано такі засоби, як HTML, CSS, Bootstrap, JavaScript, ThemeLeaf. Для спілкування з сервером використовувався AJAX.

2.1.2 MySQL, Google Cloud Storage

Для забезпечення функціонування веб-додатку потрібне надійне сховище дних, яке б дозволяло створити всі необхідні інструменти. За систему керування базою даних було обрано MySQL.

MySQL — система керування реляційними базами даних. MySQL була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних з відкритим кодом була створена як альтернатива комерційним системам. Зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. [5]

2.2 Архітектура

Складовими елементами, що утворюють систему є: nginx веб-сервер, віртуальна машина на сервері, база даних, хмарне сховище, 5 мікросервісів (утворюють веб-додаток та WEB-API), 2 мобільні додатки, DNS-ім’я. Взаємодія цих елементів між собою зображена на рисунку 1. Сервер розділено на декілька віртуальних машин. На одній з них розміщені база даних та мікросервіси, на хмарному сховищі розміщені результати всіх, з’єднань та всі картинки, а посилання на всі файли, що знаходяться в сховищі містяться в базі даних. Зроблено це було для запезпечення збільшення мобільності і зменшення розміру та складності бази даних. Всі клієнти звертаються до сервера, ip-адрес якого прив’язаний до доменного імені duo-books.com. SSL-сертифікат на сервері дозволяє встановлювати з’єднання по https протоколу, так само як і по http протоколу. Nginx, встановлений на сервері служить проксі-сервером для всіх запитів що приходять на сервер. Він слухає як https порт 443, так і http порт 80. Всі запити на доменне ім’я duo-books.com направляються до відповідної віртуальної машини, по її внутрішньому ip-адресу, на порт 8080. Там цей порт слухає один з мікросервісів (web-service), який далі, в залежності від шляху в адресі запиту, направляє запити на відповідні мікросервіси по адресі `localhost:порт_сервісу`. Всі порти всіх мікросервісів відомі даному сервісу завдяки роботі іншого мікросервісу(reg-service), який забезпечує реєстрацію та іформування мікросервісів щодо доступності інших мікросервісів. Сервіси dict, mobile та construct є кінцевими сервісами, що містять в собі всю необхідну бізнес-логіку та при необхідності комунікують з базою даних, та повертають через web-service інформацію назад до користувача.

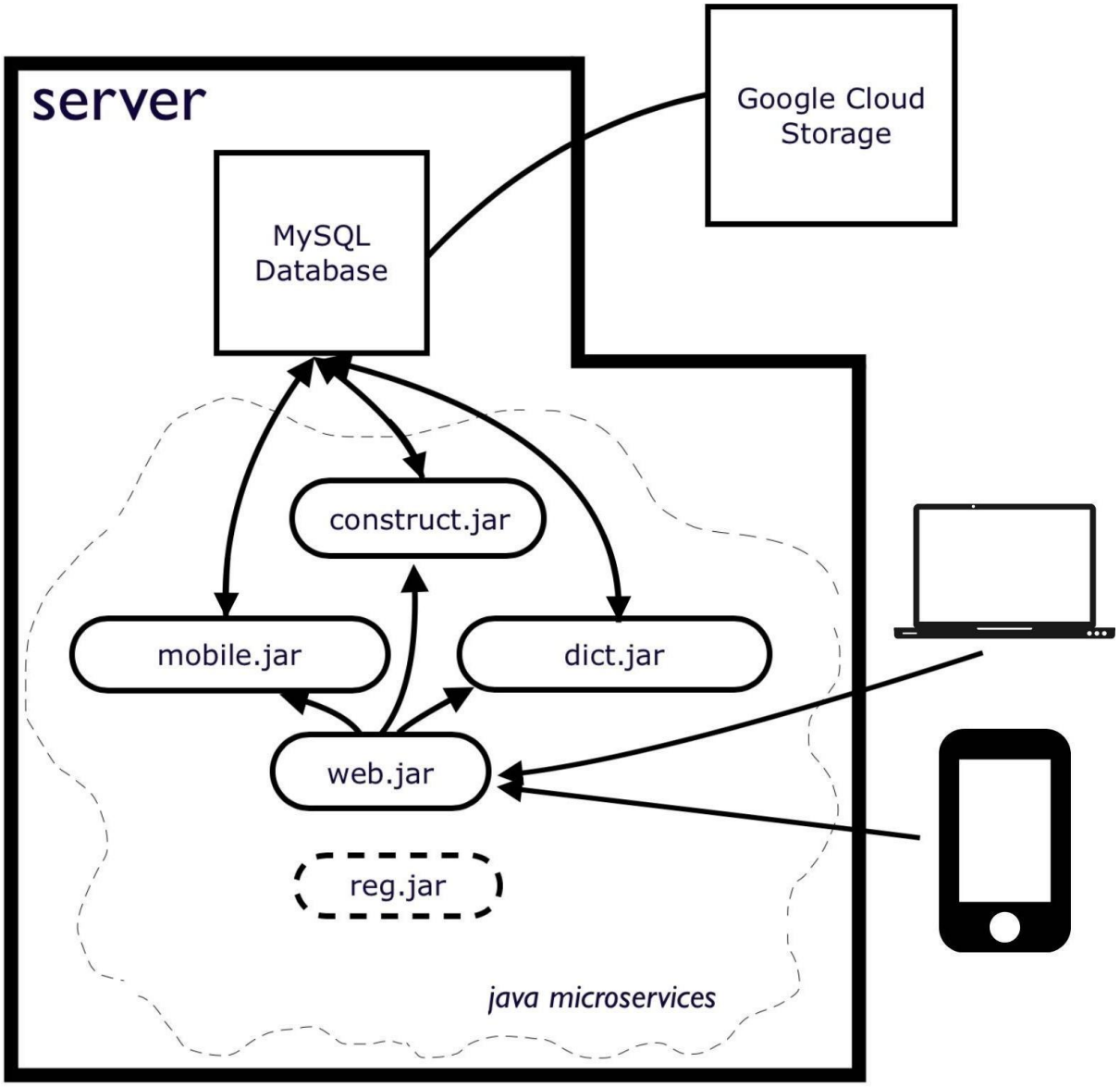

Рисунок 1 – Архітектура проекту

2.3 Мікросервіси

2.3.1 Constructor service (веб-додаток)

Constructor service представляє собою повноцінний веб-додаток, завданням якого є:

- генерація контенту (злиття текстів, їх збереження)

- адміністрування контенту (редагування даних чи параметрів пов'язних з книгами і тд.)
- надання користувацького інтерфейсу для забезпечення перерахованих вище завдань

2.3.1.1 Сутності задачі злиття текстів. Її життєвий цикл

Для забезпечення генерації контенту був розроблений та реалізований механізм що базується на принципі мультикористувацького ієрархічного процесу управління.

У веб-додатку існує три типи користвачів: ADMIN, SUPERADMIN та USER. ADMIN має доступ до перегляду, редагування та видалення будь-якої інформації, окрім інформації про інших користувачів. Він може перевіряти та схвалювати завдання виконані іншими користувачами. SUPERADMIN має всі ті ж права що і ADMIN, тільки також може редагувати ролі іних користувачів. USER в свою чергу може тільки виконувати доступні з загального пулу завдання.

Для представлення кожного окремо взятого процесу злиття двох текстів було введено поняття “завдання”(Task). Будь-яке з’єднання тексів у веб-додатку виконується в рамках одного такого завдання. Тобто кожен Task складається з двох текстів різними мовами. Також він містить локалізовані значення імен авторів та заголовків текстів з яких він скаладається. Task може бути створений тільки адміністратором, після створення він потрапляє в загальний пул завдань і може бути виконаний будь-яким користувачем. Після того, як користувач взяв завдання на виконання, воно зникає з загальнодоступного пула і буде належати користувачу доки він його не виконає. Після виконання, це завдання потрапить в пул завдань для перевірки. Будь-який адміністратор зможе взяти завдання на себе щоб верифікувати правильність кінцевого

результату. Якщо адміністратор одобрить виконане завдання - відбудеться з’єдняння цього завдання з книгою до якої воно належить. Альтернативно адміністратор може сам корегувати результат виконаного завдання, або повернути його користувачу з відповідним повідомленням.

Для представлення вже об’єднаних різномовних текстів було введено поняття “книга”(Book). Одна книга при цьому може уворюватись з довільної кількості різномовних текстів, кожен Task стосується однієї і тільки однієї книги. Одна книга може містити довільну кількість завдань. У кожної книги є головне значення у форматі XML, яке відповідає вимогам формату “.duobk” та містить результат уже виконаних завдань. Кожного разу, коли виконується нове завдання, яке стосується книги, це значення модифікується. Це значення готове до використання і повертається мобільному додатку для відображення тієї чи іншої книги.

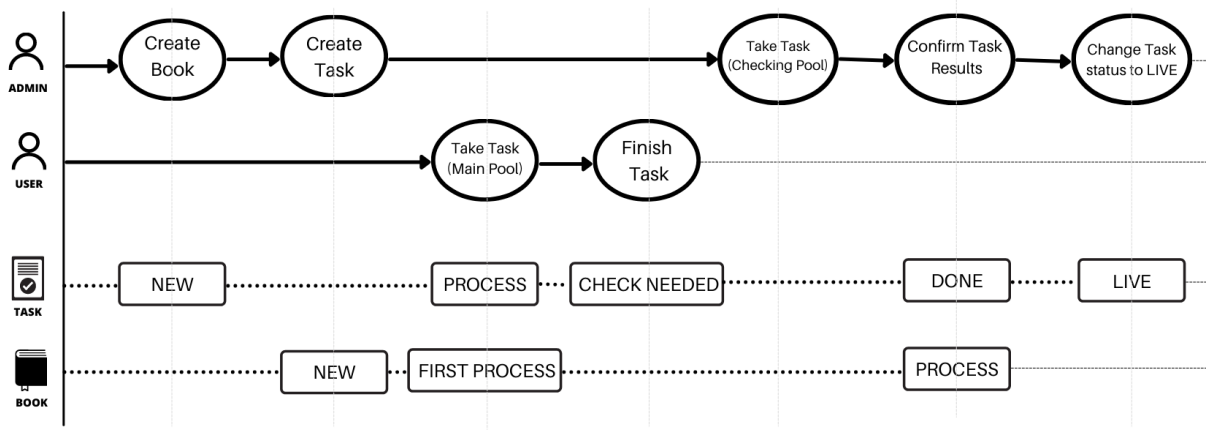

Рисунок 2 – Життєвий цикл завдання

Оскільки потрібно забезпечити можливість об’днання довільної кількості різномовних текстів, є два варінти створення завдання. Якщо книга має статус NEW, тобто після її створення не було створене жодне завдання для неї, то перше завдання утворюється на основі двох вхідних файлів, статус книги змінюється на FIRST_PROCESS, жодне інше завдання не може бути створено до того моменту, доки не буде виконане

дане, після виконання воно буде служити основою для всіх інших завдань що стосуються даної книги, а статус книги змінюється на PROCESS. Якщо у книги такий статус, то всі подальші завдання утворюються на основі тексту першого файлу першого завдання та нового файлу (текст мовою, якої ще немає в книзі). Діаграма зв'язку статусів завдань та книг і дій користувачів зображена на рисунку 2. Статус LIVE завдання означає що його результат буде доступний користувачам з мобільного додатку.

Розглянемо, як все ж таки виконується процес об'днання текстів всередині кожного завдання. І так, на вхід подаються два файли одного з двох форматів: epub або fb2. Текст всередині файлів вже розбитий на параграфи, це забезчує структура форматів. Далі, для початку роботи алгоритму, описаного в розділі 4, потрібно виділити першу та останню пару параграфів які відповідають одна одній. Після того як це було зроблено, починає роботу алгоритм (будується граф, зважуються дуги, знаходиться найдовший шлях), суть алгоритму детально описана в розділі 4. В результаті ми отримуємо утворені відповідності між параграфами, або множинами параграфів, див. рис.3. Далі отриманий результат перевіряється на правильність. Надається можливість виправити той чи інший зв'язок, якщо він був встановлений неправильно.

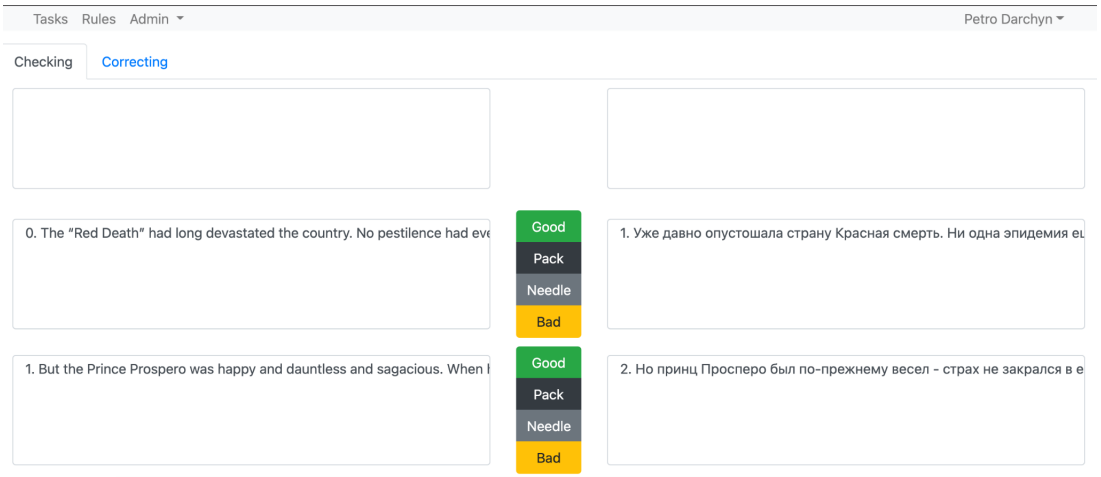

Рисунок 3 – Коректування отриманих результатів

2.3.2 Dictionary service

Dictionary service — це мікросервіс, завданням якого є:

- надання WEB-API для функціональності особистого словника в додатку
- збереження, редагування, видалення даних, реалізація логіки пов’язаної з функціонуванням словника

Детальніше про словник та його функціональність описано в пункті 3.1.4. Для прикладу наведемо декілька WEB-API ендпоінтів що забезпечують цю функціональність.

```
https://duo-books.com/dictionary/getOrCreateAndGetByAccessId
https://duo-books.com/dictionary/section/create
https://duo-books.com/dictionary/section/modifyBatch
https://duo-books.com/dictionary/section/delete
https://duo-books.com/dictionary/item/create
https://duo-books.com/dictionary/item/edit
...
```

2.3.3 Mobile service

Mobile service це мікросервіс, завданням якого є:

- надання WEB-API для функціональності всіх частин мобільного додатку, окрім словника (логін, отримання даних про доступні книги, отримання книг, тд.)

Детальніше про роботу всього мобільного додатку можна ознайомитись в розділі 3. Для прикладу наведемо декілька WEB-API ендпоінтів які надає даний мікросервіс.

```
https://duo-books.com/mobile/authors/getMenuItems
https://duo-books.com/mobile/books/getMenuItems
```

```
https://duo-books.com/mobile/books/rate
https://duo-books.com/mobile/constants/getAll
https://duo-books.com/mobile/users/loginMobileFb
https://duo-books.com/mobile/users/loginMobileApple
https://duo-books.com/mobile/users/setPrefLangs
…
```

2.3.4 Допоміжні сервіси. Eureka server та Zuul service

Сервер Eureka — це програма, що зберігає інформацію про всі доступні мікросервіси. Кожен серсвіс реєструється на сервері Eureka, і сервер Eureka знає про всі клієнтські сервіси, а також порт та IP-адрес на якому вони працюють.[6] Сервер Eureka також відомий як Discovery Server. Сервер Eureka постачається з пакетом Spring Cloud. Він реалізовується аннотацією `@EnableEurekaServer` над головним програмним класом сервісу.

Eureka сервором в нашому випадку є сервіс під назвою reg-service. Він запущений на порті 1111 та отримує інформацію, коли запущений будь-який сервіс аннотований як `@EnableDiscoveryClient` з відповідним програмним параметром `eureka.client.serviceUrl.defaultZone`. В нашому випадку для всіх мікросервісів, значенням цього параметру є `http://localhost:1111/eureka/`.

Zuul Server - це програма, що є вхідною точкою(gateway), що обробляє всі запити та виконує динамічну маршрутизацію між необхідними мікросервісами. Сервер Zuul також відомий як Edge Server. Маршрутизація відбувається на основі шляху в адресі запиту.

Наприклад, в нашому випадку, один із запитів, що має адрес:

```
https://duo-books.com/dictionary/item/create
```

буде направлений на dictionary-service. Інформація про те, який шлях відповідає якому мікросервісу міститься в файлі програмних налаштувань (application.properties). В нашому випадку Zuul сервером є сервіс під назвою web-service, і налаштування маршрутизації в ному наступні:

```
zuul:
     routes:
          constructor-service: /constructor/**
          dictionary-service: /dictionary/**
          mobile-service: /mobile/**
```

Всі мікросервіси зі своєї сторони повинні надати своє ім’я у файлі конфігурацій. У випадку, наприклад, dictionary-service, це

```
spring.application.name = dictionary-service
```

Zuul сервер реалізовується анотацією @EnableZuulProxy, яку необхідно додати до основного класу програми Spring Boot. Анотація @EnableZuulProxy використовується, щоб програма Spring Boot діяла як проксі-сервер Zuul.

2.4 База даних. Хмарне сховище

База даних повинна відповідати наступним функіцональним вимогам:

- забезпечення функціонування життєвого циклу створення синхронних текстів
- забезпечення роботи персонального словника
- зберігання посилань на файли що розташовані на хмарному сховищі
- збереження інформації про користувачів та їх дані

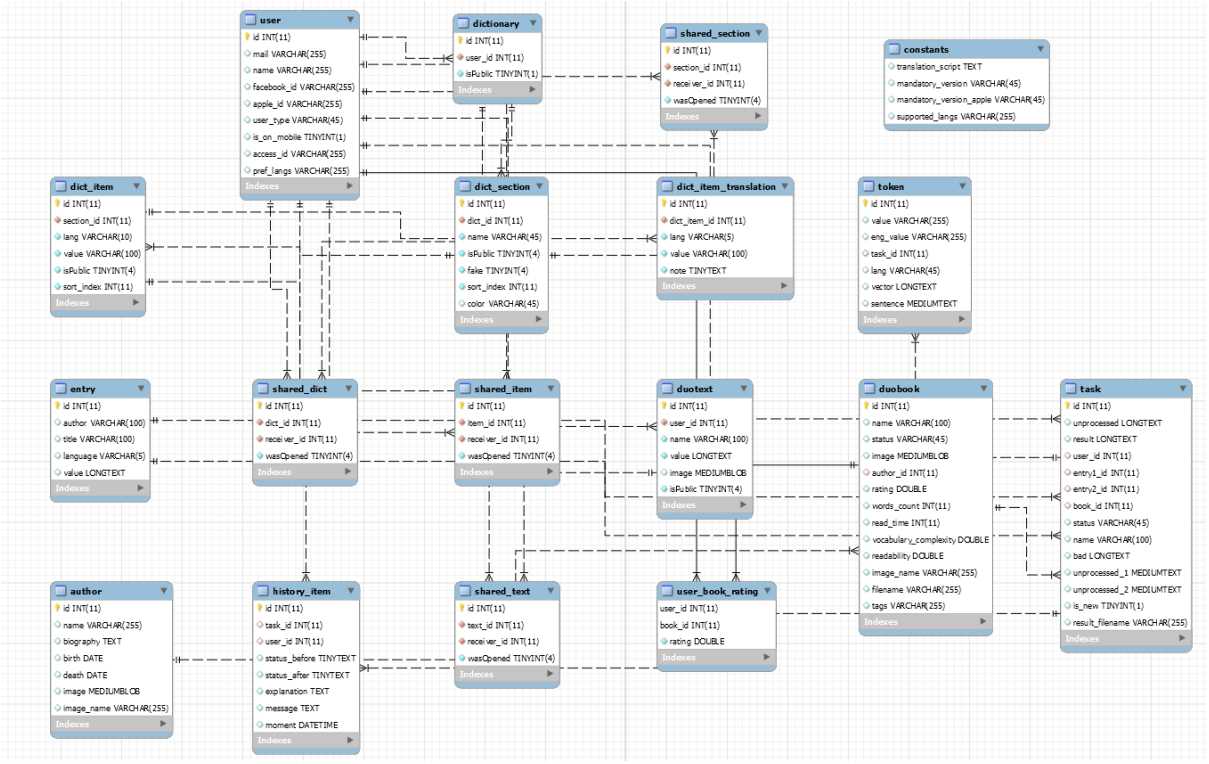

Рисунок 4 – Діаграма бази даних

Діаграма всіх таблиць бази даних зображена на рисунку 4. Далі ми окремо розглянемо декякі з них та їх призначення.

В таблиці “user” зберігається інформація про ім’я користувача, його електронну пошту, Apple Id, чи Facebook Id в залежності від способу логіну в додаток, мови яким користувач надає перевагу при використанні додатку та accessId - токен для забезпечення безпеки та ідентифікації користувача при запитах до сереверу

Таблиці duobook та author містять всю інформацію про книгу та автора для відображення в мобільному додатку. А також посилання на текст синхронної книги та зображення її обгортки, що зберігаються на хмарному сховищі. Таблиці duobook, task, entry, history_item, token забезпечують життєвий цикл створення завдань та книг (див. 2.3.1.1). Таблиці dictionary, dict_section, dict_item, dict_item_translation забезпечують функціонування персонального словника. user_book_rating — таблиця що

містить оцінки користувачів книгам, constants — допоміжна таблиця, що містить номери актуальних версій мобільних додатків, список мов, що підтримуються проектом, посилання на translation_script(лінк що використовується при перекладі окремих слів в додатку). Таблиці shared_dic, shared_section, shared_item створенні з урахуванням можливої майбутньої функціональності спільного доступу до елементів словника(для реалізації можливість ділитись елементами словника з іншимм користувачами всередині додатку)

Також база містить три збережені процедури (stored procedure). create_dict_if_not_exist_return_id — по параметру accessId визначає користувача та перевіряє чи наявний його словник в базі даних. Якщо такого не виявлено, то створює його і повертає його ідентифікатор. Інша збережена процедура update_book_rating — приймає параметр bookId та перераховує середню оцінку книги на основі всіх оцінок користувачів, update_read_time — приймає параметр WPM(Words Per Minute), на основі якого для кожної книги перераховує оцінку часу який піде на її прочитання.

Для збереження текстових файлів та зображень було прийнято рішення використати хмарне сховище Google Cloud Storage. Зберігання цих файлів в базі даних при великих нагрузках могло б привести до втрат в продуктивності та швидкості роботи серверу. А також це б значно збільшило розмір бази даних, що б ускладнило процес створення, зберігання та транспортування резервних копій. Cloud Storage - це послуга для зберігання ваших об’єктів у Google Cloud. Об'єкт - це незмінна частина даних, що складається з файлу будь-якого формату. Об'єкти зберігаються в контейнерах, які звуться бакетами(bucket). Усі бакети пов'язані з проектом, а проекти можна групувати в рамках організації.[7] Оскільки файли, що містять текст з’єднаних книг можуть модифікуватись в процесі їх

синхронізації з новими мовами, виникла необхідність створення двох бакетів, один з яких буде використовуватись в цілях розробки та тестування, а інший для використання у продакшні. Тому у нашому випадку було створено два бакети (duobooks-main та duobooks-dev).

2.5 Сервер. Хостинг

Мікросервіси та база даних проекту хостяться на одній з віртуальних машин виділених на домашньому сервері HP Proliant DL360p G8, розташованому в м. Тисмениця Івано-Франківської області. Домашня мережа має статичну зовнішню IP-адресу, інтернет підключення та електропостачання за довгий час користування проявили себе достатньо добре та стабільно, тому таке рішення є цілком виправданим, особливо враховуючи довгострокову різницю в ціні якщо порівювати з досвідом використанням зовнішніх сервісів оренди серверів чи віртуальних машин. Раніше проект був розташований на віртуальній машині Google Cloud Platform з мінімальними потрібними для проекту потужностями (2 CPUs, 4.75 GB RAM, 10 GB SSD), що обходились в більш ніж 40$ на місяць. Для розширення проекту та втілення запланованої на майбутнє функціональності ці характеристики треба як мінімум подвоїти. Домашній сервер хоч і не забезпечує бажаний рівень стабільності та надійності, але на етапі розвитку та відсутності будь-якої монетизації проекту є необхідним кроком для повної реалізації бачення розвитку продукту, оскільки навіть потужності які він може надати віртуальній машині в своїй базовій теперішній комплектації (16 CPUs, 64 GB RAM, 120 GB SSD, 1.2 TB HDD) вже є великою мірою надлишковими для проекту. Слід зазначити, що хостинг елементів проекту не був єдиною метою придбання серверу, мета та використання інших віртуальних машин з проектом ніяк не пов’язані. Але в разі необхідності ресурси виділені сервером для

проекту можуть бути змінені та підлаштовані для запезпечення повноцінного функціонування.

2.5.1 HP ProLiant DL360p G8 та його комплектація

Сервер HP ProLiant DL360p G8 заснований на двохпроцесорній платформі, яка відмінно підходить для застосувань загального призначення, віртуалізації і хмарних рішень. В порівнянні з попереднім поколінням таких серверів, він пропонує кращу продуктивність і більше внутрішньої пам'яті, також має більш швидкі контролери масивів і швидшу пам'ять з більш низькою напругою. Механізм управління HP з вбудованим інтелектуальним забезпеченням забезпечує спрощений і економічний запуск сервера. Система Always-On Active Health підвищує стабільність, і скорочує час простою завдяки постійному моніторингу. [8]

Система базується двох процесорах Intel Xeon E5-2630 v3 @ 2.40GHz, що забезпечують дуже хороший рівень продуктивності. Конфігурація з двома процесорами включає сім вбудованих вентиляторів. Внутрішній простір корпусу має продуману схему, що забезпечує легкий доступ до компонентів для заміни або оновлення (див. рис. 5). Шасі обладнано температурними датчиками, які автоматично регулюють швидкість вентиляторів, що забезпечує достатній потік повітря для охолодження компонентів системи.

На даному сервері встановлено 8 RAM-модулів по 8 GB кожен. Усього доступно 24 роз'ємів для установки RAM-модулів, кожен процесор підтримує чотири канали пам'яті з трьома слотами DIMM для пам'яті DDR4, всього 12 слотів пам'яті на один процесор. Серія HP ProLiant G8 представила SmartMemory, що збільшує продуктивність пам'яті на 16,6% в порівнянні з попередніми поколіннями. Установка різних типів модулів пам'яті не підтримується.

Сервер використовує 2 SAS жорстих диска по 600 гігабайт. Але ця пам’ять не використовується віртуальною машиною виділеною під проект. Натомість було встановлено SSD диск, 80 гігабайт якого було виділено для проекту. Загалом дана конфігурація сервера HP ProLiant DL360p Gen8 підтримує до 8 жорстких дисків SAS, SATA або SSD, з розміром 2,5-дюйма SFF і можливістю гарячої заміни.[9] HP представила носії SmartDrive на своїх серверах Gen8, додавши світлодіодні індикатори стану, щоб надати користувачеві безліч візуальних сигналів про стан жорсткого диска.

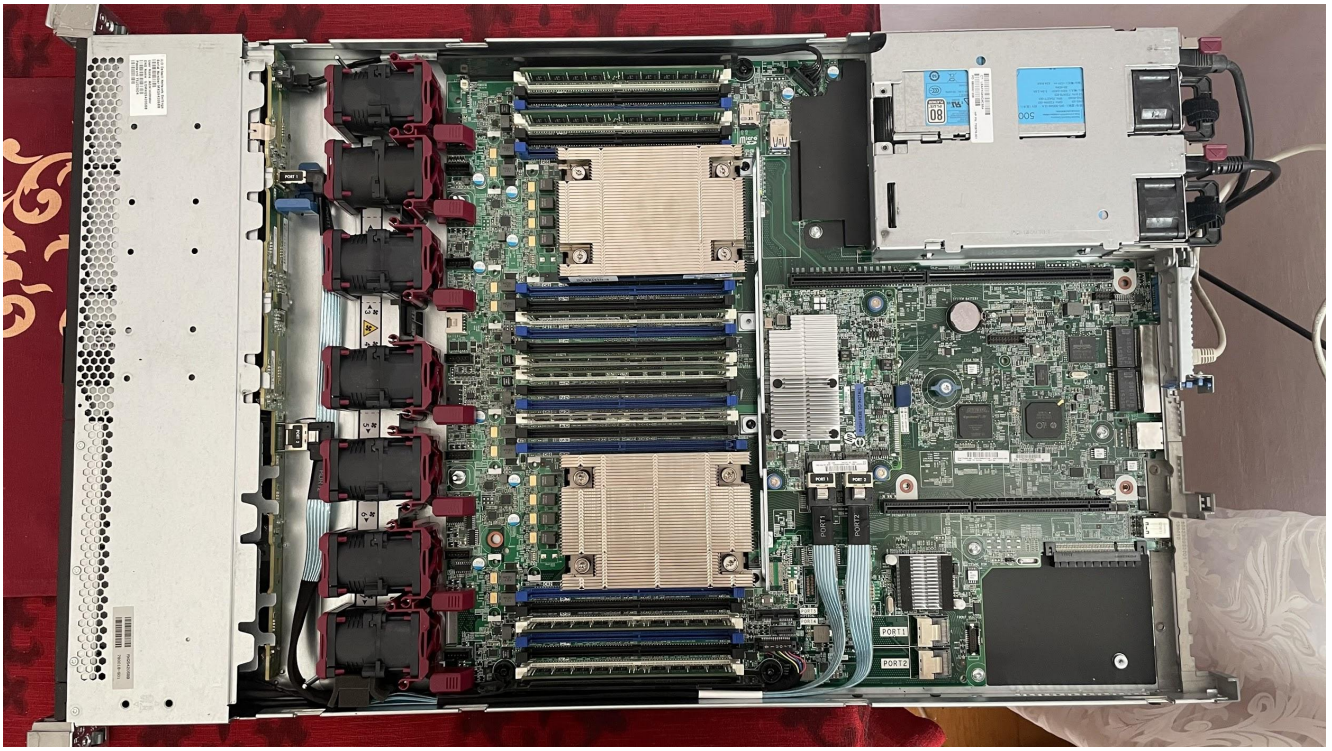

Рисунок 5 – Домашній сервер HP Proliant DL360p G8

2.5.2 Віртуалізація

Хостинг елементів проекту не був єдиною метою придбання серверу. За допомогою програми VMware ESXi потужності сервера було розділено між декількома віртуальними машинами.

VMware ESXi (раніше ESX) - це гіпервізор корпоративного класу типу 1, розроблений VMware для розгортання та обслуговування

віртуальних комп'ютерів. Як гіпервізор типу 1, ESXi не є програмним додатком, який встановлюється в операційній системі, натомість він сам включає та інтегрує життєво важливі компоненти ОС, надаючи ядро для віртуалізації комп’ютера.[10]

Під проект була виділена віртуальна машина з 24 GB оперативної пам’яті, 8 ядрами CPU, 80 GB SSD пам’яті (рис 6). Мета та використання інших віртуальних машин з проектом не пов’язані.

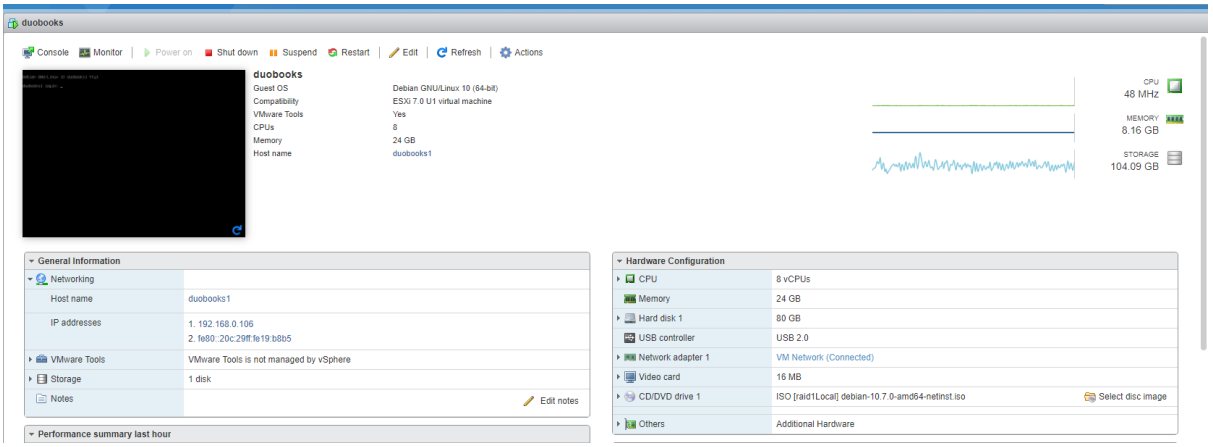

Рисунок 6 – Налаштування віртуальної машини “duobooks”

2.5.3 Налаштування серверної ОС

На віртуальній машині встановлено операційну систему Debian GNU/Linux 10. Debian, також відомий як Debian GNU/Linux, — це дистрибутив Linux, що складається із вільного програмного забезпечення з відкритим кодом, розробленого спільним проектом Debian, який був створений Яном Мердоком 16 серпня 1993 року [11]. Перша версія Debian (0,01) була випущена 15 вересня 1993 р. , а перша стабільна версія (1,1) випущена 17 червня 1996 р. Гілка Debian Stable є найпопулярнішим варіантом для використання на серверах. Debian також є основою для багатьох інших дистрибутивів, зокрема Ubuntu.

Всі подальші налаштування проводились безпосередньо в Linux терміналі, зокрема налаштування користувачів, брандмауера, SSH-підключення, встановлення необхідного програмного забезпечення, налаштування бази даних та доступу до неї, тд.

Для забезпечення функціонувння бази даних було встановлено систему керування базами даних MariaDB. MariaDB — реляційна система керування базами даних, створена на початку 2009 як відгалуження MySQL, що активно використовується на Linux-системах.

Для запуску самого веб-додатку було достатньо встановити Java Runtime Environment. Java Runtime Environment (JRE) — мінімальна реалізація віртуальної машини, що необхідна для виконання Java-додатків, без компілятора й інших засобів розробки. Складається з віртуальної машини — Java Virtual Machine — та бібліотеки Java-класів.

Для налаштування запуску сервісів, та їх авноматичного старту при перезапуску системи було використано systemd. Systemd — це набір програм, що надає обширні можливості у вигляді системних компонентів для операційних систем Linux. Її основною метою є уніфікація конфігурації та поведінки служб у дистрибутивах Linux[12].

Для встановлення віддаленого з’єднання до домашнього серверу використовується SSH-протокол. Secure Shell, SSH — мережевий протокол рівня застосунків, що дозволяє проводити віддалене управління комп'ютером і тунелювання TCP-з'єднань (наприклад, для передачі файлів). Схожий за функціональністю з протоколом Telnet і rlogin, проте шифрує весь трафік, в тому числі і паролі, що передаються[13]. З’єднання встановлюються на основі наявності публічного ключа на сервері та приватного на комп’ютері з якого встановлюється підключення. Приватний ключ захищений паролем. SSH-протокол також використовується для

передачі файлів, як з комп’ютера на сервер, так і навпаки. Для цього використовується Secure Copy Protocol (SCP) для якого існує відповідна команда “scp”. Також SSH-протокол дає можливість встановлення віддаленого підключення напряму до бази даних.

2.5.4 Доменне ім’ я

Доменне ім'я — унікальна прив'язка IP-адреси, на якій знаходиться сайт, до певної буквенної послідовності в DNS-системі. Це — унікальне алфавітно-цифрове позначення, яке є необхідним елементом адреси в мережі Інтернет. Доменне ім'я дозволяє ідентифікувати веб-сайт або адресу електронної пошти в мережі Інтернет.

Для спрощення процесу отримання доменного імені в DNS-системі, існує велика кількість сервісів що надають послуги по отриманні доменного імені. В даному випадку було обрано сервіс GoDaddy, який є одним з найбільш популярних в даній галузі. GoDaddy підтримує близько 30 мільйонів доменних імен у найбільших доменних зонах першого рівня, включаючи .com, .org, .net, .biz, .info. Як для веб-додатку, так і для WEB-API проекту було обране ім’я duo-books.com.

РОЗДІЛ 3
МОБІЛЬНІ ДОДАТКИ

Можливості та інтерфейс iOS та Android додатків значним чином різняться (див. пункт 1.2.1.2). Було прийнято рішення зосередити часові ресурси на iOS додатку та розвивати в першу чергу його. Розглянемо набір можливостей які було реалізовано в iOS додатку в рамках виконання цієї роботи, а також набір можливостей які заплановано реалізувати в майбутньому.

3.1 Вимоги до функціональності. Плануванння

3.1.1 Логін

В додатку користувач повинен мати змогу увійти в систему. Це потрібно для закріплення за користувачем особистого словника, реалізації можливості оцінювання книг та для забезпечення можливості будь-яких міжкористувацьких операцій в майбутньому. Логін в додатку можна виконати за допомгою фейсбук аккаунта, гугл аккаунта, або використовуючи Apple ID.

Логін не є обов’язковим. Користувачі що не увійшли в систему можуть використовувати всі можливості додатку, що не потребуються ідентифікації користувача.

При першому запуску додатку користувачем, запитується перелік мов які він знає або вивчає. Цей перелік зберігається в локальній пам’яті телефона, а якщо користувач увійде в систему, то ця інформація збережеться і в базі даних. Набір мов, що особа використовує в додатку необхідні для кращої та простішої подачі інформації. Наприклад, сортування книг в меню в залежності від доступних мовних варіантів цієї

книги. Книги, що доступні в одній з мов що вивчаються будуть розміщені вище в списку всіх книг. Переклад окремих слів буде здійснений на мову що вивчає користувач. Це налаштування користувач зможе змінити в разі необхідності.

3.1.2 Головний екран. Навігація

На головному екрані додатку користувач бачить список синхронних текстів доступних для читання. Кожен з них представлений у вигляді “картки” на якій міститься обгортка книги, а також наступна інформація: назва книги, автор, рейтинг на основі оцінок інших користувачів та кількість цих оцінок, приблизна тривалість читання книги, мови що доступні для синхронного читання. Також користувач має можливість згрупувати всі доступні книги по авторах і бачити всі доступні твори кожного автора окремо. При натисканні на “картку” книги користувачу надається можливість змінити мову читання і мову перекладу якщо вони були підібрані програмою неправильно. Далі книга скачується на телефон, ділиться на сторінки відповідно до розмірів екрану, шрифту, відступів, після чого відображається для читання.

На головному екрані міститься нижня панель навігації, що дозволяє користувачу перемикатись між функціональністю пов’язаною з читанням та персональним словником. Також, в правому верхньому куті міститься кнопка меню, що показує та приховує панель навігації додатку. Там міститься кнопка “Login” обо “Logout”, кнопка персонального словника, та кнопка інфромації про додаток. Якщо користувач увійшов у систему то в правому нижньому куті відображається його ім’я.

3.1.3 Режим читання

Однією з переваг даного продукту над його аналогами є ретельно пропрацьовний режим відображення тексту на основі пагінації (поділу

тексту на сторінки). Такий режим відображення використовується всіма найбільш популярними книжковими рідерами на мобільних платформах. Текст займає весь простір на екрані, вирішуючи проблему необхідності постійного гортання тексту вниз. Єдиним недоліком такого підходу є складність реалізації, але він робить користування додатком максимально зручним та приємним. А також, в майбутньому це дозволть розширити можливості додатку до повноцінного рідера будь-яких текстів.

В режимі читання взаємодія з користувачеми відбувається на основі різних типів натискання та жестів. Щоб побачити переклад речення синхронного тексту необхідно один раз натиснути на нього на екрані. Щоб отримати переклад будь-якого окремого слова в тексті потрібно зробити подвійне натискання на нього. Щоб відкрити меню читання — довге натискання на будь-яке місце на екрані. Сторінки книги перегортаються слайдом вперед або назад. При першому заході в режим читання користувач завжди бачить підказку щодо різних типів натискання, надалі, за бажанням, він може її вимкнути. При перекладі окремого слова подвійним натисканням, доступна можливість додавання його до персонального словника.

В меню читання містяться кнопка налаштування режиму читання, кнопка переходу в денний або нічний режим, кнопка зміни розміру шрифта, кнопка персонального словника, кнопка навігації назад, назва відкритої книги та її автор, а також слайдер для навігації між сторінками. На екрані налаштування режиму читання користувач може змінити наступні параметри: бокові відступи, вертикальні відступи, кольори тексту та кольори фону в денному та нічному режимах. Налаштування зберігаються в пам’яті телефона і не втрачаються між запусками додатку.

3.1.4 Книги в додатку. Плани щодо розширення кількості контенту

На даний момент для читання доступні тільки згенеровані адміністраторами синхронні книги. Планується поступово розширювати об’єм контенту доступного в додатку(не тільки синхронних текстів) шляхом надання можливості читання будь-яких файлів з пам’яті телефона, можливості завантажувати тексти та ділитись ними всередині додатку, можливості генерування синхронних текстів по вимозі користувача, а також, шляхом суттєвого збільшення кількості згенерованих синхронних текстів за допомогою максимальної автоматизації процесу та покращення точності з’єднання.

3.1.5 Словник

Як було втановлено внаслідок опитування потенціальних користувачів додатку, реалізація можливісті зберігати слова та фрази прямо диктується користувацькими потребами. Для цієї мети було реалізовано персональний словник, що автоматично “підтягує” всю необхідну інформації про кожне слово. Словник складається з секцій та слів. Призначення секцій — групувати слова в разі необхідності, з метою впорядкування словника в разі його розширення. Кожне слово може належати, або не нележати до секції. Зі сторони користувача, секція характеризується ім’ям та кольором.

Переклад слів здійснюється на базі open-source проекту googleDictionaryAPI, що розміщений на github. Гугл не надає API для свого словника, вище згаданий проект це скрапер що парсить сторінки гугл-словника та повертає в структурованій формі. Такий підхід не є надійним і не може бути довгостроковим розв’язком проблеми, оскільки доступність джерела даних не є гарантованою в майбутньому, а також підтримка проекту може стояти під питанням. Таке рішення є тимчасовим,

його перевагами є: підтримка всіх необхідних для додатку мов, простота, ціна, відкритість.

GoogleDictionaryAPI забезпечує повноту інформації про кожне слова та його значення. Правопис слова, фонетика слова (його транскрипція та вимова), всі можливі значення слова, частина мови до якої належить слово, його текстове визначення (тлумачення), приклади використання та синоніми.

Словник доступний тільки користувачам що залогінились в додаток. В словнику реалізована можливість змінювати порядок слів та секцій, переміщувати слова між секціями, видаляти та додавати слова та секції. В майбутньому планується реалізувати можливість “поділитися” словом, секцією, чи всім словником з іним користувачем, чи групою користувачів.

3.2 Інтерфейс. Дизайн-макети

Для реалізації запланованої функціональності, сумісно з колишнім студентом факультету кібернетики, а тепер дизайнером інтерфейсів Дмитром Олівко, було розроблено дизайн-макети всіх екранів додатку. Для цього використовувався інтрумент Figma.

Figma — це програма для розробки інтерфейсів, яка працює в браузері, і пропонує максимально широкий набір можливостей для спільних дизайнерських проектів на основі команд. Figma надає всі інструменти, необхідні для фази проектування проекту, включаючи векторні інструменти, якими можно створювати повноцінні ілюстрації, а також можливості прототипування та генерації коду для розробників. Figma є безкоштовним продуктом, і, незважаючи на те, що Figma базується на браузері, існують версії для робочого столу як для Windows, так і для Mac OS.[14]

Розробка нових дизайнів відбувалась в три етапи: проектування, створення макетів, виділення компонент. На етапі проектування на основі вже існуючих “старих” дизайнів Android-додатку було визначено основні екрани що повинні бути присутні в додатку, та зв’язок між ними. Це представлено на рисунку 7.

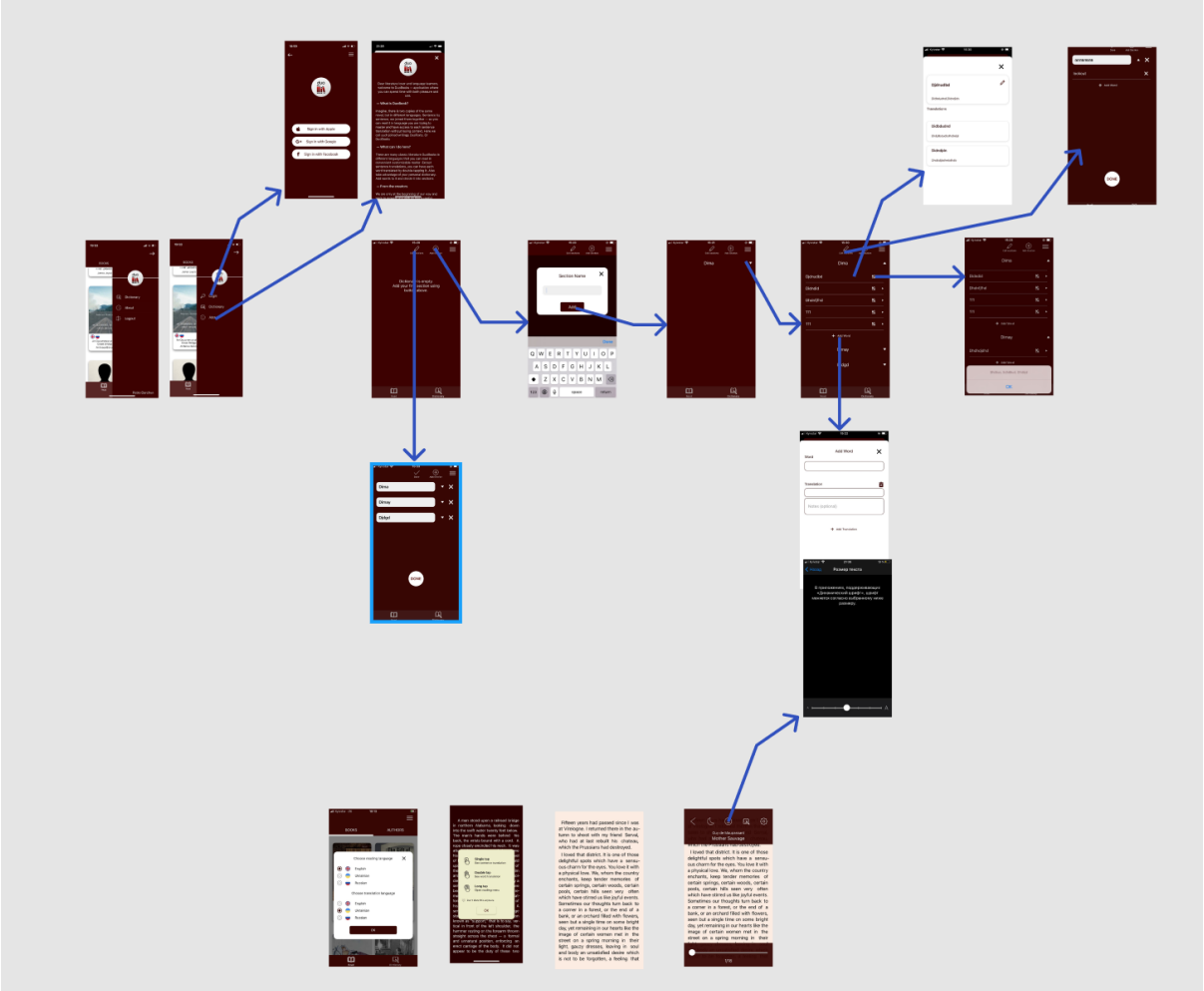

Рисунок 7 – Проектування дизайнів для додатку у Figma

На етапі створення макетів кожен окремий екран детально пропрацьовувався, даючи в результаті готовий для розробки шаблон, який використовуватиметься програмістом для втілення дизайнів в реальність.

На шаблоні розробник може бачити всі розміри тих чи інших елементів, відступи між ними, кольори, градієнти, тд.

Компоненти - це елементи багаторазового використання при дизайні. Верхній колонтитул, нижній колонтитул або навіть кнопка можуть бути компонентом. Коли компонент повторно використовується кілька разів, будь-яка структурна зміна його в майбутньому вплине на всі інші випадки його використання в шаблонах. Компоненти в Figma надзвичайно гнучкі. Виділення компонент на етапі дизайну спрощує програмісту роботу по створенню UI-елементів. Програмісту матиме сенс дотримуватись концепції компонент створених дизайнером, та створювати окремі класи для кожної окремої повторюваної компоненти щоб забезпечити можливість простої та швидкої зміни вигляду інтерфейсів при необхідності. Вигляд компонентів для додатку в середовищі Figma частково можна побачити на рисунку 8.

Рисунок 8 – Дизайн компонентів для додатку у Figma

3.3 Реалізація

3.3.1 iOS додаток

Як було зазначено в розділі 1, додаток розробрався на мові Swift. Середовищем розробки виступав XCode — це інтегроване середовище розробки Apple(IDE) для macOS, що використовується для розробки програмного забезпечення для macOS, iOS, iPadOS, watchOS та tvOS, що вперше з’явився у 2003 році та є найпопулярнішим інструментом, що дозволяє розробляти нативні додатки для пристроїв Apple.

Рекомендованим архітектурним підходом для розробки iOS-програм в XCode, який використовувся і в даному випадку, є MVC. Модель дизайну Model-View-Controller досить стара. Це модель високого рівня, оскільки вона стосується глобальної архітектури програми та класифікує об’єкти відповідно до загальних ролей, які вони відіграють у програмі. Класи рівня Model інкапсулюють дані та надають варіанти їх представлення базуючись на основних різновидах їх можливої поведінки. Об’єкти рівня View представляють інформацію користувачу. Об'єкти контролера прив'язують модель до об’єктів інтерфейсу (рівня View)[15].

Каркас для реалізації рівнів View та Controller міститься в фреймворку UIKit. Він надає необхідну інфраструктуру для реалізації взаємодії між даними рівнями, наприклад методи для реагування на події пов’язані з взаємодією користувача з інтерфейсом, необхідна інформація про інтерфейс, стан системи та програми, методи доступу до головного потоку. Прикладами класів що надає фреймворк є: UIApplication, UIWindow, UIViewController.

При розробці інтерфейсів використовувся підхід “Programmatic UI”, при якому всі елементи інтерфейсу створені програмістом шляхом написання коду. Альтернативно, для простішої реалізації та візуального

представлення інтерфейсу в процесі розробки Apple пропонує інструмент під назвою Storyboards. Таким чином код для інтерфейсу створювався б “під капотом”. Недоліком цього підходу є менша гнучкість та додатковий шар реалізації. Тому було прийнято рішення дотримуватись принципу “Programmatic UI”

Для взаємодії з сервером використовувалась бібліотека Alamofire — це мережева бібліотека HTTP на базі Swift для iOS та macOS. Вона забезпечує елегантний інтерфейс поверх стандартного інструменту(Foundation), що пропонує Apple. Перевагами її використання є спрощення написання коду для нетворкінгу в додатку, краща читабельність та менша кількість коду.

3.3.2 Android додаток

Для розробки самого додатку було використано Android Studio. Дане середовище дає змогу повноцінної розробки мобільних додатків для пристроїв з операційною системою Android. Використані мови розробки — Kotlin/Java.

Для клієнт-серверної взаємодії було використано бібліотеки Retrofit та Java RX. База для додатку створювалась на мові Java в 2019 році. В рамках даної роботи, Android проект було перероблено на мову Kotlin і надалі підтримувалась Kotlin-версія додатку. Класи, напасані на Java і Kotlin можуть одночасно використовуватись в проекті та навіть звертатись один до одного. Єдиним класом в проекті, написаним на Java залишився клас що відповідає за пагінацію тексту, оскільки він є досить громіздким та важким для модифікації.

3.4 Впровадження

App Store — це платформа цифрового розповсюдження, розроблена та підтримувана Apple Inc., для мобільних додатків на операційних системах iOS та iPadOS.

Для поширення iOS додатку в App Store потрібно створити аккаунт розробника та бути учасником “iOS Developer program”. Для членства в цій прогармі потрібно проводити щорічні платежі в розмірі 100$. Зі своєї сторони Apple надає розробнику низку корисних та зручних інструментів для тестування та впровадження готових продуктів. Але при цьому досить жостко контролює дотримання власних правил щодо того, чи відповідає програма критеріям безпеки, приватності, корректності інформації та навіть продуктивності і дизайну. Всі ці правила описані в документі під назвою “App Store Review Guidelines” та перевіряються при кожному новому завантаженні реліз-версії додатку.

Для тестування додатку Apple пропонує сервіс під назвою TestFlight — це онлайн-сервіс для інсталяції та тестування мобільних додатків, що пропонується розробникам лише в рамках “iOS Developer program”. Дозволяє зручно та швидко поширювати тестові версії програми між учасниками команди, надає інструменти для фідбеку та репортує про збої в програмі.

Google Play - крамниця застосунків від Google, що дозволяє власникам пристроїв з мобільної операційної системи Android та інші завантажувати та купувати різні застосунки, книги, фільми та музику.

Вигляд сторінок duoBooks на просторах App Store та Google Play можна побачити на рисунках 9 та 10.

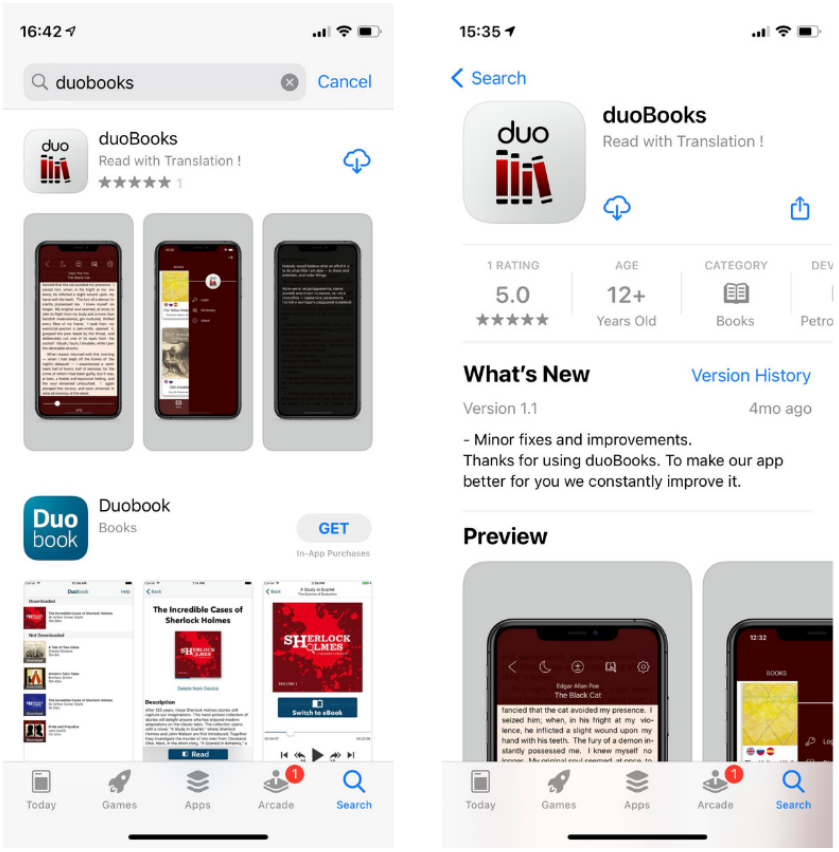

Рисунок 9 – Сторінка проекту в App Store

Рисунок 10 – Сторінка проекту в Google Play

РОЗДІЛ 4
ТЕХНОЛОГІЯ ЗЛИТТЯ ТЕКСТІВ

4.1 Проблематика. Постановка задачі

На вхід подаються два файли – варіанти одного і того ж тексту різними мовами. На виході потрібно отримати один файл, всередині якого встановлено відповідності між реченнями двох текстів. Всередині файлів, що подаються на вхід, тексти вже розподілені на параграфи. Це забезпечують такі формати електронних книг, як “.epub” і “.fb2”. Отже на вході маємо два файли вищезгаданих форматів. На виході потрібно отримати один файл формату “.duobk” (власний формат, що базується на базі XML, містить згруповані різномовні речення та може відображатись мобільними додатками duoBooks)

Проблеми, що потрібно вирішити при встановленні відповідностей виникають коли структура параграфів та речень в двох текстах відрізняються одна від одної. Наприклад одному параграфу першого тексту відповідають два або й більше параграфів другого. Така ж ситуація можлива і з реченнями обох текстів.

```
<result>
  <chapter>
    <dp>
      <ds>
        <s1 index="0" pIndex="337">The King’s son was going to be married, so there were general rejoicings. </s1>
        <s lang="ru">Царский сын собрался жениться, и вся страна ликовала. </s>
        <s lang="uk">Син Короля збирався одружитися, тож усі навкруги безмежно раділи. </s>
        <s lang="de">Des Königssohnes Hochzeit stand bevor, und darob war allgemeine Freude. </s>
      </ds>
      <ds>
        <s1 index="1" pIndex="337">He had waited a whole year for his bride, and at last she had arrived. </s1>
        <s lang="ru">Он целый год ждал невесты, и она наконец приехала. </s>
        <s lang="uk">Він цілий рік чекав на свою наречену, й нарешті вона прибула. </s>
        <s lang="de">Er hatte ein ganzes Jahr auf seine Braut gewartet, und endlich war sie gekommen. </s>
```

Рисунок 11 – Приклад результату з’єднання чотирьох текстів

В більшості випадків відмінності між текстами що необхідно “згладити” досить незначні щоб це було можливо зробити достатньо точно. В подальших пунктах буде детально описано процес вирішення задачі. Приклад кінцевого результату з’єднання показано на рисунку 11. Це знімок екрана фрагменту готової книги, доступної адміністраторам у веб-додатку.

4.2 Представлення задачі у вигляді графа

Подамо поставлену задачу у вигляді графа-дерева. Зробимо це на прикладі з задачі встановлення відповідності між параграфами. Присвоїмо параграфам кожного з текстів індекси, які будуть відповідати порядковому номеру параграфа в тексті.

Нехай вершинами графа будуть пари множин таких індексів, при чому найменший елемент кожної з множин відповідає індексу найбільшого елемента відповідної множини батька, збільшеного на 1.

Коренем дерава при цьому є вершина, індекси множин якої відповідають індексам множин першої правильної відповідності між параграфами в тексті. А листками — вершини, індекси множин якої відповідають індексам множин останньої можливої відповідності між параграфами в тексті. Графічний приклад можна побачити на рисунку 12.

При цьому кожній дузі надамо вагу, яка буде відповідати значенню від 0 до 100, яке буде відповідати наближеній оцінці, наскільки відповідність, що задається вершиною, в яку направлена дуга, є правильною. Це значення буде еврестичною оцінкою того, наскільки правдиве твердження що параграфи, які відповідають цій вершині, є відповідними. Спосіб обчислення цього значення буде описано в пункті 4.3.

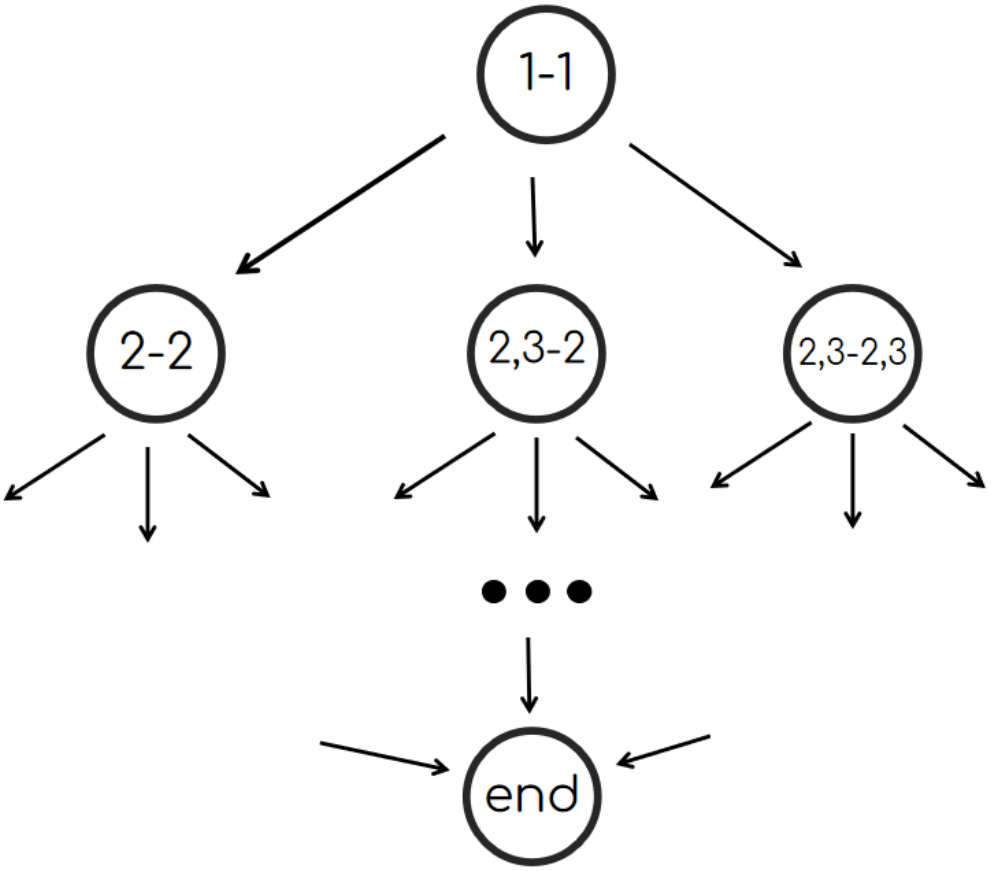

Рисунок 12 – Подання у вигляді графа

Тоді можна розлядати задачу злиття текстів як задачу знаходження шляху з найбільшою сумою ваг. Нагадаємо, що кожна вершина графа містить індекси параграфів обох текстів і фактично відповідає за одну “відповідність” між параграфами. Знайшовши найдовший шлях ми по індексах можемо повернутись назад до текстового формату.

Звісно, результат не буде повністю точним та буде залежати від таких факторів, як відмінність в структурі параграфів обох текстів, порядку та кількості речень всередині параграфів і точності перекладу взагалі.

Таку ж модель подання можна застосувати і для проблеми встановлення відповідності між реченнями всередині параграфів.

4.3 Визначення ваг ребер графа

Як було описано вище, вага дуги направленої до вершини графа, є евристичною оцінкою того, наскільки правдиве твердження, що параграфи, які відповідають цій вершині, є аналогами одне одного в різномовних текстах. Зважування графа проводиться з метою подальшого знаходження найдовшого шляху. Пари множин параграфів, що відповідають вершинам з якого складатиметься цей шлях, будуть вважатися відповідними частинами різномовних текстів.

4.3.1 Визначення схожості текстів

4.3.1.1 Існуючі способи визначення схожості текстів

Метрики що можуть використовуватись для порівняння відповідності двох текстів є необхідними для надання оцінки якості перекладу зробленого системами машинного перекладу. З цією метою, доступні різні автоматизовані показники для порівняння машинного перекладу з якісним перекладом зробленим людиною. Оскільки кожна людина перекладач продукує переклад з різним підбором і порядком слів, найкращі метрики оцінюють якість перекладу на основі людських перекладів з різних джерел. Метриками, що найчастіше використовуються є: BLEU(Bilingual Evaluation Understudy), NIST (the U.S. National Institute of Standards & Technology metric), METEOR (the metric for Evaluation of Translation with Explicit Ordering), TER (Translation Error Rate). BLEU був одним з перших показників, який продемонстрував високу кореляцію при використанні з еталонними людськими перекладами. Загальний підхід для BLEU полягає у спробі знайти схожі фрази різної довжини у двох текстах. Потім для обчислення метрики використовують певні зважені середні значення збігів[16].

Оцінка базується на двох основних показниках: семантична схожість, та корегуючий коефіцієнт. Семантична схожість показує наскільки смислове навантаження двох текстів схоже між собою. Корегуючий коефіцієнт враховує різницю в довжині складових елементів текстів.

Для визначення семантичної схожості в обох текстах виділяються ключові слова та відкидаються другорядні(процес токенізації). Після чого всі ключові слова другого тексту перекладаються на мову першого тексту. Далі всі ключові слова перетворюються у вектори, що відображають їхнє значення. Потім покомпонентно знаходиться середній вектор значення для кожного з текстів. Два вектори отримані в результаті порівнюються і отрмуємо оцінку семантичної схожості текстів.

Розроблений в рамках даної роботи підхід використовує інструменти, що дозволяють робити оцінку схожості текстів на основі оцінки їх семантичнох схожості.

4.3.1.2 Опис процесу визначення оцінки схожості текстів

На вході маємо два тексти. В контексі даної задачі цими текстами будуть два параграфи однієї і тієї ж книги різними мовами. Першими кроком є виділення та лематизація ключових слів обох параграфів. Другим кроком є переклад ключових слів другого тексту на мову першого тексту. В контектсті нашої програми мовою першого тексту завжди є англійська, тому в нашому випадку, другий крок - це переклад ключових слів другого тексту на англійську. Третім кроком є визначення вкладання слів (word embedding), тобто перетворення слів у вектори. Четвертим кроком є визначення середнього вектору для кожного з текстів(параграфів) та порівняння цих векторів. Внаслідок цього порівняння й отримуємо оцінку схожості текстів. Далі розглянемо кожен крок детальніше.

4.3.2 Токенізація тексту

Лексичний розбір — це процес перетворення послідовності символів в послідовність токенів (груп символів що відповідають певним шаблонам), та визначення їх типів. Програма, чи функція що виконує лексичний аналіз, називається лексичним аналізатором, токенізатором. Токенізація — це процес розмежування та, можливо, класифікації секцій рядка вхідних символів. Потім отримані токени передаються іншій формі обробки.

Для процесу токенізації використовувався інструмент під назвою UDPipe. UDPipe містить треновані моделі, що використовуються для токенізації, тегування та лемматизації, та доступні для 65 різних мов. UDPipe доступний як бінарник для Linux/Wondows/OS X, як бібліотека для C++, Python, Perl, Java, C#, а також як веб-сервіс.

В проекті використовується Java бібліотека для роботи з UDPipe. На вхід подається текст та шлях до моделі. На виході отримуємо текст, де кожен рядок містить токен та всю інформацію про нього. Цей текст легко парситься, при парсингу витягуємо всі параметри токена які нас цікавлять. З необхідного можна виділити такі характеристики як частина мови до якої належить слово та лемма(початкова форма слова, в якій воно присутнє в словнику). В процесі токенізації відсіюються всі сполучнки, прийменники та розділові знаки. Далі лемми слів другого тексту перекладаються на мову першого тексту. Для подальшої обробки перший текст також лематизується.

4.3.3 Векторизація тексту

Для визначення семантичної схожості ключових слів використовуються треновані моделі NLP-технології word2vec. Word2vec - це техніка для роботи з природною мовою. Алгоритм word2vec

використовує модель нейронної мережі для вивчення асоціацій між словами із великого корпусу тексту. Після навчання така модель може виявляти слова-синоніми або пропонувати додаткові слова для часткового речення. Як випливає з назви, word2vec представляє кожне окреме слово з певним списком чисел, який називається вектором. Вектори вибираються ретельно таким чином, щоб проста математична функція косинус подібності, вказувала на рівень семантичної подібності між словами, представленими цими векторами. Word2vec - це група пов’язаних моделей, які використовуються для створення вкладання слів (word embeddings). Ці моделі являють собою дрібні двошарові нейронні мережі, які навчені реконструювати мовний контекст слів. Word2vec бере для введення великий корпус тексту і створює векторний простір, як правило, декількох сотень вимірів, при цьому кожному унікальному слову в корпусі присвоюється відповідний вектор у цьому просторі. Вектори слів розташовані у векторному просторі так, що слова, що мають спільний контекст у корпусі, розташовані близько один до одного у просторі[17]. Word2vec було створено та опубліковано 2013 року командою дослідників від проводом Томаша Міколова з Google.

У проекті використовують готова тренована модель для векторизації англійськийх слів. Для її застосування до неангломовних текстів, ключові слова попередньо перекладені на англійську(яка завжди є мовою першого тексту процесу злиття). Для подальшого порівняння та застосування функції косинусу подібності, знаходиться середній вектор з векторів ключових слів кожного з порівнюваних текстів. В результаті процесу векторизації отримуємо два вектори(наприклад для двох параграфів тексту), які далі можемо порівнювати для визначення семантичної схожості.

4.3.4 Обчислення оцінки співпадіння

Оцінка схожості двох векторів, що представляють два тексти вираховується на основі формули косинусу подібності. Косинус подібності (англ. cosine similarity) — коефіцієнт подібності двох не нульових векторів у предгільбертовому просторі, який обчислюється як косинус кута між ними. Косинус 0° дорівнює 1, а для всіх інших значень кута в інтервалі $(0,\pi]$ буде менше за 1 [18]. Отож, це оцінка напрямку, а не величини: два вектори з однаковим напрямком мають косинус подібності 1, а два вектора, які утворюють кут 90° один відносно одного, мають подібність 0, а два діаметрально направлені вектори мають подібність -1, незалежно від їх довжини. Косинус подібності часто використовують в позитивному просторі, для якого результат обмежений проміжком [0, 1]. Назва походить від терміну «направлений косинус»: в цьому випадку одиничні вектори максимально «подібні», якщо вони паралельні і максимально «різні», якщо вони ортогональні (перпендикулярні). Це аналогічно косинусу, який є одиницею (максимальне значення), коли відрізки утворюють нульовий кут і нулем (не корельовані), коли відрізки ортогональні.

Ці межі застосовуються до будь-якої кількості вимірів, але найчастіше косинус подібності використовується у багатовимірних додатних просторах. Наприклад, при інформаційному пошуку та аналізі тексту, кожен термін пов'язаний з окремим виміром, і тому документ характеризується вектором, де значення кожного виміру відповідає кількості разів, що термін з'являється у документі. Тоді косинус подібності дає корисну оцінку того, наскільки подібні два документи у термінах теми.

У нашому випадку косину подібності вираховується функцією cosineSim бібліотеки nd4j, яка є частиною проекту deeplearning4j, що надає інструменти глибинного навчання для JVM.

```
return  cosineSim(Nd4j.create(vector1), Nd4j.create(vector2));
```

В результаті отримуємо оцінку від 0 до 1 наскільки два тексти є семантично схожими.

Корегуючий коефіцієнт був введений для покращення результатів співставлень частин текстів які очевидно не відповідають одна одній по довжині слів, але можуть мати схожі семантичні вектори. Наприклад якщо в одному параграфі 200 слів, а в іншому 20, корегуючий коефіцієнт зменшить вагу ребра що відповідає за оцінку співпадіння цих параграфів.

4.4 Пошук найкоротшого шляху

На вході маємо зважений напрямлений ациклічний граф і початкову вершину s в ньому, задача — знайти найбільші відстані від s до всіх інших вершин даного графа.

Проблема найдовшого шляху для загального графа не така проста, як проблема найкоротшого шляху. Насправді, проблема найдовшого шляху є NP-важкою для загального графа. Однак задача найдовшого шляху має лінійне часове рішення для напрямлених ациклічних графів. Для цього використовується топологічне сортування [19].

Під топологічним сортуванням графа розуміють процес лінійного впорядковування його вершин таким чином, що якщо в графі існує ребро (a,b), то, в упорядкованому списку вершин графа, вершина a передує вершині b. Отримавши топологічний порядок (або лінійне представлення), ми по черзі обробляємо всі вершини в топологічному порядку. Для кожної вершини, яка обробляється, ми оновлюємо відстані сусідніх до неї, використовуючи відстань поточної вершини.

```
1) Поставимо dist[] = {NINF, NINF, ….} і dist[s] = 0 де s це початкова вершина, NINF це мінус безкінечність

2) Знайдемо топологічне впорядкування графу

3) Для кожної вершини u в топологічному порядку
```

```
Для кожної суміжної вершини v до u
  if (dist[v] < dist[u] + weight(u, v))
     dist[v] = dist[u] + weight(u, v)
```

Крім цього, в ході алгоритму зберігаємо та модифікуємо певну колекцію даних, яка по завершенні алгоритму дозволить відтворити шлях з найдовшою відстанню. Як уже згадувалось, пари множин параграфів, що відповідають вершинам з якого складатиметься цей шлях, будуть вважатися відповідними частинами різномовних текстів.

ВИСНОВКИ

У результаті виконання магістерської роботи було розроблено технологію, що забезпечує можливість створення синхронних текстів, створено веб-додаток, що реалізовує цю технологію. Створена база даних та мікросервіси, що забезпечують WEB-API для різноманітних цілей проекту. Було розроблено та впроваджено iOS та Android мобільні додатки для читання синхронних текстів. Також були проведені роботи по хостингу та серверному налаштуванні для проекту. Було пройдено всі етапи розробки, від постановки задачі, планування, вибору стеку технологій, до реалізації, тестування, впровадження та практичного використання.

У даній магістерській роботі було описано етапи розробки різних структурних елементів проекту. Зокрема сформовано проблему та постановку задачі, було знайдено та описано оригінальний спосіб розв’язання задачі, який було повністю програмно реалізовано.

У другому розділі увагу було присвячено загальній концепції роботи проекту. Було описано загальну архітектуру і компоненти з яких він складається та за допомогою яких буде вирішуватись поставлена задача. Зокрема було приділено увагу технологіям та програмним інструментам, що використовуються у проекті. Описано мікросервісну архітектуру та схему роботи веб-додатку. Було сформовано опис загального процесу об’єднання текстів, введено основні сутності та поняття, що стосуються цього процесу, такі як “завдання” та “книга”, було описано їх взаємодію один з одним у контексті процесу з’єднання.

У ході виконання проекту було створено базу даних для забезпечення всіх необхідних функціональних можливостей притаманних веб-додатку та мобільним додаткам. Також для виконання допоміжної

роботи по зберіганню даних, до проекту було підключено Google Cloud Storage.

Для функціонуванн веб-додатку та бази даних у веб-середовищі було придбано сервер, виділено та налаштовано віртуальну машину. Було пояснено переваги прийняття такого рішення. Також було описано деякі налаштування, що проводились на сервері з метою запуску та функціонування веб-додатку та бази даних, зокрема було описано необхідні для цього компоненти, які було встановлена на сервері.

Третій розділ роботи було присвячено мобільним додаткам. Було описано технології, що використовувались для розробки iOS та Android додатків. Були створені дизайн-макети для відповідності інтерфейсу сучасним вимогам. Додатки було впроваджено і вони доступні на площадках App Store та Play Market за ім’ям duoBooks.

В останньому розділі роботи розповідалось про технологію, що дозволила досить точно створювати синхронні тексти. Технологія використовує Natural Language Processing tools та деякі треновані моделі машинного навчання, а також подає задачу у вигляді графа та використовує алгоритм пошуку найдовшого шляху.

Загалом робота є комплексним повноцінним рішенням конкретної задачі, що включає в себе повний комплекс різноманітних елементів навчально-практичної, та деяких елементів навчально-дослідницької роботи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як вивчати англійську, читаючи книги [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://catalogueofarticles.com/uk/tehnologiyi/jak-vivchati-anglijsku-chitajuchi-knigi/.

2. Мікросервіси, їхні переваги, створення за допомогою фреймворка .NET — Internetdevels офіційний блог [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://internetdevels.ua/blog/building-microservices-dotnet

3. Kotlin VS Java: Basic Syntax Differences [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://yalantis.com/blog/kotlin-vs-java-syntax/.

4. Spring: the source for modern java [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://spring.io/.

5. What is MySQL? (thesitewizard.com) [Електронний ресурс] – Режим доступу до ресурсу: https://www.thesitewizard.com/faqs/what-is-mysql-database.shtml.

6. Spring Boot - Eureka Server [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/spring_boot/spring_boot_eureka_server.htm

7. What is Cloud Storage? | Google Cloud [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/storage/docs/introduction

8. HP ProLiant DL360p Gen8 (DL360 p G8) [Електронний ресурс] – Режим доступу до ресурсу: https://www.proliant.ru/catalog/servers/DL/HP_ProLiant_DL360p_Gen8.html

9. HPE ProLiant DL360p Gen8 Server - Overview [Електронний ресурс] – Режим доступу до ресурсу: https://support.hpe.com/hpesc/public/docDisplay?docId=emr_na-c03223744

10. VMware ESXi - Wikipedia [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/VMware_ESX

11. Debian -- About Debian [Електронний ресурс]. – Режим доступу до ресурсу: https://www.debian.org/intro/about

12. Systemd - Interface Portability and Stability [Електронний ресурс]. – Режим доступу до ресурсу: https://systemd.io/PORTABILITY_AND_STABILITY/

13. SSH Port [Електронний ресурс]. – Режим доступу до ресурсу: https://www.ssh.com/academy/ssh/port

14. What is Figma? [Електронний ресурс]. – Режим доступу до ресурсу: https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272

15. Model-View-Controller [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.apple.com/library/archive/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html

16. Building subject-aligned comparable corpora and mining it for truly parallel sentence pairs [Текст] : Krzysztof Marasek, Polish Japanese Institute of Information Technology, IICST 2014, 3-5 September 2014, Warsaw, Poland, – 130 с.

17. Efficient Estimation of Word Representations in Vector Space [Текст] : Mikolov, Tomas, 2013.

18. Cosine similarity - Wikipedia [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cosine_similarity

19. Longest Path in a Directed Acyclic Graph [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/find-longest-path-directed-acyclic-graph/