НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему:____Розробка системи збору даних про курси дистанційного навчання та впровадження її у систему «Cloud» _____

Виконав (-ла): студент (-ка) __4__ курсу, групи __ДА-32__
(шифр групи)

_____Намінас_Владислав_Вікторович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ___доц. к.т.н. Цурін О.П._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний _____доц. к.е.н. Рощина Н.В._____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____старший викладач Бритов О.А._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України

«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Намінасу Владиславу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи збору даних про курси дистанційного навчання та впровадження її у систему «Cloud»,
керівник роботи ____доц. к.т.н. Цурін Олег Пилипович_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи
   1. API популярних платформ з проектами дистанційного навчання.
   2. Дистанційне навчання в «Cloud».
   3. Інструменти веб-скрапінгу.
   4. Розробка мобільного додатку на ОС Android.

4. Зміст роботи

1. Огляд і порівняння існуючих реалізацій МВОК-агрегаторів.
2. Постановка та алгоритм розв’язку задачі
3. Класифікація курсів дистанційного навчання та побудова архітектури системи.
4. Розробка та аналіз програмного продукту
5. Функціонально-вартісний аналіз програмного продукту

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

1. Архітектура розробленої системи — плакат.
2. Ілюстрація роботи розробленого веб-сервісу — плакат.
3. Ілюстрація роботи розробленого Android-додатку — плакат.
4. Порівняння з існуючими реалізаціями МВОК-агрегаторів — плакат.
5. Презентація проекту.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина	Доц. к.е.н. Рощина Н.В.		

7. Дата видачі завдання _01.02.2017_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації	25.02.2017	
3	Проектування системи	17.04.2017	
4	Реалізація агрегатора даних	20.04.2017	
5	Реалізація веб-сервісу	04.05.2017	
6	Класифікація курсів ДН	11.05.2017	
7	Реалізація мобільного додатку	19.05.2017	
8	Оформлення дипломної роботи	29.05.2017	
9	Отримання допуску до захисту		

Студент _____ _____В.Н. Намінас_____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____О.П.Цурін_____
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

Тема роботи: Розробка системи збору даних про курси дистанційного навчання та впровадження її у систему “Cloud”.

Виконавець: Намінас Владислав Вікторович

Пояснювальна записка містить: 133 сторінки, 30 рисунків, з яких 8 діаграм, 11 таблиць, 3 додатки, 10 посилань.

Ключові слова: система збору даних, МВОК, курси дистанційного навчання, агрегатор , REST API, мобільний додаток.

Мета роботи: створення системи пошуку та класифікації навчальних ресурсів для дистанційного навчання.

Результати: веб-сервіс, що надає інформацію про доступні курси дистанційного навчання на МВОК-платформах з можливістю фільтрації та пошуку; мобільний додаток, що є клієнтом цього веб-сервісу та надає ті ж можливості; варіанти використання API веб-сервісу у системі “Cloud”.

АННОТАЦИЯ

Тема работы: разработка системы сбора данных про курсы дистанционного обучения и внедрение её в систему “Cloud”.

Исполнитель: Наминас Владислав Викторович

Пояснительная записка содержит: 133 страницы, 30 рисунков, из которых 8 диаграмм, 11 таблиц, 3 приложения, 10 ссылок.

Ключевые слова: система сбора данных, МООК, курсы дистанционного обучения, агрегатор , REST API, мобильное приложение.

Цель работы: создание системы поиска и классификации учебных ресурсов для дистанционного обучения.

Результаты: веб-сервис, предоставляющий информацию о доступных курсах дистанционного обучения на МООК-платформах с возможностью фильтрации и поиска; мобильное приложение, которое является клиентом этого веб-сервиса и предоставляет те же возможности; варианты использования API веб-сервиса в системе "Cloud".

.

ABSTRACT

R&D: develop data collection system of distance learning courses and implementation at the system "Cloud".

Author: Naminas Vladyslav

Work contains: 133 pages, 30 images, 11 tables, 3 attachments, 10 references.

Key words: data collection system, MOOC, distance learning courses, aggregator, REST API, mobile application.

The purpose: establishing a system of search and classification of educational resources for distance learning.

Results: a web service that provides information about distance learning courses available at MOOC platforms with the ability to filter and search; mobile application client of the service that provides the same features; using of the service API in the system "Cloud".

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ

МВОК — Масові відкриті онлайн курси.

ДН — Дистанційне навчання.

СУБД — Система управління базами даних.

NLP — Natural language processing.

SGD — Stochastic gradient descent.

ВСТУП

Мета, актуальність розробки об’єкту проектування

Концепція Всесвітньої павутини народилася в ЦЕРН в 1989 році. З тих пір пройшло менше 30 років, але більша частина людей вже не може уявити собі життя без Інтернету. Безсумнівно, він влився в наш світ і став невід’ємною частиною повсякденності. А завдяки Wi-fi і 3G ми можемо отримати до нього доступ майже з будь-якої точки планети.

Покупки онлайн, розваги, оплата комунальних послуг, спілкування, медіа-контент — завдяки Інтернету все це знаходиться всього в парі кліків. Інтернет можна з упевненістю назвати головним на даний момент джерелом інформації. Це ж стосується і освіти.

В ідеї онлайн-освіти немає нічого нового, однак лише в останні 7 років вона набула такої популярності. Онлайн-освіту важко переоцінити, адже це основне джерело самоосвіти — способу самостійного отримання знань в певній галузі науки, мистецтва, техніки, політичного життя, культури чи ремесла. Саме тому розробка технологій для розвитку цього напряму є актуальною та важливою темою.

Мета цього проекту — створення системи пошуку та класифікації навчальних ресурсів для дистанційного навчання та представлення цих результатів в зручному для користувача вигляді.

Обгрунтування основних проектних рішень або напрямків досліджень

RESTful API, що було вибрано для веб-сервісу як основний спосіб передачі інформації додатку, є найкращим для задачі передачі стану системи, тому що принципи REST є інтуїтивно зрозумілими для розробників та задовольняють простим потребам отримання даних з сервера.

Метод SVM для класифікації курсів дистанційного навчання був вибран на основі порівняльного аналізу з іншими методами класифікації.

Необхідність розробки мобільного додатку декларована тенденцією, що наблюдається на даний момент у світі — все більше людей відмовляється він настільних персональних комп'ютерів та переходить на планшети та смартфони для особистого користування.

Можливі галузі застосування результатів роботи

Результати, отримані в роботі, можуть бути застосовані як звичайними користувачами проектів онлайн-освіти, так і розробниками в цьому напрямку. Так як веб-сервіс, що був розроблений в цій роботі, є відкритим, розробники мобільних та веб-додатків, інженери та вчені в області науки про дані зможуть користуватись ним як в цілях розробки, так і в цілях аналізу даних.

1 ОГЛЯД І ПОРІВНЯННЯ ІСНУЮЧИХ РЕАЛІЗАЦІЙ МВОК-АГРЕГАТОРІВ

1.1 Основні поняття

1.1.1 Дистанційне навчання

Дистанційне навчання (ДН) - взаємодія вчителя і учнів між собою на відстані, що відображає всі властиві навчальному процесу компоненти (цілі, зміст, методи, організаційні форми, засоби навчання) та реалізовується специфічними засобами Інтернет-технологій або іншими засобами, які передбачають інтерактивність.

Дистанційне навчання - це самостійна форма навчання, інформаційні технології в дистанційному навчанні є провідним засобом.

У чому основні переваги онлайн-освіти перед звичним нам навчанням?

- доступ до інформації в будь-який зручний для учнів час;

- можливість вчитися в тому темпі, який підходить конкретному учневі;

- вибір напрямків, предметів і навіть викладачів;

- географічна доступність та інше.

Дистанційну форму навчання фахівці з стратегічним проблемам освіти називають освітньою системою 21 століття. Актуальність теми дистанційного навчання полягає в тому, що результати суспільного прогресу, раніше зосереджені в сфері технологій сьогодні концентруються в інформаційній сфері. Настала ера інформатики. Етап її розвитку в даний момент можна характеризувати як телекомунікаційний. Ця область спілкування, інформації і знань. Виходячи з того, що професійні знання старіють дуже швидко, необхідно їх постійне вдосконалення. Дистанційну форму навчання дає сьогодні

можливість створення систем масового безперервного самонавчання, загального обміну інформацією, незалежно від тимчасових і просторових поясів. Крім того, системи дистанційної освіти дають рівні можливості всім людям незалежно від соціального стану (школярам, студентам, цивільним і військовим, безробітними та т. Д.) В будь-яких районах країни і за кордоном реалізувати права людини на освіту і отримання інформації. Саме ця система може найбільш адекватно і гнучко реагувати на потреби суспільства і забезпечити реалізацію конституційного права на освіту кожного громадянина країни. Виходячи з названих вище факторів можна зробити висновок, що дистанційне навчання увійде в 21 століття як найефективніша система підготовки і безперервної підтримки високого кваліфікаційного рівня фахівців.

1.1.2 Масові Відкриті Онлайн Курси (МВОК)

Масовий відкритий онлайн-курс (МВОК) являє собою інтернет-курс з необмеженою участю і відкрим доступом через Інтернет. У доповненні до традиційних матеріалів курсу, такі як лекції, читання і задачі, більша частина МВОК надають інтерактивні форуми користувачів для підтримки взаємодії серед студентів, викладачів і асистентів.

Сам термін з'явився в 2008 році, його авторство приписують Д. Кормье і Б. Александер.

Вперше про відкриті онлайн-курсах заговорили більше десяти років тому, коли канадським педагогам з університету Манітоби вдалося запустити курс «Connectivism і Connective знань», який зібрав більше двох тисяч передплатників. З ростом кількості і популярності перших онлайн-курсів, стало зрозуміло, що це не новий модний інтернет-тренд, а сучасний відповідь на багато недоліків і упущення традиційної освітньої системи. Такі проблеми, як нестача компетентних викладачів, висока вартість навчання, неактуальне

матеріал і застарілі методи роботи з інформацією загальні для багатьох світових університетів. Онлайн-освіта була покликана їх вирішити.

Перша відкрита масова освітня платформа - Khan Academy (Академія Хана) - з'явилася в онлайн-просторі в 2006 році. Сьогодні Khan Academy надає більше п'яти тисяч абсолютно безкоштовні курси, а її команда з однієї людини виріс в колективі з 80 - ї співробітниками.

Але справжній ривок онлайн-освіти відбувся в 2012 році, який газета The New York Times назвала «роком масових відкритих онлайн курсів». Причиною тому стала поява сайтів EDX, Coursera і Udacity, які з моменту заснування отримали істотну фінансову підтримку.

І якщо ще десять років тому записатися на курси Массачусетського технологічного інституту або вивчати основи програмування під керівництвом викладачів Гарвардського університету могла тільки жменька обраних студентів, яким таке навчання обходилося в десятки тисяч доларів, з появою університетських курсів онлайн безкоштовний доступ до кращих кладезем знань на планеті став відкрито всім, у кого є комп'ютер і підключення до інтернету.

У той же час, стало зрозуміло, що потенціал онлайн-освіти набагато більше, ніж можливість слухати лекції в мережі. Інтернет не тільки зробив навчання доступним широким масам людей (як це сталося у випадку з поширенням університетських курсів), але і повністю змінив сам підхід до процесу навчання [1].

Це, в свою чергу, дає значну перевагу при пошуку нової роботи. Саме тому слідом за платформи, що надає доступ до онлайн-курси університетів, все активніше став розвиватися сервіси інший формату. Вони вибрали інший шлях - структурування навчального контенту, зміна форматів подачі матеріалу і орієнтацію на практичні професійні теми. Результатом стала можливість вивчити невеликий обсяг інформації, яку можна негайно застосувати на

практиці. Найбільш популярні серед таких сервісів - Udacity, Udemy, Lyndy безліч інших.

Цінність освітніх платформ в тому, що вони видають строго необхідну інформацію. Питання відсікання непотрібного - найгостріший за методичної опрацювання програм.

Крім можливості швидко отримати нову кваліфікацію або оновити існуючу, короткі онлайн-формати зробили більш доступним отримання знань в області хобі.

Згадувані вище платформи, як Coursera, Udacity, EDX, Udemy - лише одні з найбільш популярних. Існує десятки МВОК платформ, в тому числі відомий в Україні Prometheus.

1.1.3 Агрегатори МВОК

З огляду на популярність, швидке поширення і зростання МВОК-платформ, заходити на кожну з них в пошуках потрібних курсів стало незручно. Для вирішення цієї проблеми з'явилися агрегатори МВОК.

Основні переваги агрегаторів МВОК:

• **Великий вибір**. Представлений великий каталог різноманітних онлайн-курсів від різних провайдерів.

• **Можливість порівняти**. Багато агрегаторів публікують рейтинги онлайн-курсів та відгуки слухачів.

• **Менше часу на пошук**. Це не просто посилання на онлайн-платформи з курсами. Агрегатори дозволяють легко знаходити онлайн-курси за спеціальністю, мовою викладання, викладачем та іншими параметрами відразу за кількома онлайн-платформами.

• **Можливість скласти індивідуальний план навчання**. Багато агрегаторів дають можливість зберегти зацікавлені вас онлайн-курси в особистому кабінеті і встановити нагадування про наближення дати їх початку.

1.2 Порівняння та аналіз існуючих веб-сервісів агрегаторів МВОК

Список МВОК-агрегаторів:

- CourseBuffet
- Degreed
- Mooc Advisor
- EClass
- Class Central
- CourseTalk
- MOOC-list
- Mooctivity

1.2.1 Mooctivity

Рисунок. 1.1 — Сайт Mooctivity.com

Сайт: http://www.mooctivity.com/

Переваги: підтримується список з декількох адрес електронної пошти.

Недоліки: мало платформ, відсутність сортування.

1.2.2 MOOC-list

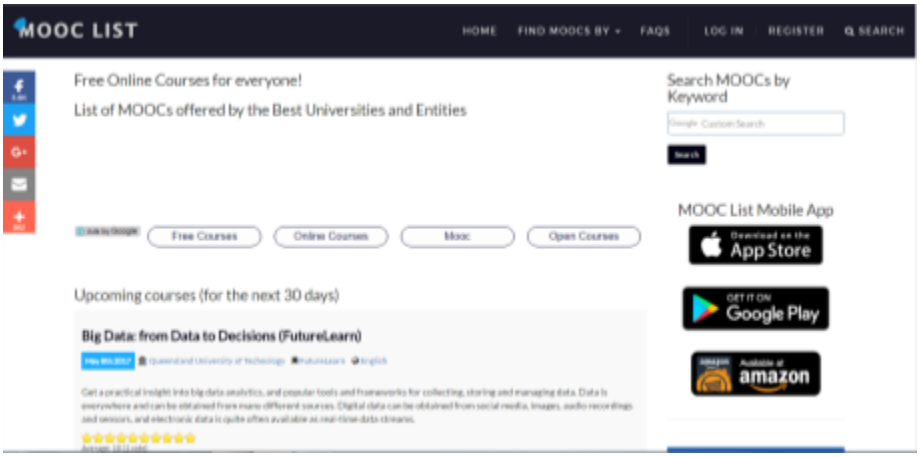


Рисунок 1.2 — Сайт mooc-list.com

Сайт: https://www.mooc-list.com/

Переваги: є мобільний додаток, підтримує велику кількість платформ, безліч критеріїв для фільтрації і сортування.

Недоліки: відносно незручний інтерфейс сайту.

1.2.3 CourseTalk

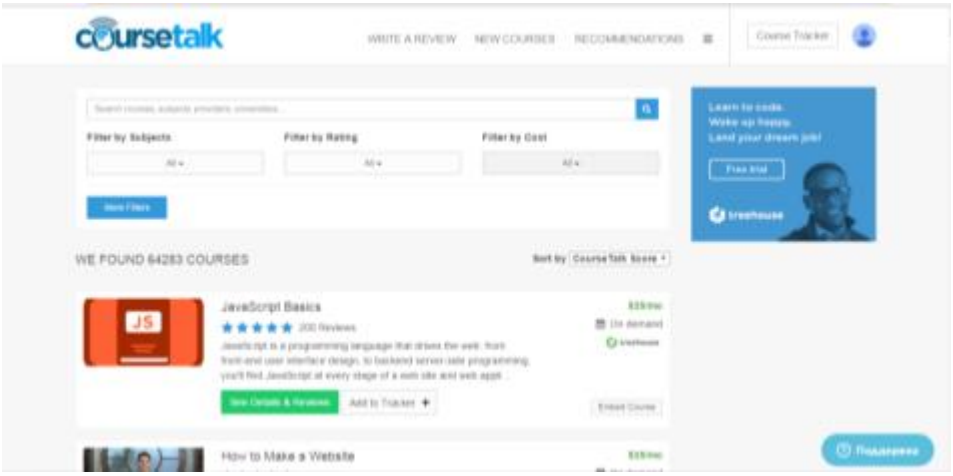


Рисунок 1.3 — Сайт coursetalk.com

Сайт: https://www.coursetalk.com/

Переваги: підтримка великої кількості платформ, цікавий курс-трекер, існує course advisor, що базується на активності користувача.

Недоліки: нема фільтрації за мовою.

1.2.4 Class Central

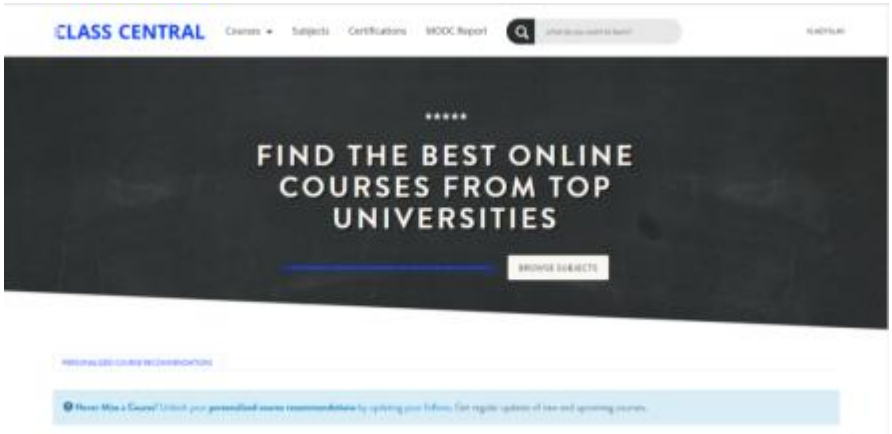


Рисунок 1.4 — Сайт class-central.com

Сайт: https://www.class-central.com/

Переваги: щомісячні статті про найбільш популярні курси.

Недоліки:

1.2.5 EClass

Рисунок. 1.5 — Сайт eclass.cc

Сайт: http://eclass.cc/

Переваги: інтерфейс на декількох мовах, підтримка не тільки МВОК-платформ.

Недоліки:

1.2.6 Learning Advisor

Рисунок 1.6 — Сайт learningadvisor.com

Сайт: http://www.learningadvisor.com/

Переваги:

Недоліки: відносно обмежена функціональність.

1.2.7 Degreed

Рисунок 1.7 — Сайт degreed.com

Сайт: https://degreed.com/

Переваги: особлива система навчання: в залежності від обраного напрямку генерує новинну стрічку з статей, людей, курсів та іншого. Є мобільний додаток.

Недоліки: складно зорієнтуватися в ресурсі, якщо не йти за планом, наданим програмно.

1.2.8 CourseBuffet

Рисунок 1.8 — Сайт coursebuffet.com

Сайт: https://www.coursebuffet.com/

Переваги: крім курсів є можливість так само шукати шляхи, за рівнем фільтрації (cs0-cs600).

Недоліки: немає можливості вибрати курси на інших, крім англійської, мовами.

В таблицях 1.1-1.3 порівнюються можливості фільтрації, пошуку та інші, що надають розглянуті MBOK-агрегатори (+ означає, що можливість надається).

Таблиця 1.1

Агрегатор	Фільтрація за:			
	мовою	платформою	категорією	тегами
Mooctivity	+	+	+	
Mooc List	+	+	+	+
CourseTalk		+	+	
Class Central	+		+	
EClass	+	+	+	
Learning Advisor		+	+	
Degreed		+		+
CourseBuffet		+	+	

Таблиця 1.2

Агрегатор	Фільтрація за:				
	заванта женніст ю	триваліс тю	Датою початку / закінчення	Мовою субтитрів	Вартістю
Mooctivity	+	+			
Mooc List	+	+	+	+	+

Таблиця 1.2 (продовження)

1	2	3	4	5	6
CourseTalk					+
Class Central			+		
EClass					+
Learning Advisor					
Degreed		+	+		
CourseBuffet	+				

Таблиця 1.3

Агрегатор	Сортування за:					Зберігає вибрані курси
	датою початку	рейтингом	назвою	датою закінчення	вартістю	
Mooctivity						+
Mooc List	+	+	+			+
CourseTalk	+	+	+		+	+
Class Central	+	+	+			+
EClass	+	+	+			
Degreed						+

1.3 Порівняння та аналіз існуючих мобільних додатків агрегаторів МВОК

Android-додатки:

- Поиск курсов
- Degreed
- MOOC-list

1.3.1 “Поиск курсов”

Додаток: (зображено на рис. 1.9)
https://play.google.com/store/apps/details?id=com.brodski.android.coursefinder

Переваги:

Недоліки: вкрай обмежений функціонал і стиль.

Рисунок 1.9

1.3.2 MOOC-List

Додаток: (зображено на рис. 1.10)

https://play.google.com/store/apps/details?id=pt.cotonet.mooclist

Переваги:

Недоліки: незручний інтерфейс, неможливість пошуку.

Рисунок 1.10

1.3.3 Degreed

Додаток: (зображено на рис. 1.11)

https://play.google.com/store/apps/details?id=com.degreed.android

Переваги: зручний інтерфейс, майже повністю повторює можливості, що представлені на сайті

Недоліки: не реалізовує необхідний функціонал пошуку.

Рисунок 1.11

1.4 Висновки

Порівнюючи доступні рішення задачі створення системи агрегації інформації про курси дистанційного з різних платформ, можна зробити висновок, що у кожної з них є свої недоліки та переваги.

З розглянутих MBOK-агрегаторів мобільний додаток мають лише двоє: “Degreed” та “MOOC-List”; але “Degreed” є проектом дещо іншого направлення і не реалізує необхідний функціонал пошуку курсів дистанційного навчання.

Найбільшими функціональними можливостями володіє “MOOC-List”. Вона має широкі можливості пошуку:

- багато різноманітних параметрів фільтрації
- сортування за датою, рейтингом, назвою
- можливість зберігання вибраних курсів

Також для неї існує мобільний додаток. Але, незважаючи на це, ряд недоліків змушує відмовитись від її використання та створити своє рішення:

- немає відкритого API, що ускладнює можливість її використання в інших системах (зокрема в системі «Cloud»)
- незручний інтерфейс веб-сайту та мобільного додатку
- обмежений функціонал мобільного додатку (єдина можливість, що він надає — вибір курсів дистанційного навчання за категоріями).

Таким чином, проаналізувавши доступні рішення, робиться висновок у необхідності реалізації своєї системи для реалізації поставлених задач.

2 ПОСТАНОВКА ТА АЛГОРИТМ РОЗВ'ЯЗКУ ЗАДАЧІ

2.1 Мета дипломної роботи

Створення системи пошуку та класифікації навчальних ресурсів для дистанційного навчання.

2.2 Огляд літератури

Основна джерела інформації, що необхідні для вирішення заданої задачі, це документація фреймворків, що будуть використовуватися (Django, Android SDK, DBFlow та інші), API, що надають MBOK платформи та опис алгоритмів текстової категоризації.

Окрім цього також розглядаються статті, що стосуються актуальності роботи та альтернативні реалізації.

2.3 Задачі роботи

Задачами дипломної роботи є:

1) Аналіз доступних MBOK-платформ та їх API.

2) Проектування архітектури веб-сервісу пошуку та класифікації курсів з MBOK-платформ.

3) Реалізація цього веб-сервісу.

4) Тестування цього веб-сервісу.

5) Реалізація android-клієнту розробленого веб-сервісу.

5) Тестування реалізованого android-додатоку.

2.4 Цілі роботи, ключові фактори успіху результатів роботи

Основними цілями бакалаврської роботи є зниження затрат часу на пошук та моніторинг шуканих онлайн курсів, надання зручного інтерфейсу для цих цілей (у вигляді Android-додатку).

Основним фактором успіху бакалаврської роботи є таке виконання цілей проекту, при якому програмний продукт надає всі задані можливості та є зручним для користування. Але важливими також є і чітко сформульована мета роботи та спроектовані етапи роботи, доступність теоретичних даних та опису використання подібних методів.

2.5 Вихідні дані до роботи

Вихідні дані до роботи:

5. API популярних платформ з проектами дистанційного навчання.
6. Дистанційне навчання в «Cloud».
7. Інструменти веб-скрапінгу.
8. Розробка мобільного додатку на ОС Android.

2.6 ARIS Objective Diagram

На рисунку 2.1 зображена діаграма цілей дипломного проекту. «Розробка інформаційної системи» та «Розробка Android-клієнту цієї інформаційної системи» описані в розділі 4 цього документу та в додатках Б та В.

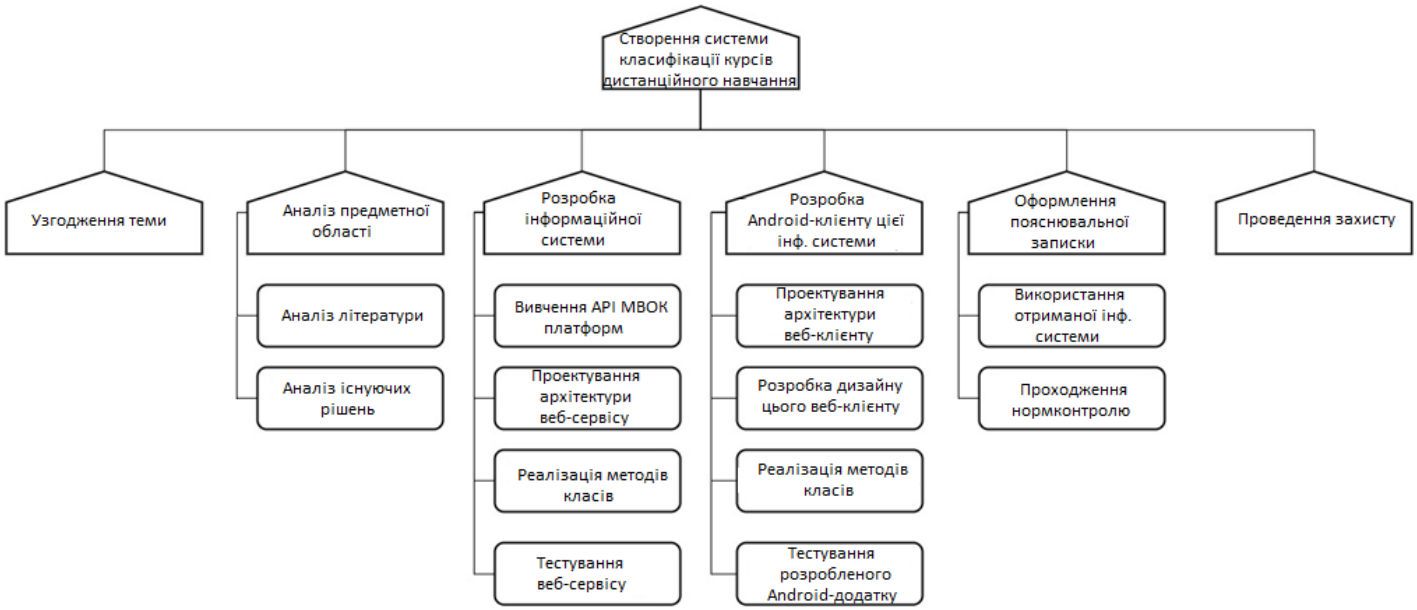


Рисунок 2.1. — Діаграма цілей дипломного проекту

2.7 Розгорнуті вимоги до сервісу та мобільного додатку

Зважаючи на можливості вже реалізованих агрегаторів MBOK, можна виділити функціонал, яким повинен володіти сервіс, що необхідно реалізувати.

Можливості агрегаторів:

- створюється аккаунт на людину, підтримується список обраних ним курсів.
- фільтрація за: мовою, провайдером (платформою), категорією, тегами, завантаженості (workload в годин / тиждень), тривалістю, мовами субтитрів (підтримка вибору відразу декількох мов), країною, вартістю, сертифікатам.
- для кожного курсу присутній рейтинг (5 зірок), статус (upcoming, current, past), необхідний рівень знань (якщо є), опис, посилання на курс, список тегів, мова, вартість.
- головна інформація про курс у списку: назва, посилання, рейтинг, мова, короткий опис (або усічений).
- сортування за: датою початку, рейтингом, назвою.

- список популярних курсів.
- підтримується список з декількох email-адрес.
- статистика по курсам.
- course advisor - підказує курси, базуючись на історії пошуку та іншої активності.
- генерація стрічки новин за обраними перевагам.

Вибираємо основне:

- фільтрація за: мовою, рейтингом, категорією.
- для кожного курсу присутній рейтинг (5 зірок), опис, посилання на курс, мова.
- сортування за: рейтингом, назвою.

Таким чином буде створений веб-сервіс з API, що надає можливості пошуку онлайн-курсів з платформ Stepik, Coursera, Udacity, що були описані вище. Також буде створений Android додаток — клієнт цього API.

2.8 DFD (IDEF0) діаграми

На рис. 4.2-4.4 знаходяться DFD-діаграми сервісу-агрегатору. DFD-діаграма описує роботу системи у вигляді блоків, що можна декомпозувати.

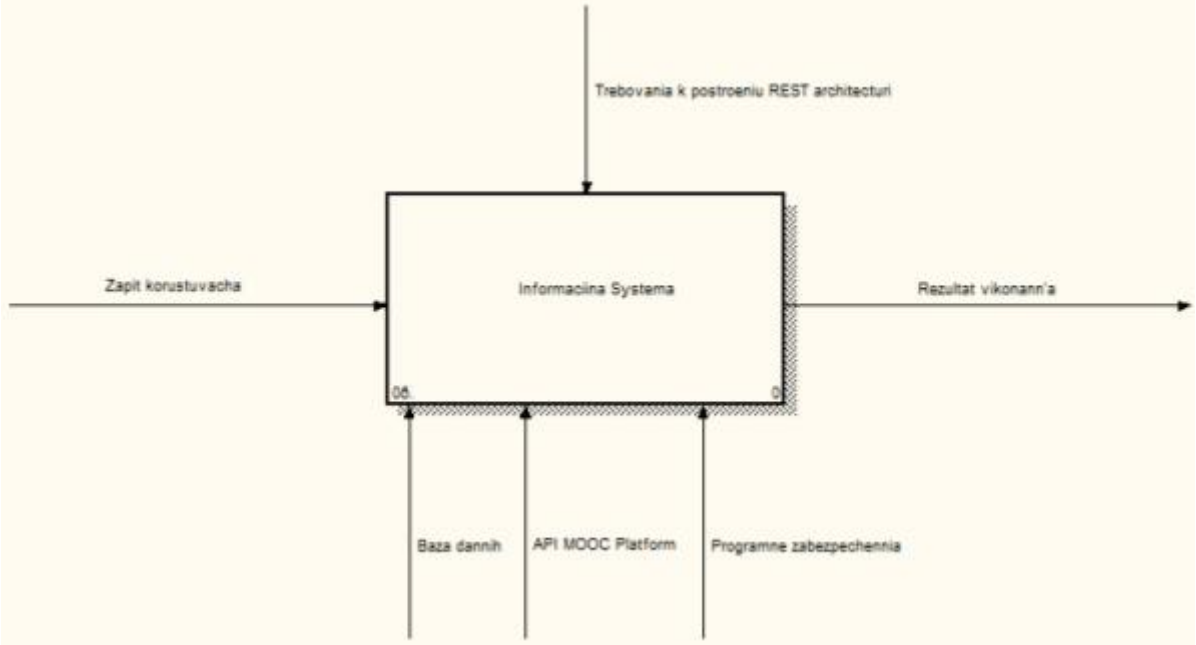


Рисунок 4.2 — Загальна DFD діаграма

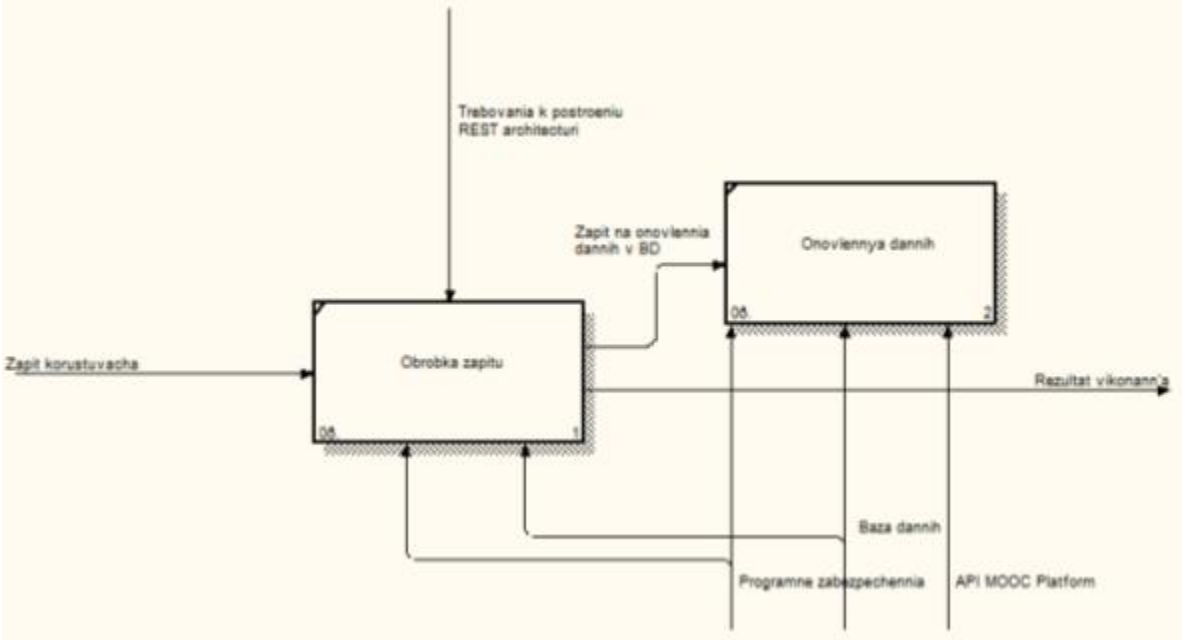


Рисунок 4.3 — Загальна DFD діаграма

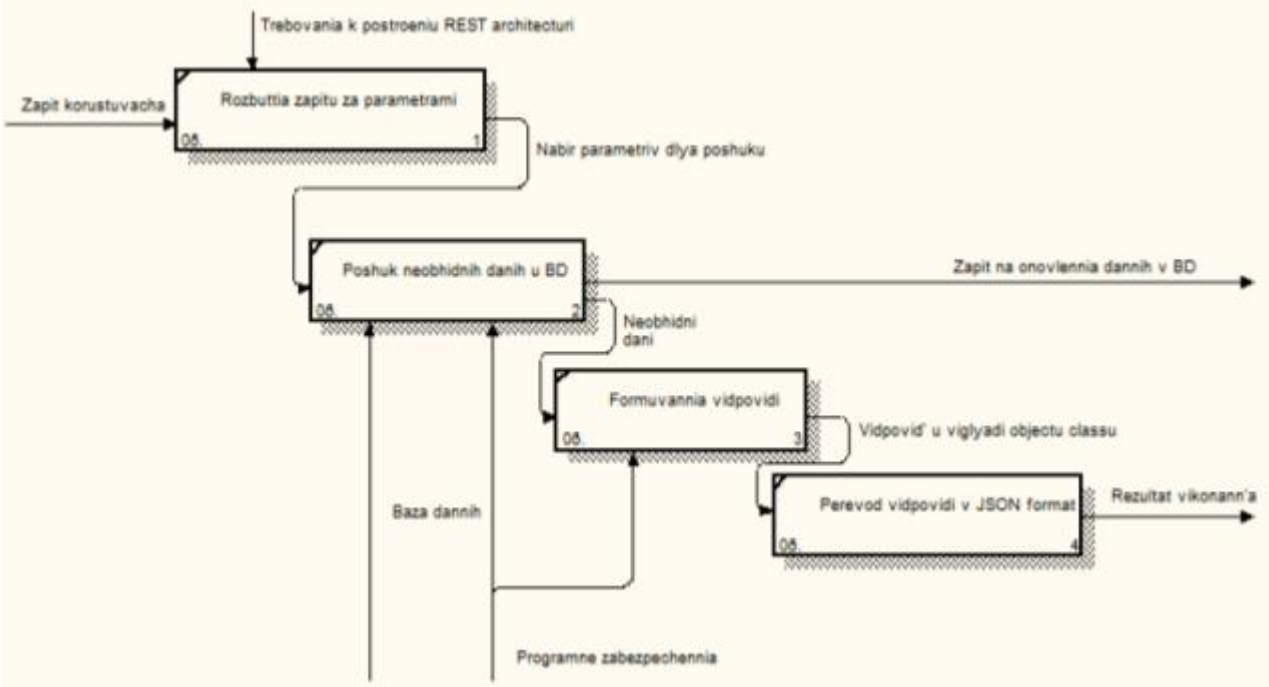


Рисунок 4.4 — Декомпозиція блоку “Обробка запиту”

2.9 Висновки

Сформульовано мету роботи, окреслено основні вимоги до неї. Визначено основні характеристики системи, що має бути розроблена в ході виконання роботи.

Алгоритм розв’язку задачі:

1. Вибір найбільш популярних MBOK-платформ.
2. Для кожної платформи, якщо для неї існує відкрите API, то реалізувати модуль, що отримує інформацію про курси ДН за допомогою нього, інакше — реалізувати модуль парсингу веб-сайту цієї платформи для отримання інформації про курси ДН.
3. Вибрати категорії, за якими будуть розподілятися курси.
4. Отримати теги для курсів ДН.

5. Розподілити отримані курси ДН за вибраними категоріями.
6. Спроектувати та реалізувати REST API для доступу до інформації про курси ДН.
7. Використовуючи реалізоване API розробити мобільний додаток.

3 КЛАСИФІКАЦІЯ КУРСІВ ДИСТАНЦІЙНОГО НАВЧАННЯ ТА ПОБУДОВА АРХІТЕКТУРИ СИСТЕМИ

Розробляючи агрегатор курсів ДН стає проблема їх класифікації та пошуку тегів. Ця проблема є актуальною, тому що:

- на різних МВОК-платформах курси ДН розподіляються за різними категоріями
- на деяких МВОК-платформах розподілення за категорії немає
- фільтрація за категоріями є одною з основних можливостей МВОК-агрегатора.

Для вирішення цих задач існує напрямок аналізу даних, інформатики та математичної лінгвістики, що називається “NLP” або “Методи обробки природної мови”.

3.1 Основні поняття

3.1.1 NLP

Обробка природної мови — загальний напрямок інформатики, штучного інтелекту та математичної лінгвістики. Він вивчає проблеми комп'ютерного аналізу та синтезу природної мови. Стосовно штучного інтелекту аналіз означає розуміння мови, а синтез — генерацію розумного тексту. Розв'язок цих проблем буде означати створення зручнішої форми взаємодії комп'ютера та людини.

NLP — спосіб для комп'ютерів аналізувати, розуміти і витягувати сенс з людської мови в розумний і корисний спосіб. Використовуючи NLP, розробники можуть організувати і структурувати знання для виконання таких завдань, як автоматичне реферування, переклад, аналіз настроїв, розпізнавання мови та інших.

3.1.2 Токенізація (Лексичний аналіз)

В інформатиці лексичний аналіз («токенізація», від англ. Tokenizing) - процес аналітичного розбору вхідної послідовності символів на розпізнані групи - лексеми, з метою отримання на виході ідентифікованих послідовностей, званих «токенами» (подібно до угрупованню букв в словах). У простих випадках поняття «лексема» і «токен» ідентичні, але більш складні токенізатори додатково класифікують лексеми по різним типам («ідентифікатор, оператор», «частина мови» і т. П.). Лексичний аналіз використовується в компіляторах і інтерпретатора вихідного коду мов програмування, і в різних парсерах слів природних мов.

Як правило, лексичний аналіз проводиться з точки зору певної формальної мови або набору мов. Мова, а точніше її граматика, задає певний набір лексем, які можуть зустрітися на вході процесу.

3.1.3 Стемінг

Стемінг - це процес знаходження основи слова для заданого вихідного слова. Основа слова не обов'язково збігається з морфологічним коренем слова.

Завдання знаходження основи слова являє собою давню проблему в області комп'ютерних наук. Перша публікація з цього питання 1968 роком. Стемінг застосовується в пошукових системах для розширення пошукового запиту користувача, є частиною процесу нормалізації тексту.

Конкретний спосіб вирішення завдання пошуку основи слів називається алгоритм стемінгу, а конкретна реалізація – Стемер.

3.1.4 TF-IDF

TF-IDF (от англ. TF — term frequency, IDF — inverse document frequency) — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален количеству употребления этого слова в документе, и обратно пропорционален частоте употребления слова в других документах коллекции.

Мера TF-IDF часто используется в задачах анализа текстов и информационного поиска, например, как один из критериев релевантности документа поисковому запросу, при расчёте меры близости документов при кластеризации [2].

Если документ содержит 100 слов и слово «заяц» встречается в нём 3 раза, то частота слова (TF) для слова «заяц» в документе будет 0,03 (3/100). Вычислим IDF как десятичный логарифм отношения количества всех документов к количеству документов содержащих слово «заяц». Таким образом, если «заяц» содержится в 1000 документов из 10 000 000 документов, то IDF будет равной: log(10 000 000/1000) = 4. Для расчета окончательного значения веса слова необходимо TF умножить на IDF. В данном примере, TF-IDF вес для слова «заяц» в выбранном документе будет равен: 0,03 × 4 = 0,12.

3.1.5 Класифікація

Класифікації є продуктом організації речей в види речей (класи) або на групи класів.

Теоретично, отримані в результаті структури є одним з найважливіших аспектів метаданих, часто представлені у вигляді ієрархічної структури і в супроводі описової інформації з класів або груп. Така схема класифікації

призначена для використання для розташування або поділів окремих об'єктів на класи або групи, а класи або групи засновані на характеристиках, що об'єкти (члени) мають в загальному.

3.2 Алгоритми класифікації

3.2.1 Наївний байесовский класифікатор

Наївний байесовский класифікатор - простий імовірнісний класифікатор, заснований на застосуванні Теорема Баєса зі строгими (наївними) припущеннями про незалежність [3].

3.2.2 Метод опорних векторів (SVM)

Метод опорних векторів (англ. SVM, support vector machine) - набір схожих алгоритмів навчання з учителем, що використовуються для задач класифікації та регресійного аналізу. Належить сімейству лінійних класифікаторів і може також розглядатися як спеціальний випадок регуляризації за Тихоновим. Особливою властивістю методу опорних векторів є невпинне зменшення емпіричної помилки класифікації і збільшення зазору, тому метод також відомий як метод класифікатора з максимальним зазором.

Основна ідея методу - переклад вихідних векторів в простір більш високої розмірності і пошук роздільної гіперплощини з максимальним зазором в цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє класи. Роздільна гіперплощина є гіперплощина, що максимізує відстань до двох паралельних гіперплощин. Алгоритм працює в припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим менше буде середня помилка класифікатора[3].

3.2.3 Градієнтний спуск

Градієнтний спуск - метод знаходження локального екстремуму (мінімуму або максимуму) функції за допомогою руху уздовж градієнта. Для мінімізації функції в напрямку градієнта використовуються методи одновимірної оптимізації, наприклад, метод золотого перерізу. Також можна шукати не найкращу точку в напрямку градієнта, а будь-яку краще поточної.

3.2.4 Метод k-найближчих сусідів

Метод k-найближчих сусідів (англ. K-nearest neighbors algorithm, k-NN) - метричний алгоритм для автоматичної класифікації об'єктів. Основним принципом методу найближчих сусідів є те, що об'єкт присвоюється тому класу, який є найбільш поширеним серед сусідів даного елемента.

Сусіди беруться виходячи з безлічі об'єктів, класи яких вже відомі, і, виходячи з ключового для даного методу значення k вираховується, який клас найбільш численний серед них. Кожен об'єкт має кінцеву кількість атрибутів (розмірностей).

3.2.5 Дерево прийняття рішень (Decision Trees)

Дерево прийняття рішень (також може називатися деревом класифікації або регресійний деревом) - засіб підтримки прийняття рішень, що використовується в статистиці і аналізі даних для прогнозних моделей. Структура дерева являє собою «листя» і «гілки». На ребрах («гілках») дерева рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах - атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення. Подібні дерева рішень широко використовуються в

інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної на основі декількох змінних на вході.

Каждый лист представляет собой значение целевой переменной, измененной в ходе движения от корня по листу. Каждый внутренний узел соответствует одной из входных переменных. Дерево может быть также «изучено» разделением исходных наборов переменных на подмножества, основанные на тестировании значений атрибутов. Это процесс, который повторяется на каждом из полученных подмножеств. Рекурсия завершается тогда, когда подмножество в узле имеет те же значения целевой переменной, таким образом, оно не добавляет ценности для предсказаний.

3.2.6 Нейронні мережі (Neural networks)

Штучна нейронна мережа (ШНМ) - математична модель, а також її програмне або апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж — мереж нервових клітин живого організму.

ШНС є системою з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості (особливо в порівнянні з процесорами, що використовуються в персональних комп'ютерах). Кожен процесор подібної мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І, тим не менше, будучи з'єднаними в досить велику мережу з керованою взаємодією, такі окремо прості процесори разом здатні виконувати досить складні завдання.

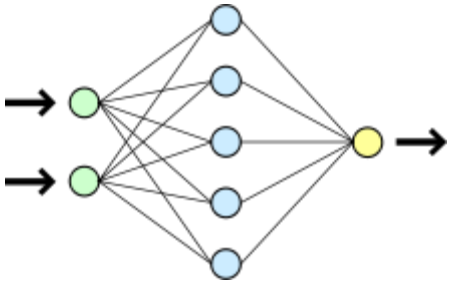


Рисунок 3.1 — Схема простої нейронної мережі

3.3 Інструменти аналізу даних

Для аналізу даних в роботі були використані такі інструменти:

- Бібліотека NLTK, або NLTK - пакет бібліотек і програм для символьного і статистичної обробки природної мови, написаних на мові програмування Python[4]. Містить графічні уявлення і приклади даних. Супроводжується великої документацією, включаючи книгу з поясненням основних концепцій, що стоять за тими завданнями обробки природної мови, які можна виконувати за допомогою даного пакету.
- Scikit-learn (раніше scikits.learn) є вільним програмним забезпеченням — бібліотекою машинного навчання для мови програмування Python [5]. Він має різні класифікації, регресію і алгоритми кластеризації, включаючи підтримку SVM, random-forests, градієнт підвищення, K-means і DBSCAN, і призначений для взаємодії з чисельними і науковими бібліотеками Python — NumPy і SciPy.

3.4 Отримання тегів

Окрім категорії ще одним важливим параметром курсу є теги. В інформаційних системах теги є неструктурованими ключовими словами, що відносяться до частин інформації (це можуть бути закладки браузера, цифрові зображення,

файли). Такі метадані покликані описати ці частини інформації і допомагають знаходити їх в процесі перегляду або через пошуковий запит.

Теги допомагають зрозуміти основний зміст, при пошуку та більш детальної класифікації.

3.4.1 Алгоритм отримання тегів

Для вибору тегів був вибраний такий алгоритм дій:

1. Збираємо всі текстові дані про курс (назва, опис, метаінформація) в один.

   Для прикладу візьмемо такий текст:

   «Алгоритмы: теория и практика. Структуры даннях. В курсе будут рассмотрены структуры данных, наиболее часто использующиеся на практике: массивы, списки, очереди, стеки, динамические массивы, очереди с приоритетами, системы непересекающихся множеств, хеш-таблицы, сбалансированные деревья. »

2. Позбавляємось розділових знаків та приводимо до малих літер:

   «алгоритмы теория и практика структуры даннях в курсе будут рассмотрены структуры данных наиболее часто использующиеся на практике массивы списки очереди стеки динамические массивы очереди с приоритетами системы непересекающихся множеств хеш-таблицы сбалансированные деревья»

3. Перекладаємо англійською (для перекладу використовувалось безкоштовне Translation API — transltr.org).

   «algorithms theory and practice of the structure of data in the course will be considered data structures the most frequently used in practice arrays queue lists stacks dynamic arrays of queues with priorities of the system of disjoint sets of hash tables balanced trees»

4. Позбавляємося стоп-слів (до таких відносяться сполучники, вигуки, частки, займенники та інше):

«algorithms theory practice structure data course considered data structures most frequently used practice arrays queue lists stacks dynamic arrays queues priorities system disjoint sets hash tables balanced trees»

5. Токенізуємо.
6. Отримаємо tf-idf значення для усіх даних [2].
7. Зберігаємо для кожного тексту тільки ті фрази, що мають найбільший коефіцієнт tf-idf . Були взяті фрази розміром 1-2 слова. З кожного тексту вибиралося по 5 таких слів. Для прикладу, що наведений вище, це можуть бути «algorithms», «structure data», «data structures», «arrays», «trees».
8. Вручну позбавляємось «зайвих» фраз. Це ті фрази, що не можуть бути тегами, через те, що вони не описують ніяку категорію, не мають сенсу окремо від контексту, чи за інших причин. Для прикладу вище можна залишити «algorithms» та «data structures».
9. Для кожної фрази застосовуємо стемінг. «algorithms» -> «algorithm», «data structures» -> «data struct».
10. Позбавляємося «стем-копій». Тобто якщо існує дві фрази з однаковим стем-значенням, залишаємо тільки одну з них. Наприклад, фрази «data structures» та «data structure» мають одне й те саме стем-значення «data struct», тому залишаємо тільки «data structures».

3.4.2 Результати

Було оброблено 2548 текстів та виділено близько 10000 ключових слів. В результаті отримано 845 тегів.

Наприклад, для курсу «Закрытая группа Яндекс.Директ», в якому йдеться про те, як правильно рекламувати сайти, було знайдено такі теги:

- marketing
- analytic
- advertise
- freelance
- data analytics

Вони повністю відповідають темі курсу.

3.5 Класифікація курсів

Для класифікації курсів було вибрано 10 категорій:

- Personal Development / Особистий розвиток
- Languages / Мови
- Data Science / Наука про дані
- Mathematics / Математика
- Biology and Medical Sciences / Біологія й Медицина
- Physical Sciences & Engineering / Природничі наука та інженерія
- Business & Management / Бізнес й управління
- Social Sciences / Соціальні науки
- Humanities / Гуманітарні науки
- Computer Science / Комп’ютерні науки

3.5.1 Етапи класифікації

Частина алгоритму класифікації, що стосується передобробки даних, збігається з частиною алгоритму знаходження тегів, тому ці етапи не будуть детально описуватись.

Категорії, на які розбиті курси на платформі Coursera можна однозначно відобразити в вибрані категорії. Це надає змогу сформувати навчальну вибірку для класифікатора на основі даних з курсів на платформі Coursera.

Етапи класифікації курсів:

1. Збираємо всі текстові дані про курс.
2. Позбавляємось розділових знаків та приводимо до малих літер.
3. Перекладаємо англійскою.
4. Позбавляємося стоп-слів.
5. Токенізуємо.
6. Для кожного точену застосовуємо стемінг.
7. Отримаємо Tf-Idf значення для навчальної вибірки [2]
8. Класифікуємо за допомогою алгоритму SVM з використанням SGD [3].

3.5.2 Порівняння розглянутих алгоритмів класифікації

Окрім алгоритму SVM з використанням SGD були розглянуті та порівняні й інші алгоритми класифікації. Результати порівняння наведені в таблиці 3.1.

Для отримання відсотку правильних відповідей навчальна вибірка була розбита на дві — навчальну та тестову — з відношенням 60/40.

Таким чином, навчальна вибірка становить 1293 тексти, тестова — 862 тексти.

Таблиця 3.1. Порівняння алгоритмів класифікації

Алгоритм класифікації	Клас, яким він представлений в Scikit-learn [5]	Відсоток правильних відповідей
Наївний Байесовський класифікатор	MultinomiaNB	32%
Нейронна мережа	MLPClassifier	74%
Random forest алгоритм	RandomForestClassifier	27%
Ada Boost алгоритм	AdaBoostClassifier	34%
Дерево рішень	DecisionTreeClassifier	37%
Метод k-найближчих сусідів	KNeighborsClassifier	17%
Лінійна SVM	SVC	27%
SVM з використанням SGD	SGDClassifier	79%

Краще за все з поставленою задачею впорався алгоритм класифікації SVM з використанням SGD.

3.5.3 Результати

Було виділено 10 категорій для класифікації курсів ДН. Після передобробки, тексти, що були отримані з описів та метаданих курсів, були класифіковані за цими категоріями. Всього класифіковано 393 курси ДН.

Для вибору алгоритму класифікації класифіковані дані були розбиті на навчальну та тестову вибірки та проведений порівняльний аналіз 8 алгоритмів. З них був вибраний найкращий за відсотком правильних відповідей для тестової вибірки — SVM з використанням SGD.

Наприклад, курс «Закрытая группа Яндекс.Директ», в якому йдеться про те, як правильно рекламувати сайти, був класифікован як «Business & Management / Бізнес й управління», що відповідає дійсності.

3.6 Архітектурна модель системи

На малюнку 3.2 представлена архітектурна модель системи, що реалізовувалась, та її зв’язок з системою «Cloud».

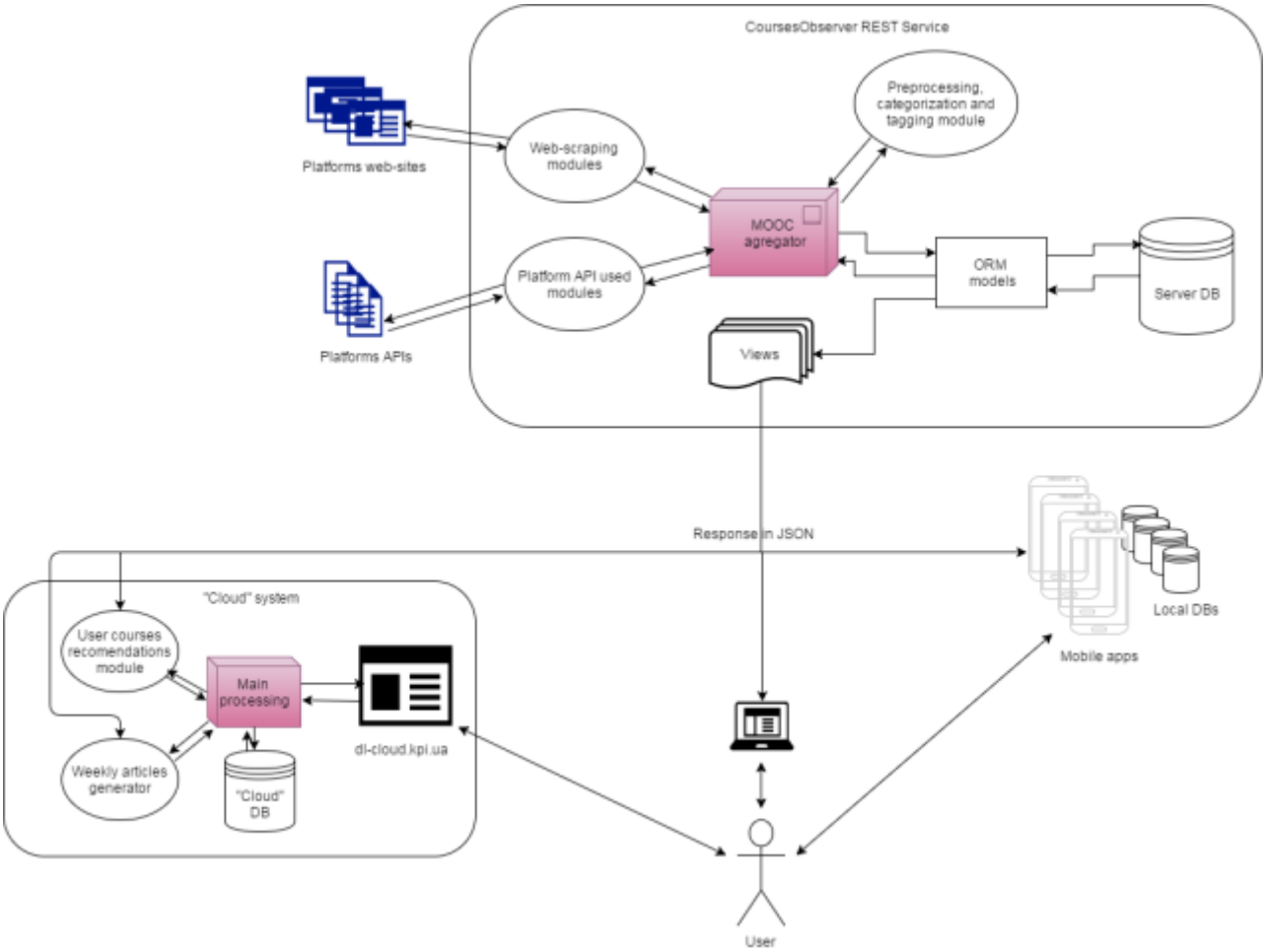


Рисунок 3.2 — Архітектурна модель проекту

3.7 Загальна UML діаграма прецедентів

На діаграмі прецедентів (рис.3.3) представлена основна функція, що надає сервіс агрегатор-МВОК — пошук онлайн-курсів. Його розширенням є пошук за різними параметрами, що задає користувач.

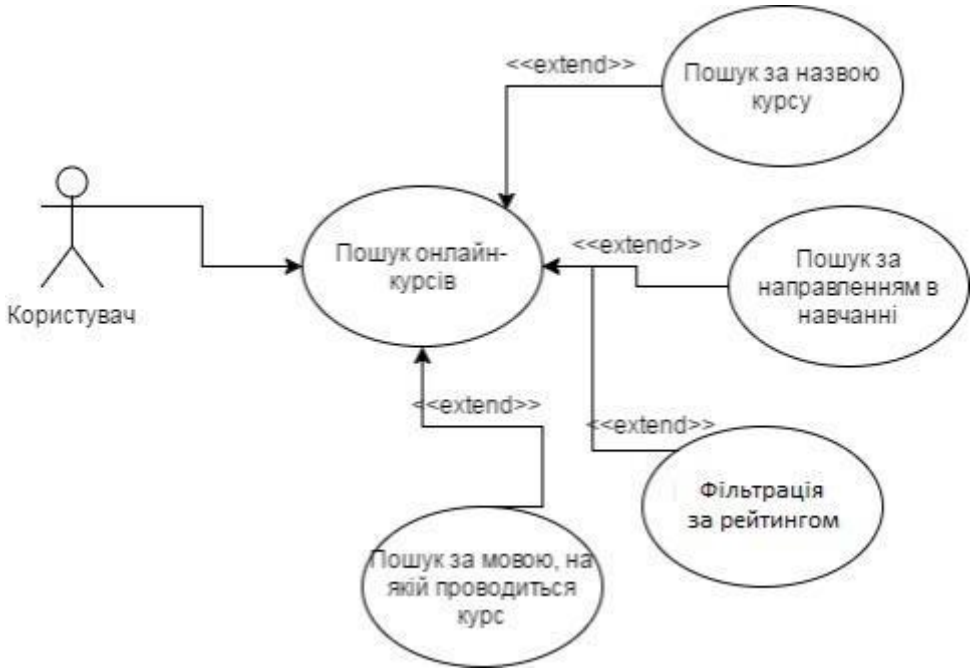


Рисунок 3.3 — Діаграма прецедентів

3.8 UML Діаграми класів

На рис. 3.4 представлена діаграма класів Android-додатку. На ній зображено класи, що відповідають за завантаження даних. CoursesObserverAPI, CourseSearchParameters, APIHelper — робота з CoursesObserver REST API; CoursesDB, DBHelper — робота з локальною БД; DataLoader, BasicDataLoader — об’єднує роботу з завантаженими даними в єдиний інтерфейс.

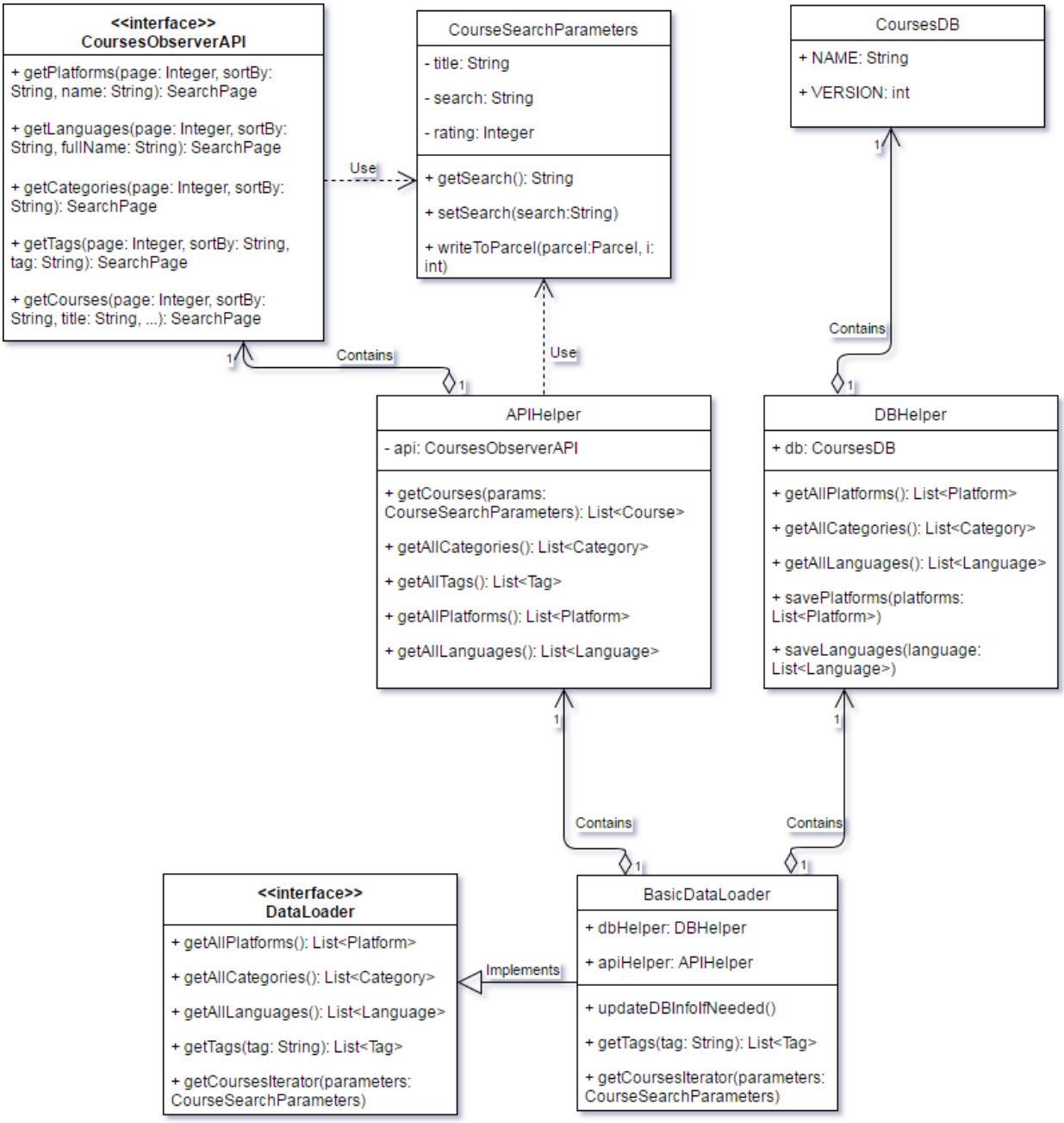


Рисунок 3.4 — Діаграма класів мобільного додатку

3.9 ERD діаграма БД веб-сервісу

ERD діаграма бази даних сервісу, що представлена на рисунку 3.5, описує зв'язок між таблицями, що зберігаються в БД. В ній зберігаються такі об’єкти як:

- Platform — платформа (Coursera, Udacity, etc.)
- Language — мова (en, ru, uk, etc.)
- Category — категорія курсу (Computer Science, Languages, etc.)
- Tag — теги курсів (android, python, etc.)
- Course — курс.

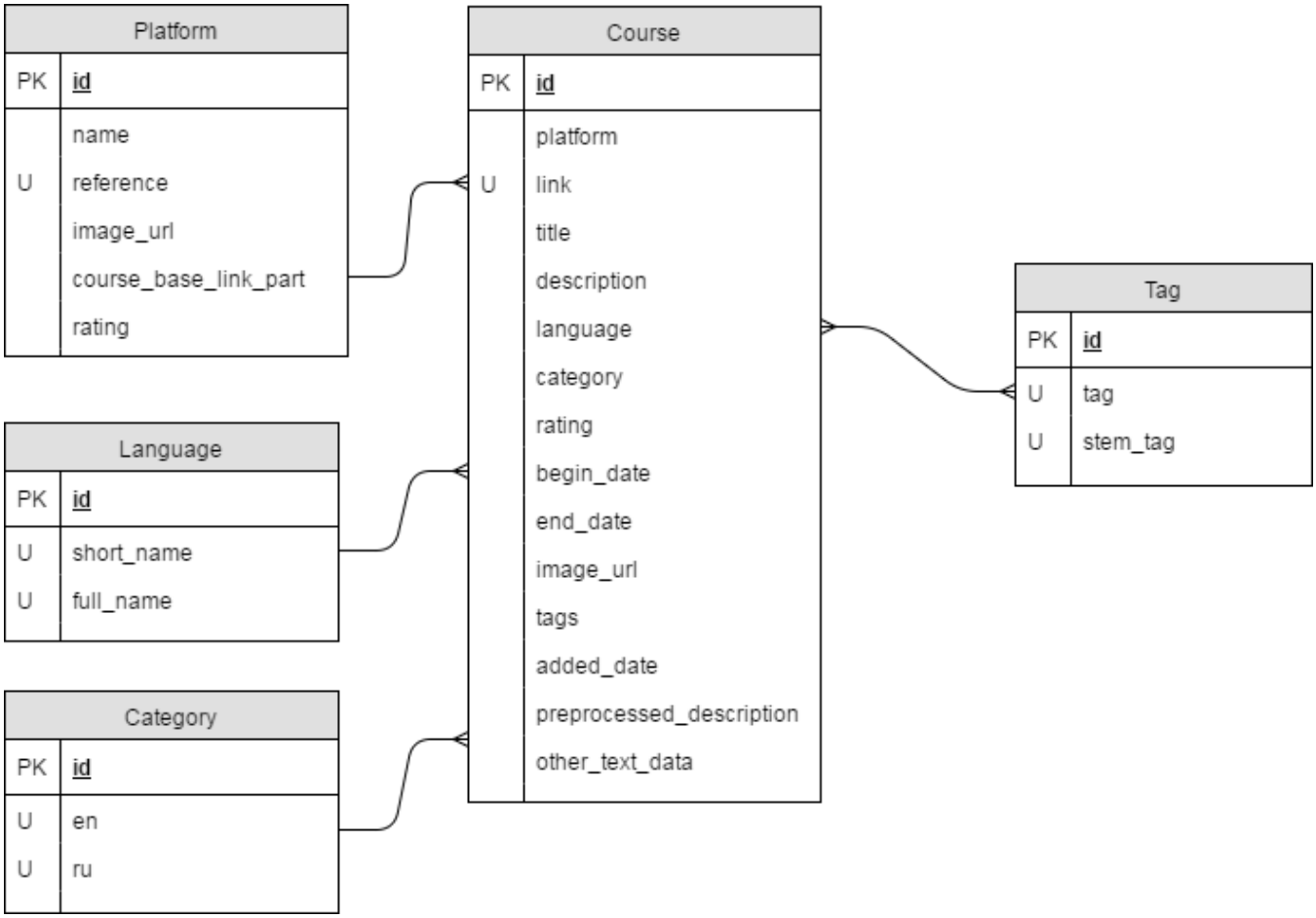


Рисунок 3.5 — ERD діаграма БД

3.10 Висновки

В результаті роботи був проведений порівняльний аналіз та був обраний алгоритм класифікації, що давав найбільшу кількість вірних результатів. Усі курси ДН, інформацію про які було зібрано, були класифіковані за категоріями

та кожному з них були присвоєні теги. Архітектурна модель, що була спроектована, та UML діаграми класів та прецедентів, ERD діаграма бази даних, описують реалізовану систему та підводять к етапу її реалізації.

4 РОЗРОБКА ТА АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Етапи виконання роботи

Розробка програмного продукту проводилась поетапно. В цьому підрозділі описуються основні етапи роботи та час, що був на них затрачений.

Основні етапи:

1. Постановка задачі та аналіз ринку
    1. Постановка задачі
    2. Аналіз и створення списку існуючіх платформ для ДН
    3. Аналіз існуючих рішень поставленої задачі (агрегаторів)
    4. Вибір мінімальних вимог до продуктів реалізації
        - Вибір можливостей, які будуть підтримуватися в реалізації
        - Вибір платформ, які будуть підтримуватися
    5. Вибір інструментів розробки системи (СУБД, мови, бібліотеки, і т.д.)
2. Проектування системи
    1. Побудова архітектури системи
    2. Побудова діаграми прецедентів системи
    3. Створення логотипу і вибір імені для системи
3. Реалізація агрегатора даних про курси ДН
    1. Аналіз і вивчення API / способів скрапінга сайтів обраних платформ
    2. Реалізація агрегаторів даних з обраних платформ
    3. Об'єднання агрегаторів даних в єдиний інтерфейс
    4. Побудова ERD-діаграми бази даних веб-сервісу

5. Збір інформації про курси ДН і збереження їх в БД
6. Отримання тегів для курсів ДН
    - Алгоритм для отримання тегів
    - Передобробка даних для отримання тегів
    - Отримання тегів заданим алгоритмом
7. Класифікація курсів ДН за категоріями
    - Вибір категорій для класифікації курсів
    - Аналіз способів текстової класифікації
    - Створення навчальної вибірки для класифікації курсів за категоріями
    - Передобробка даних для класифікації за категоріями
    - Порівняння методів класифікації для передоброблених даних
    - Класифікація курсів за допомогою обраного алгоритму

4. Реалізація API веб-сервісу
    1. Створення дизайну API [6]
    2. Написання документації до API
    3. Написання веб-сервісу з REST API
    4. Модульне тестування веб-сервісу
    5. Функціональне тестування реалізованого API
5. Реалізація мобільного додатка
    1. Створення дизайну додатка [7]
    2. Побудова діаграми класів мобільного додатка
    3. Реалізація додатку
    4. Модульне тестування мобільного додатку
    5. Функціональне тестування мобільного додатка
6. Вбудовування результатів роботи в систему "Cloud"
    1. Аналіз можливостей dl-cloud.kpi.ua

2. Створення варіантів використання створеної системи в dl-cloud.kpi.ua
3. Тестування варіантів використання створеної системи в dl-cloud.kpi.ua

4.2 Основні поняття

4.2.1 REST та Restful

REST (Representational State Transfer - «передача стану уявлення») - архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгодженим набіром обмежень, що враховуються при проектуванні розподіленої гіпермедіа-системи. У певних випадках (інтернет-магазини, пошукові системи, інші системи, засновані на даних) це призводить до підвищення продуктивності і спрощення архітектури.

В мережі Інтернет виклик віддаленої процедури може являти собою звичайний HTTP-запит (зазвичай «GET» або «POST»; такий запит називають «REST-запит»), а необхідні дані передаються в якості параметрів запиту.

Для веб-служб, побудованих з урахуванням REST, застосовують термін «RESTful» [6].

4.2.2 JSON

JSON (JavaScript Object Notation) — текстовий формат обміну даними, що заснований на JavaScript.

Об’єкт, що описує людину, представлений у форматі JSON:

```
{
  “first_name”: “John”,
  “last_name”: “Doe”,
```

```
  “address”: {
    “street”: “Allen Street”,
    “city”: “New York”,
  }
}
```

4.2.3 ORM

ORM (Object-Relational Mapping — об'єктно-реляційне відображення) — це технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».

Іншими словами, це зручний спосіб програмно працювати з реляційною базою даних, (в більшості випадків) без необхідності використання SQL. Об'єктно-реляційне відображення — це відображення вигляду:

- таблиця <-> клас,
- рядок таблиці <-> об'єкт класу.

4.3 Обгрунтування вибраних платформ , мов реалізації та фреймворків

1) **REST** архітектура більш за все підходить для поставленої задачі, тому що спроектована для передачі стану системи. Зважаючи на це, сервіс буде надавати REST API.

2) **Python** — мова програмування, що була вибрана для написання сервісу. Вона дозволяє за мінімальний час реалізувати необхідний функціонал, зручний

для побудови серверних додатків. На Python також існує багато бібліотек аналізу даних, таких як *Sckit-learn* та *nltk*, тому агрегатор також був написан на ній.

4) **Django** — це Python фреймворк для реалізації сервера, простий в освоєнні та має вбудовану ORM та шаблоні затор [8].

5) **PostgreSQL —** багатофункціональна СУБД.

6) **Android** — найбільш популярна ОС для мобільних платформ (встановлена на 81.7% мобільних пристроїв у світі на момент написання документу [9]).

7) **Retrofit** — це зручний фреймворк для роботи с REST API на Android, дозволяє додавати нові методи API буквально в один рядок.

8) **Picasso** дозволяє завантажувати зображення в Android додаток без додаткових зусиль.

9) **DBFlow** — один з найбільш зручних та швидких ORM-фреймворків для Android.

4.4 Результати роботи

У результаті були отримані два основних програмних продукта — веб-сервіс та мобільний додаток.

Назва системи — “CoursesObserver”.

Логотип зображений на рис.4.1.

Рисунок 4.1 — Логотип розробленого продукту “CoursesObserver”

4.4.1 Веб-сервіс «CoursesObserver»

Веб-сервіс “CoursesObserver” заснований на технології REST, тобто він є Restful веб-сервісом.

Проектування API проводилось згідно REST API best practices [6]

Формат запиту: HTTP GET запит.

Формат відповіді: JSON.

На даний момент сервер розміщений за адресою http://77.47.204.127:16522/.

Документація знаходиться на головній сторінці (рис. 4.2)

Рисунок 4.2 — Сторінка з документацією до CoursesObserver API

Він реалізує:

- пошук курсів ДН (search=пошуковий запит)
- фільтрацію курсів ДН за:
    - платформою (platform=перелік платформ)
    - категорією (category=перелік категорій)
    - тегами (tag=перелік тегів)
    - мовами (language=перелік мов)
    - рейтингом (rating=мінімальний рейтинг)
- сортування за:
    - датою додавання інформації в систему (sort_by=added)
    - рейтингом (sort_by=rating)
    - назвою (sort_by=title)
- пагінацію відповіді (page=номер сторінки)

Наприклад, запит пошуку курсів за параметрами: платформи: Coursera та Udacity, сортувати за: назвою, сторінка 4 має вигляд:

http://77.47.204.127:16522/courses/?page=4&platform=Coursera,Udacity&sort_by=title

Інформація про курс ДН, що надається у відповіді:

- id — уникальний ідентифікатор курса ДН в системі,
- tags — список тегів до курсу ДН,
- title — назва курсу ДН,
- description — опис курсу ДН,
- rating — рейтинг курсу (від 1 до 5), якщо є, інакше — 0,
- category — назва категорії, до якої відноситься курс ДН,
- link — посилання на сторінку з курсом ДН,

- image_url — посилання на картинку курсу (якщо є),
- platform — назва платформи, на якій розміщено курс,
- language — основна мова курсу ДН,
- begin_date — дата запуску курсу (якщо є),
- end_date — дата закінчення курсу (якщо є),
- added — дата додавання інформації про курс в систему.

Приклад відповіді в якості об’єкту курсу ДН у форматі JSON:

{

- **platform**: "Udacity",
- **begin_date**: null,
- **rating**: 0,
- **end_date**: null,
- **link**: "https://classroom.udacity.com/courses/ud1012/",
- **language**: "en",
- **id**: 2954,
- **added**: "2017-06-07 00:52:05.904346+00:00",
- **category**: "Computer Science",
- **image_url**: "https://lh3.googleusercontent.com/eIOkiYRITNN1cfPmJqyHwTGvoCvZCVTBQyTkE0uMSpkqBOja6PWNIRyI_f_-fKpP9I5NA-RipTl8amKEgCbY=s0#w=1725&h=1060",
- **title**: "Introduction to Virtual Reality",

- **description**: "This course is designed for students who are new to virtual reality and want to learn about the principles of VR technology including optics, displays, stereopsis, tracking, and major hardware platforms. You don’t need any programming experience to get started. By the end of this course, you will have created and deployed a VR application. You will understand the physical principles of VR and you will use that knowledge to create a comfortable, high-performance VR application using Unity.",
- **tags**:

[

  - "hardware",
  - "vr",
  - "unity",
  - "tracking",
  - "virtually",
  - "deployments",
  - "virtual reality"

]

}

4.4.2 Мобільний додаток «CoursesObserver»

Мобільний додаток “CoursesObserver” розроблений з урахуванням принципів Material Design та повністю повторює можливості розглянутого вище веб-сервісу.

Вимоги користування: пристрій з ОС Android 4.4 або вище.

На рис. 4.3 та 4.4 зображений приклад відображення шуканого списку курсів для пристроїв різної ширини та/або положення екрану.

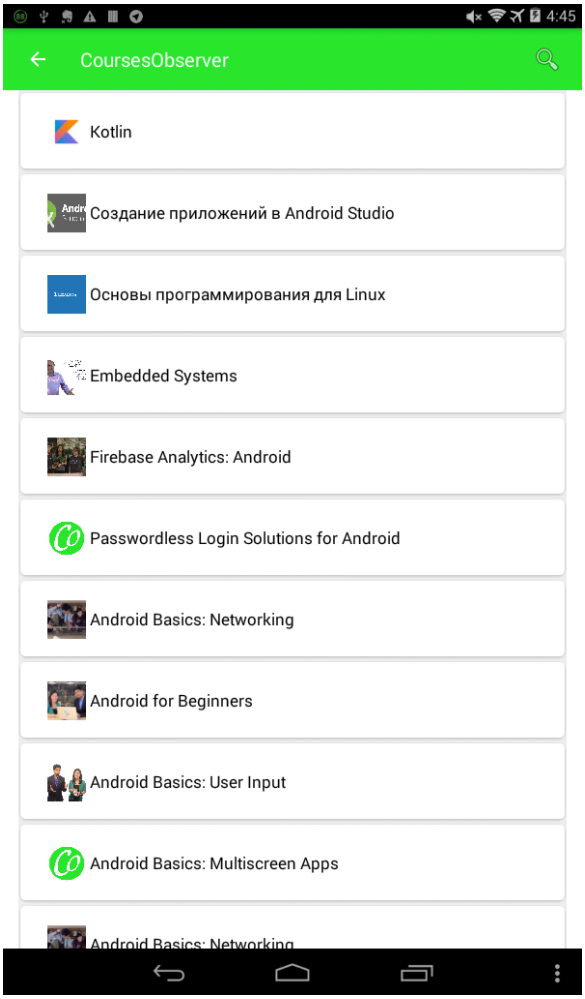


Рисунок 4.3 — Відображення списку курсів на планшеті у вертикальному положенні

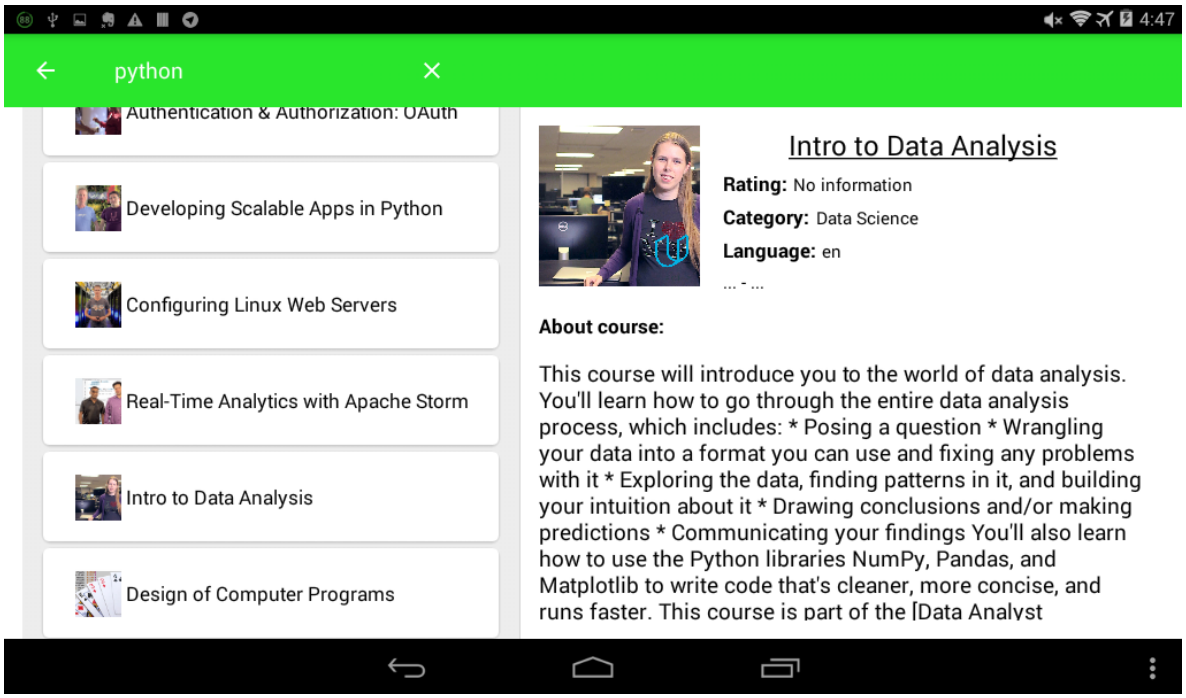


Рисунок 4.4 — Відображення списку курсів на планшеті у горизонтальному положенні

На рис. 4.5 зображений приклад пошук курсів про «machine learning».

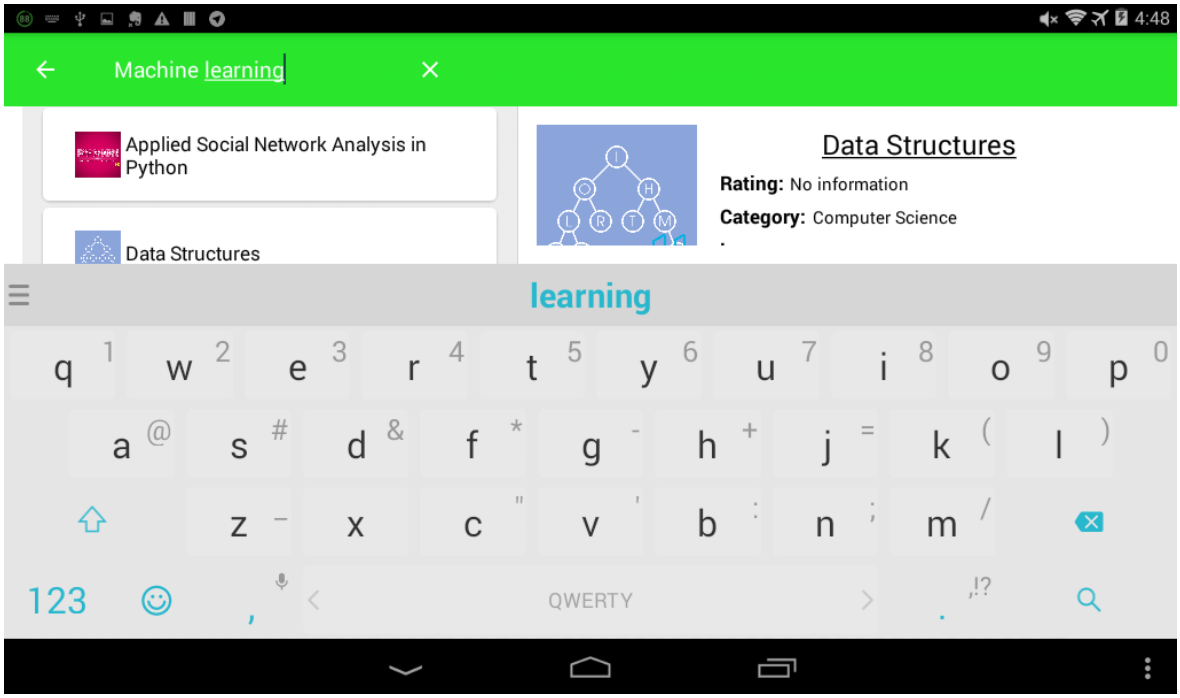


Рисунок 4.5 — Пошук курсів про «machine learning»

Таким чином, був розроблений Android-додаток з широким функціоналом для пошуку курсів ДН, з адаптивним та якісним дизайном.

4.5 Варіанти подальшого розвитку роботи

Подальший розвиток роботи складається як з розширення вже реалізованих можливостей, так і додавання нових:

- Додавання до системи нових платформ онлайн-освіти,
- Додавання нових параметрів курсів ДН,
- Додавання нових параметрів для фільтрації та сортування,
- Створення кабінату користувача,
- Зберігання пошукових запитів користувача та створення рекомендацій на їх основі,
- Завантаження та моніторинг курсів користувача, нотифікування про дедлайни,
- Відображення статистичних даних про користувача.

4.6 Висновки

Оцінивши час, наданий на реалізацію дипломної роботи, була проведена декомпозиція завдань та складений план роботи. В результаті отримали два програмних продукту, що готові до випуску:

- веб-сервіс «CoursesObserver».
- Android-додаток «CoursesObserver».

5 ФУНКЦІЙОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для пошуку курсів дистанційного навчання. Android-додаток був створений на мові програмування Java 1.7. Розробка проводилась у середовищі Android Studio.

Програмний продукт призначено для використання на мобільних пристроях та планшетних ПК під управлінням ОС Android 4.4+.

5.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на мобільних пристроях та планшетних ПК під управлінням ОС Android версії 4.4+;
- забезпечувати стабільну роботу, а при некоректних вхідних даних чи у разі виникненні помилок в процесі роботи програми надавати інформацію про їх походження;
- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;
- передбачати мінімальні витрати на впровадження програмного продукту.

5.2 Обґрунтування функцій програмного продукту

5.2.1 Формування варіантів функцій

Головна функція F0– розробка програмного продукту, який шукає курси дистанційного навчання за наданими параметрами. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – мова програмування;

F2 – середовище розробки;

F3 – спосіб зберігання даних, отриманих з веб-сервісу;

Кожна з основних функцій може мати декілька варіантів реалізації:

Функція F1:

a) Java;

b) Kotlin;

Функція F2:

a) Eclipse;

b) Android Studio;

Функція F3:

a) Не зберігати дані на мобільному пристрої;

b) Зберігати дані частково;

5.2.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 5.1).

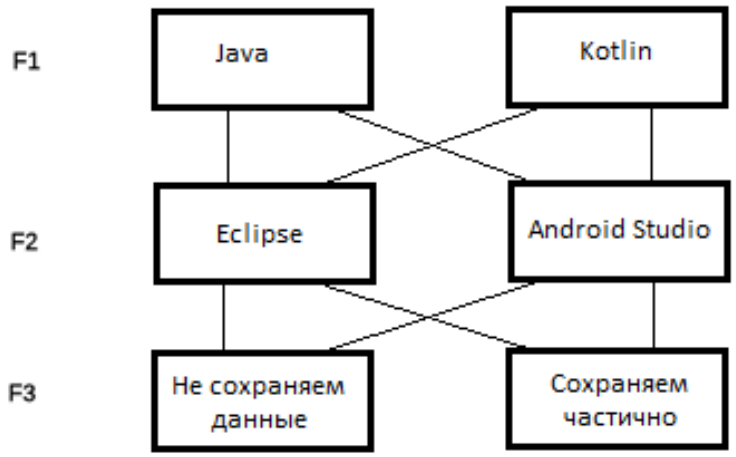


Рисунок 5.1 — Морфологічна карта системи

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 5.1 — Позитивно-негативна матриця варіантів основних функцій

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	a	Написана більша частина Android-додатків, документації	Менш зручна, ніж Kotlin, «зайвий» код
	b	Зручна у використанні	Мало спеціалістів
F2	a	Підлаштована спеціально для розробки під Android	Потребує багато ресурсів ЕОМ
	b	Підтримує багато різних мов	Необхідно підлаштовувати під вибрану мову
F3	a	Потребує менше пам’яті	Потребує більше часу на обробку запитів
	b	Потребує менше часу на обробку запитів	Потребує більше пам’яті

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, як такі, що вони не відповідають поставленим перед програмним продуктом технічним вимогам.

Функція F1.

Варіант b) можна відкинути, тому що це відносно нова мова, тому знайти спеціалістів буде складно, що потенційно означає зменшення надійності коду.

Функція F2.

Варіант a) можна відкинути зважаючи на те, що Android Studio більше підходить для поставленої задачі.

Функція F3

Варіанти a) та b) вважаємо гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1a – F2b – F3a

2.F1a – F2b – F3b

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.3 Обґрунтування системи параметрів ПП

Для характеристики розроблюваної програми можна використати такі параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті, потрібної для програми;

– *X3* – час обробки запитів користувача;

– *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2.

Таблиця 5.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Гб	16	8	4

Таблиця 5.2 (продовження)

1	2	3	4	5	6
Час обробки запитів користувача	X3	с	1000	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	3000	2500	2000

За даними таблиці 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

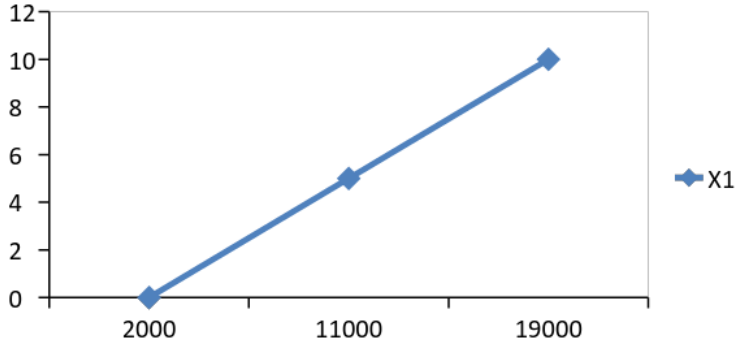


Рисунок 5.2 – X1, швидкодія мови програмування

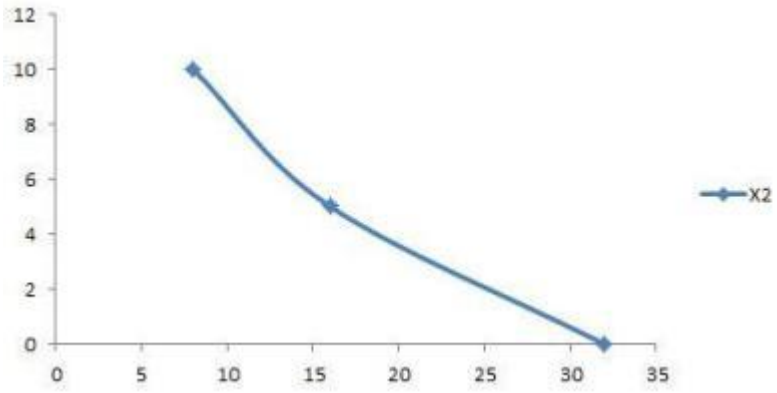


Рисунок 5.3 – X2, об’єм пам’яті для збереження даних

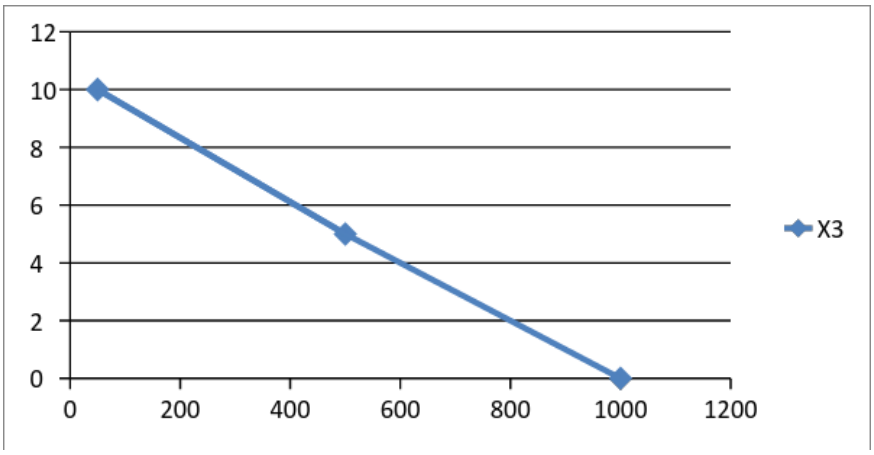


Рисунок 5.4 – X3, час виконання запитів користувача

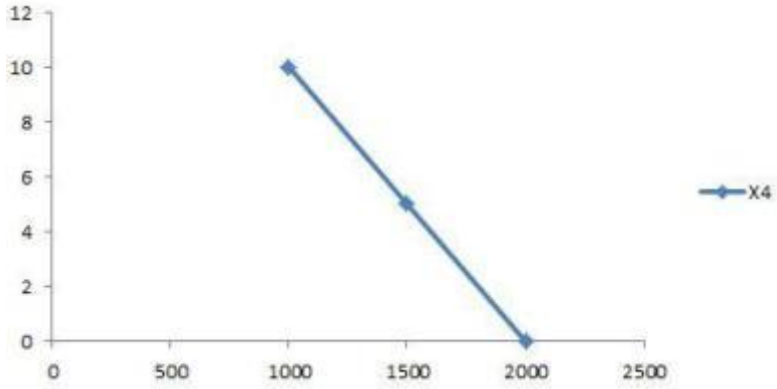


Рисунок 5.5 – X4, потенційний об’єм програмного коду

5.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень. Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

визначення рівня значимості параметра шляхом присвоєння різних рангів;

перевірку придатності експертних оцінок для подальшого використання;

визначення оцінки попарного пріоритету параметрів;

обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

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

$$R_i=\sum_{j=1}^{N} r_{ij}R_{ij}=\frac{Nn(n+1)}{2}=105, \qquad (5.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T=\frac{1}{n}R_{ij}=26,25 \qquad (5.2)$$

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

$$\Delta_i=R_i-T \qquad (5.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S=\sum_{i=1}^{N}\Delta_i^2=420,75 \qquad (5.4)$$

Порахуємо коефіцієнт узгодженості:

$$W=\frac{12S}{N^2(n^3-n)}=\frac{12\cdot 420,75}{7^2(5^3-5)}=1,03>W_k=0,67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5

Таблиця 5.4 (продовження)

1	2	3	4	5	6	7	8	9	10
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

З отриманих числових оцінок переваги складемо матрицю $A=\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

Kві=bii=1nbi,де bi=i=1Naij (5.6)

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

Kві=bi'i=1nbi',де bi'=i=1Naijbj (5.7)

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметрихі	Параметрихj				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	bi	Кві	bi1	Кві1	bi2	Кві2
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

5.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так:

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів; $K_{вi}$ – коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Результати розрахунків зведено в табл. 5.6.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	b	15000	7.5	0,208	1.56
F2	a	24	2.5	0,158	0.395
F3	a	500	4	0,361	1.444
	b	300	8	0,361	2.888

За даними з таблиці 5.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$K_{K1} = 1.56 + 0.395 + 1.444 = 3.40$

$K_{K2} = 1.56 + 0.395 + 2.888 = 4.84$

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт рівня якості має найбільше значення.

5.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Але варіант II реалізації програмного забезпечення включає ще одне завдання:
3. Написання алгоритму збереження інформації у вигляді компонента, зручного для візуалізації.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б, завдання 3 до групи Г. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Завдання 3 відноситься за складністю до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \qquad (5.10)$$

де T_P – трудомісткість розробки ПП; $K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид

нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Для третого завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

$Т_р$ = 8 людино-днів;

$К_п$ = 0,6; $К_{ст}$ = 1;

$Т_3 = 8 \bullet 0,6 \bullet 1 = 4,8$.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I = (122.4 + 19.44) \cdot 8 = 1134.72$ людино-годин;

$Т_{II} = (122.4 + 19.44 + 4.8) \cdot 8 = 1173.12$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці бере участь один спеціаліст з мобільної розробки з окладом 15 765 грн. Визначимо зарплату за годину за формулою:

$$C_ч = M/(Tm \cdot t) грн., \quad (5.11)$$

де M – місячний оклад працівників; Tm – кількість робочих днів тиждень; t – кількість робочих годин в день.

$C_ч = 15\ 765/(21 \cdot 8) = 93.84$ грн.

Тоді, розрахуємо заробітну плату за формулою

$$C_{ЗП} = Сч \cdot Ti \cdot K_Д, \quad (5.12)$$

де $C_ч$ – величина погодинної оплати праці програміста; Ti – трудомісткість відповідного завдання; $K_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробника за варіантами становить:

I. $C_{ЗП} = 93.84 \cdot 1134.72 \cdot 1.2 = 127778.55$ грн.

II. $C_{ЗП} = 93.84 \cdot 1173.12 \cdot 1.2 = 132102.70$ грн.

Відрахування на єдиний соціальний внесок становить 22% на 01.05.2017. Отримуємо такі значення:

I. $C_{ВІД} = C_{ЗП} \cdot 0.22 = 56372.89 \cdot 0.22 = 28111.28$ грн.

II. $C_{ВІД} = C_{ЗП} \cdot 0.22 = 58208.60 \cdot 0.22 = 29062.59$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($C_М$)

Так як одна ЕОМ обслуговує одного аналітика з окладом 15 765 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 15\ 765 \cdot 0{,}2 = 37836$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 37836 * (1 + 0.2) = 45403.2$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 45403.2 \cdot 0{,}22 = 9988.70$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 18 888 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 18\ 888 = 5430.3$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 18\ 888 \cdot 0.05 = 1086.06$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 105 – 11 – 16) \cdot 8 \cdot 0.9 = 1677.6$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t – кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677.6 \cdot 0.156 \cdot 1.94 = 507.71$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 18\ 888 \cdot 0.67 = 12654.96$ грн.

Тоді, річні експлуатаційні витрати будуть:

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$

$С_{ЕКС} = 45403.2 + 9988.70 + 5430.3 + 1086.06 + 507.71 + 12654.96 = 75070.93$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 75070.93 / 1706.4 = 43.99$ грн/год.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М\text{-}Г} \cdot Т$

I. $С_М = 43.99 \cdot 1134.72 = 49916.33$ грн.;

II. $С_М = 43.99 \cdot 1173.12 = 51605.55$ грн.;

Накладні витрати складають 67% від заробітної плати:

$С_Н = С_{ЗП} \cdot 0{,}67$

I. $С_Н = 127778.55 \cdot 0.67 = 85611.63$ грн.;

II. $С_Н = 132102.70 \cdot 0.67 = 88508.81$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}$

I. $С_{ПП} = 127778.55 + 28111.28 + 49916.33 + 85611.63 = 291417.79$ грн.;

II. $С_{ПП} = 132102.70 + 29062.59 + 51605.55 + 88508.81 = 301279.65$ грн.;

5.7 Висновки. Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$К_{ТЕРj} = К_{Кj} / С_{Фj}$,

$К_{ТЕР1} = 3.4 / 291417.79 = 1.17 * 10 ^-5$;

$К_{ТЕР2} = 4.84 / 301279.65 = 1.61 * 10^-5$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР2} = 1.61*10^-5$.

ВИСНОВКИ

У даній роботі була реалізована система збору інформації про курси дистанційного навчання. Особливий акцент робився на реалізації мобільного додатка, який би надавав можливості пошуку курсів за параметрами в зручному для користувача вигляді.

Після того, як основна задача була визначена, був проведений пошук і порівняльний аналіз готових рішень — існуючих МВОК-агрегаторів. Визначивши те, чого в них не вистачає, і вимоги до реалізації, була поставлена мета реалізувати систему, яка б задовольняла цим вимогам.

Після постановки вимог були обрані інструменти для реалізації. Так, мовою для написання агрегатора і веб API став Python, а мобільний додаток було вирішено реалізовувати на ОС Android. Крім цього були обрані СУБД, фреймворки і бібліотеки, які б допомогли у вирішенні поставленого завдання.

Наступним етапом стало проектування системи. Були построєні діаграми архітектури системи, прецедентів, ARIS та DFD. Був створений логотип системи та вибрана назва для неї — «CoursesObserver».

Після цього починається реалізації системи. Для деяких МВОК-платформ, таких як Udacity та Stepik, існує веб-API [10], що дозволяє отримувати дані про курси дистанційного навчання, що розташовані на них. Для інших платформ інформація може бути отримана за допомогою скрапінгу веб-сторінок сайтів цих платформ. Обидва підходи використовується у розробленому агрегаторі. Вся отримана інформація про курси дистанційного навчання зберігається в базі даних та оновлюється два рази на добу.

Так як на різних платформах різна категоризація курсів дистанційного навчання, необхідно було вибрати одну та класифікувати їх за допомогою

алгоритму класифікації. Після вибору категорій була створена навчальна та тестові вибірки для алгоритму. Усі текстові дані, що були зібрані про курс, піддалися передобробці — були переведені англійською, розбиті на слова, які потім були приведені до їх основ. Для вибору найкращого алгоритму класифікації проводився порівняльний аналіз між 8 алгоритмами класифікаціями (Наївний Байес,SVM, нейронні мережі та інші). Кращим виявився алгоритм SVM, що використовує SGD. За допомогою якого і були прокатегоризовані курси.

Задача вибору тегів для курсів ДН теж була поставлена. Передобробка інформації проводилась так само як і для класифікації, потім, за tf-idf були вибрані найважливіші слова і з них вручну відібрані теги.

Наступний крок — розробка REST API для доступу до отриманої інформації. Спочатку спроектований, потім реалізований і протестований веб-сервіс на даний момент розташовується за адресом http://77.47.204.127:16522/ з документацією до користування на головній сторінці.

Після розробки веб-сервісу необхідно було придумати варіанти його використання для dl-cloud.kpi.ua. Такі варіанти, як автоматичне генерування статей та рекомендацій, були придумані та протестовані на можливість додавання.

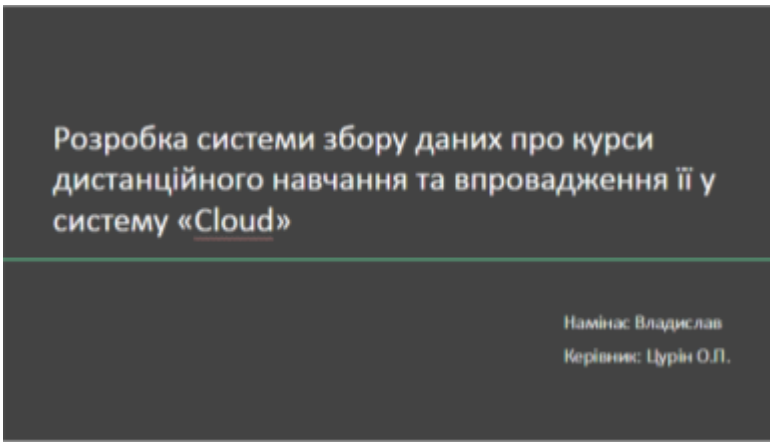
І, в решті-решт, розробка мобільного додатку на ОС Android. Спочатку створення дизайну, побудова діаграм, написання коду додатку, його модульне та функціональне тестування.

Таким чином, всі поставлені задачі були виконані в певному об’ємі та реалізовані у вигляді продуктів програмного забезпечення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Article “An Early Report Card on Massive Open Online Courses” / Geoffrey A. Fowler // The Wall Street Journal // 08.10.2013 — Режим доступу: https://www.wsj.com/articles/an-early-report-card-on-massive-open-online-courses-1381266504?tesla=y .
2. Кристофер Д. Маннинг. Введение в информационный поиск / Кристофер Д. Маннинг, Прабхакар Рагхавар, Хайнрих Шютце ;[Пер.с англ. — М.: ООО «И.Д. Вильямс»] — Москва - Санкт-Петербург - Киев: «И.Д. Вильямс», 2011 — 512 с.
3. Alex Smola. Introductino to Machine Learning / Alex Smola, S.V.N. Vishwanathan — UK: Cambridge University Press, 2008 — 234 с.
4. Steven Bird. Natural Language Processing with Python [Електронний ресурс] / Steven Bird, Ewan Klein, Edward Loper // — Режим доступу: http://www.nltk.org/book/ .
5. Документація бібліотеки машинного навчання на Python Scikit-learn [Електронний ресурс] — Режим доступу: http://scikit-learn.org/stable/ — Дата доступу: 05.06.2017.
6. Mark Masse. REST API Design Rulebook — O’Reilly Media, 2011 — 116 с.
7. Ian G. Clifton. Android User Interface Design (Second Edition) — Crawfordsville, Indiana, USA: RR Donnelley, 2015 — 448 с.
8. Antonio Melé. Django by Example — Packt Publishing, 2015 — 474 с.
9. Article “99.6 percent of new smartphones run Android or iOS” / James Vincent // The Verge // 16.02.2017 — Режим доступу: https://www.theverge.com/2017/2/16/14634656/android-ios-market-share-blackberry-2016 .
10. Stepik REST API: — Режим доступу: https://stepik.org/api/docs/ — Дата доступу: 05.06.2017 — Назва з екрану.

ДОДАТОК А. ІЛЮСТРАТИВНИЙ МАТЕРІАЛ



Постановка задачі

Основні задачі розробки:

- Створення MOOK-агрегатору та REST API для нього.
- Створення мобільного додатку, що надає можливості пошуку курсів дистанційного навчання.

Особливий акцент робиться на розробку зручного та функціонального мобільного додатку, як такого, що задовольняв би потребам користувача в пошуку курсів.

Існуючі рішення

Mooc-list — найкраще з існуючих на даний момент рішень.

Він має широкі можливості пошуку:

- багато різноманітних параметрів фільтрації
- сортування за датою, рейтингом, назвою
- можливість зберігання вибраних курсів

Існуючі рішення

Але, незважаючи на широкі можливості пошуку, інтерфейс веб-сайту не відповідає вимогам зручного. Mooc-list також має свій Android-додаток, але єдина можливість, що він надає — вибір курсів дистанційного навчання за категоріями, що ніяк не можна назвати рішенням поставленої задачі до мобільного додатку.

Архітектура системи, що розроблялась

Агрегація інформації про курси дистанційного навчання з різних платформ

- Різний підхід до різних платформ.
- Основні способи отримання інформації:
  - Використовуючи API, що надає платформа для розробників.
  - Використовуючи інструменти веб-скрапінгу (парсингу).
  - Комбінуючи ці підходи.
- Основні інструменти, що були використані для цієї задачі:
  - мова програмування – Python 3.5
  - бібліотека "динамічного" парсингу – selenium + phantomjs
  - бібліотека "статичного" парсингу – beautifulsoup

Пошук тегів для курсів

Алгоритм пошуку тегів:

1. Позбавляємось розділових знаків
2. Перекладаємо англійською
3. Токенізація
4. Позбавляємося стоп-слів
5. Отримаємо tf-idf значення для усіх даних
6. Зберігаємо для кожного тексту тільки ті фрази, що мають найбільший коефіцієнт tf-idf
7. Вручну позбавляємось "зайвих" фраз

Класифікація курсів за категорією

Класифікація курсів проводиться за наступним алгоритмом:

1. Позбавляємось розділових знаків.
2. Перекладаємо англійською.
3. Токенізація.
4. Позбавляємося стоп-слів.
5. Стемінг.
6. Отримаємо Tf-Idf значення для навчальної вибірки
7. Класифікуємо за допомогою алгоритму "Наївного Баєса"

Результати: веб-сервіс "CoursesObsever"

- Restful веб-сервіс
- Формат запиту: HTTP GET запит
- Формат відповіді: JSON
- Документація: http://77.47.204.127:16522/

Результати: веб-сервіс "CoursesObsever"

Запит пошуку курсів за параметрами:

- платформи: Coursera та Udacity.
- сортувати за: назвою.

http://77.47.204.127:16522/courses/?page=4&platform=Coursera,Udacity&sort_by=title

Результати: мобільний додаток “CoursesObserver”

Мобільний додаток “CoursesObserver” розроблен з урахуванням принципів Material Design та повністю повторює можливості розглянутого вище веб-сервісу.

Вимоги користування: пристрій з ОС Android 4.4 або вище.

Використання у системі “Cloud”

- Автоматичне генерування статей про нові курси дистанційного навчання та за категорією
- Можливість генерувати та відправляти рекомендації зареєстрованим користувачам
- Можливість надавати пошук у вигляді веб-інтерфейсу користувача.

Висновки

- Проаналізовані платформи дистанційного навчання (Udacity, Coursera, Stepik, etc.), існуючі MOOK-агрегатори (Mooc-list, Mooctivity, etc.).
- Розроблена програма агрегації даних з популярних платформ дистанційного навчання, їх категоризування та пошуку тегів.
- Розроблен веб-сервіс, що надає REST API до агрегованих даних з платформ дистанційного навчання
- Розроблен Android-додаток, що є клієнтом описаного веб-сервісу.
- Надані та протестовані варіанти для використання розробленого API у системі “Cloud”.

Варіанти подальшого розвитку роботи

- Додавання до системи нових платформ
- Додавання нових параметрів курсів
- Фільтрація та сортування за новими параметрами
- Створення кабінету користувача
- Зберігання пошукових запитів користувача та створення рекомендацій на їх основі
- Завантаження та моніторинг курсів користувача, нотифікування про дедлайни

ДОДАТОК Б. ПРОВАЙДЕРИ КУРСІВ

У таблиці 1 представлений список провайдерів курсів. Також для декількох показано, чи надають вони api, чи можна зпарсити їх статично та в якому форматі надаються курси на сайті.

Таблиця 1 — Провайдери курсів

Сайт	api	static	Формат
www.coursera.org	+	+	MOOC / відеолекції
www.edx.org	+	-	MOOC
www.udacity.com	+	+	MOOC
www.udemy.com	+	-	MOOC
www.stepic.org	+	-	MOOC
www.prometheus.org.ua		+	MOOC
www.iversity.org	+	+	MOOC
www.futurelearn.com		+	MOOC
www.ru.hexlet.io			MOOC

www.geekbrains.ru			MOOC / відеолекції
https://teamtreehouse.com			MOOC
http://www.sololearn.com/			MOOC
http://courses.caveofprogramming.com/			
https://www.techrocket.com/			
https://www.lynda.com/			
http://www.edureka.co/			
https://prince2.co/			
https://www.eduonix.com			
http://www.simplilearn.com/			
https://mva.microsoft.com/			
http://www.mastergradeit.co.za/			
https://novoed.com/			
https://www.khanacademy.org/			

http://www.saylor.org/			
http://learno.net/			
https://www.codeschool.com/			
http://www.w3schools.com/			tutorials
https://www.open2study.com/			
https://www.canvas.net/			
https://www.openlearning.com/			
https://openeducation.blackboard.com/			
https://compscicenter.ru/			відеолекції
https://yandexdataschool.ru/			відеолекції
http://tutsplus.com/			
https://www.kadenze.com			
https://www.thinkful.com/			
https://textilelearning.com			

https://www.pluralsight.com/			
http://www.thegreatcourses.com/			
https://www.techchange.org/			
http://ocw.mit.edu/			
https://courses.platzi.com/			
http://www.tutorialspoint.com/			tutorials
http://eclass.cc/			
https://www.courson.ru			
https://www.sophia.org/			
https://smart.ly/			
https://www.skillshare.com/			
https://www.goskills.com/			
http://learn.filtered.com/			
https://www.datacamp.com/			

http://www.360training.com/			
https://onemonth.com/			
https://open.hpi.de			
https://alison.com/			
https://codermanual.com/			
http://courses.eazl.co/			
http://www.ed2go.com/			
https://www.edcast.org/			
http://coggno.com/			

ДОДАТОК В. ЛІСТИНГ РОЗРОБЛЕНИХ ПРОГРАМ

“platforms/utils.py”

```
from bs4 import BeautifulSoup

import requests

import re

from nltk.tokenize import word_tokenize

from nltk.stem import PorterStemmer

from sklearn.feature_extraction.text import TfidfVectorizer

import numpy as np

from sklearn.naive_bayes import MultinomialNB

from sklearn.linear_model import SGDClassifier

from sklearn.neural_network import MLPClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier



def get_text_from_html(html):

    bs = BeautifulSoup(html, "html.parser")

    return bs.get_text("\n")



def translate_words_to_en(words):

    translated = []

    for word in words:
```

```
        params = {
            "to": "en",
            "text": word
        }
        response = requests.post("http://www.transltr.org/api/translate", data=params, timeout=10000).json()
        if "translationText" not in response:
            translated.append(word)
        else:
            translated.append(response["translationText"])

    return translated


punctuation_removing_pattern = re.compile(u'[^\w ]', re.UNICODE)


def remove_punctuation(text):
    return punctuation_removing_pattern.sub(' ', text).strip().replace("  ", " ")


removing_non_en_words_pattern = re.compile(u"[^a-zA-Z0-9' ]")


def remove_non_en(text):
    return removing_non_en_words_pattern.sub('', text).strip()
```

```
with open('D:\\Python\\CoursesObserver\\data\\terrier-stop.txt') as f:
    stop_words = [i.strip() for i in f.readlines()]



def text_preprocessing_with_translation(text, translate=True):
    stemmer = PorterStemmer()
    text = text.lower()
    text_words = remove_punctuation(text).split(" ")
    if translate:
        eng_text_words = translate_words_to_en(text_words)
    else:
        eng_text_words = text_words
    eng_text_words = [w for w in eng_text_words if w]
    eng_text = remove_non_en(" ".join(eng_text_words)).lower()
    eng_words_tokens = word_tokenize(eng_text)
    filtered_eng_words = list(filter(lambda w: w not in stop_words, eng_words_tokens))
    filtered_stems = list(map(lambda word: stemmer.stem(word), filtered_eng_words))

    return filtered_eng_words, filtered_stems



def find_keywords(texts):
    vectorizer = TfidfVectorizer(ngram_range=(1, 2))
    train_tfidf = vectorizer.fit_transform(texts)
    feature_names = np.array(vectorizer.get_feature_names())  # все слова

    keywords = set()
```

```
    k = 0
    for i in range(0, len(texts)):
        text_phrases = train_tfidf.getrow(i).todense().tolist()[0]  # получаем оценку tf_idf для i-того текста
        text_notnull_phrases = [pair for pair in zip(range(0, len(text_phrases)), text_phrases) if
                                pair[1] > 0]  # оставляем только ненулевые элементы
        sorted_text_notnull_phrases = sorted(text_notnull_phrases, key=lambda x: x[1] * -1)
        for phrase, score in [(feature_names[word_id], score)
                              for (word_id, score) in
                              sorted_text_notnull_phrases[0:min(5, len(sorted_text_notnull_phrases))]]:
            keywords.add(phrase)
        k += 1
        print(k)

    return keywords


def get_courses_categories(categorized_courses_texts, courses_categories, not_categorized_courses_texts,
                           classifier=MultinomialNB()):
    vectorizer = TfidfVectorizer(ngram_range=(1, 2))
    train_tfidf = vectorizer.fit_transform(categorized_courses_texts)
    feature_names = np.array(vectorizer.get_feature_names())  # все слова

    clf = classifier.fit(train_tfidf, courses_categories)
    new_tfidf = vectorizer.transform(not_categorized_courses_texts)
    predicted = clf.predict(new_tfidf)
    return predicted
```

```
def stemming(words):
    stemmer = PorterStemmer()
    stems = list(map(lambda word: stemmer.stem(word), words))
    return stems
```

“platform.stepik.py”

```
import requests
import re
import datetime


from observer.models import Course
from observer.models import Language
from observer.models import Platform



class Stepik:
    PLATFORM_NAME = "Stepik"
    platform = None
    datetime_pattern = "([0-9]{4})-([0-9]{2})-([0-9]{2})T([0-9]{2}):([0-9]{2}):([0-9]{2})Z"


    # TODO: add logs/listener for progress
    # TODO: format date
    @staticmethod
    def get_courses(**kwargs):
        if not Stepik.platform:
            Stepik.platform = Platform.objects.get(name=Stepik.PLATFORM_NAME)
```

```
    min_learners_count = 30
    # если есть дата, с которой нужно просматривать инфу
    if "from_date" in kwargs:
        return []
    # иначе всю инфу
    else:
        courses = []
        page_number = 1
        while True:
            page = requests.get("https://stepik.org/api/courses?page={}".format(page_number)).json()
            json_courses = page["courses"]
            for json_course in json_courses:
                # добавляем, если он публичный, если достаточное количество учеников и есть картинка
                if json_course["is_public"] and json_course["learners_count"] >= min_learners_count \
                        and json_course["cover"]:
                    courses.append(Stepik.from_json(json_course))
            if not page["meta"]["has_next"]:
                break
            page_number += 1

        return courses

@staticmethod
def from_json(json_course):
    rating = int(
        requests.get("https://stepik.org/api/course-review-summaries/" + str(json_course["review_summary"]))
            .json()["course-review-summaries"][0]["average"]) if "review_summary" in json_course else 0
```

```
language = json_course["language"] if "language" in json_course else None
if language:
    try:
        language = Language.objects.get(short_name=language[0:2])
    except Language.DoesNotExist:
        language = None

begin_date = json_course["begin_date"] if "begin_date" in json_course else None
end_date = json_course["last_deadline"] if "last_deadline" in json_course else None
if begin_date:
    begin_date = Stepik.get_datetime_from_str(begin_date)

if end_date:
    end_date = Stepik.get_datetime_from_str(end_date)

course = Course(platform=Stepik.platform, title=json_course["title"] if "title" in json_course else None,
                description=json_course["description"] if "description" in json_course else None,
                language=language,
                rating=rating,
                link=str(json_course["id"]) + "/",
                begin_date=begin_date,
                end_date=end_date,
                image_url=(
                    "https://stepik.org" + json_course["cover"]) if ("cover" in json_course) and (
                json_course["cover"]) else None
                )
```

```
        return course


    @staticmethod
    def get_datetime_from_str(str):
        m = re.search(pattern=Stepik.datetime_pattern, string=str)
        if m:
            year = m.group(1)
            month = m.group(2)
            day = m.group(3)
            hour = m.group(4)
            minute = m.group(5)
            second = m.group(6)


            return datetime.datetime(int(year), int(month), int(day), int(hour), int(minute), int(second))
        else:
            return None
```

“observer.views.py”

```
from django.shortcuts import render


from django.http import HttpResponse
import json


from .models import *
from .utils import PAGE_SIZE
from .utils import paginate
from .query_params import *
```

```
from platforms.utils import stemming



#stepik example
# # print(request.GET.get('code', ''))
# # 1. Get your keys at https://stepik.org/oauth2/applications/
# # (client type = confidential, authorization grant type = client credentials)
# client_id = "N5Zhxj6JjTqC1jEvVYLomlZc9PdnwOLmBFn8dxI1"
# client_secret = "FUOI07n7PxZpOd0JRtdrrgjgZNNBy7tTvHmyFon8Z48rpRckLmnXErSlqL5svlrx37dquUDQeb6MyLpMxq4NHVtT7OPQnhitNzryeDb79AfwpZFTrQesBu8gyVpvgIBs"
# # my_code = "dZYL1EprM2WhesTcqsWAgB1qmq2x6y"
#
# # 2. Get a token
# auth = requests.auth.HTTPBasicAuth(client_id, client_secret)
# resp = requests.post('https://stepik.org/oauth2/token/',
#                      data={'grant_type': 'authorization_code', 'redirect_uri': 'http://77.47.204.127:16522/',
#                          'code': request.GET.get('code', '')},
#                      auth=auth,
#                      )
#
# print(resp.json())
# token = resp.json()['access_token']
#
# api_url = 'https://stepik.org/api/stepics/1'
# resp = json.loads(requests.get(api_url, headers={'Authorization': 'Bearer ' + token}).text)
```

```
#
# user = resp['users']
# name = user[0]['first_name'] + ' ' + user[0]['last_name']
#
# print(name)
# return HttpResponse("Hello, world. It is Stepik API request")



# TODO: make API documentation page



def docs(request):
    context = {"api_endpoints": API_ENDPOINTS}
    return render(request, 'observer/docs/docs.html', context)



def platforms(request, pk=None):
    if pk:
        try:
            platform = Platform.objects.get(pk=pk)
        except Platform.DoesNotExist:
            platform = None
        return HttpResponse(paginate([platform] if platform else [], "platforms", 1, False),
                            content_type=CONTENT_TYPE_JSON)
    else:
        try:
            page_number = int(request.GET.get("page", 1))
```

```
    assert page_number >= 1
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid page parameter value"}))
from_obj = (page_number - 1) * PAGE_SIZE
to_obj = page_number * PAGE_SIZE

result_query = Platform.objects

# filtering
filter_by_name = request.GET.get(platforms_filter_by_name, None)
if filter_by_name:
    result_query = result_query.filter(name__in=filter_by_name.split(","))

# sorting
try:
    sort_by = request.GET.get(SORT_BY_PARAM, None)
    assert (sort_by == platforms_filter_by_name) or (sort_by == platforms_sort_by_rating) or not sort_by
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid sort_by parameter value"}))

if sort_by == platforms_sort_by_name:
    result_query = result_query.order_by("name")
elif sort_by == platforms_sort_by_rating:
    result_query = result_query.order_by("-rating")

result_query = result_query.all()
```

```
        try:
            next_obj = Platform.objects.all()[to_obj]
            has_next = True
        except IndexError:
            has_next = False

        return HttpResponse(paginate(result_query[from_obj: to_obj], "platforms",
                                     page_number=page_number, has_next=has_next), content_type=CONTENT_TYPE_JSON)


def languages(request, pk=None):
    if pk:
        try:
            language = Language.objects.get(pk=pk)
        except Language.DoesNotExist:
            language = None
        return HttpResponse(paginate([language] if language else [], "languages", 1, False),
                            content_type=CONTENT_TYPE_JSON)
    else:
        try:
            page_number = int(request.GET.get("page", 1))
            assert page_number >= 1
        except (ValueError, AssertionError) as e:
            return HttpResponse(json.dumps({"Error": "Invalid page parameter value"}))
        from_obj = (page_number - 1) * PAGE_SIZE
        to_obj = page_number * PAGE_SIZE
```

```
result_query = Language.objects


# filtering
filter_by_full_name = request.GET.get(languages_filter_by_full_name, None)
if filter_by_full_name:
    result_query = result_query.filter(full_name__in=filter_by_full_name.split(","))


# sorting
try:
    sort_by = request.GET.get(SORT_BY_PARAM, None)
    assert (sort_by == languages_sort_by_full_name) or (sort_by == languages_sort_by_short_name) or not sort_by
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid sort_by parameter value"}))


if sort_by == languages_sort_by_short_name:
    result_query = result_query.order_by("short_name")
elif sort_by == languages_sort_by_full_name:
    result_query = result_query.order_by("full_name")


result_query = result_query.all()


try:
    next_obj = Language.objects.all()[to_obj]
    has_next = True
except IndexError:
    has_next = False
```

```
        return HttpResponse(paginate(result_query[from_obj: to_obj], "languages",
                                     page_number=page_number, has_next=has_next), content_type=CONTENT_TYPE_JSON)



def categories(request, pk=None):
    if pk:
        try:
            category = Category.objects.get(pk=pk)
        except Category.DoesNotExist:
            category = None
        return HttpResponse(paginate([category] if category else [], "categories", 1, False),
                            content_type=CONTENT_TYPE_JSON)
    else:
        try:
            page_number = int(request.GET.get("page", 1))
            assert page_number >= 1
        except (ValueError, AssertionError) as e:
            return HttpResponse(json.dumps({"Error": "Invalid page parameter value"}))
        from_obj = (page_number - 1) * PAGE_SIZE
        to_obj = page_number * PAGE_SIZE


        result_query = Category.objects


        # sorting
        try:
            sort_by = request.GET.get(SORT_BY_PARAM, None)
            assert (sort_by == categories_sort_by_en) or not sort_by
```

```
        except (ValueError, AssertionError) as e:
            return HttpResponse(json.dumps({"Error": "Invalid sort_by parameter value"}))

        if sort_by == categories_sort_by_en:
            result_query = result_query.order_by("en")

        result_query = result_query.all()

        try:
            next_obj = Category.objects.all()[to_obj]
            has_next = True
        except IndexError:
            has_next = False

        return HttpResponse(paginate(result_query[from_obj: to_obj], "categories",
                                     page_number=page_number, has_next=has_next), content_type=CONTENT_TYPE_JSON)


def tags(request, pk=None):
    if pk:
        try:
            tag = Tag.objects.get(pk=pk)
        except Tag.DoesNotExist:
            tag = None
        return HttpResponse(paginate([tag] if tag else [], "tags", 1, False),
                            content_type=CONTENT_TYPE_JSON)
    else:
```

```
try:
    page_number = int(request.GET.get("page", 1))
    assert page_number >= 1
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid page parameter value"}))
from_obj = (page_number - 1) * PAGE_SIZE
to_obj = page_number * PAGE_SIZE


result_query = Tag.objects


# filtering
filter_by_tag_name = request.GET.get(tags_filter_by_tag_name, None)
if filter_by_tag_name:
    result_query = result_query.filter(tag__contains=filter_by_tag_name)


# sorting
try:
    sort_by = request.GET.get(SORT_BY_PARAM, None)
    assert (sort_by == tags_sort_by_name) or not sort_by
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid sort_by parameter value"}))


if sort_by == tags_sort_by_name:
    result_query = result_query.order_by("tag")


result_query = result_query.all()
```

```
        try:
            next_obj = Tag.objects.all()[to_obj]
            has_next = True
        except IndexError:
            has_next = False

        return HttpResponse(paginate(result_query[from_obj: to_obj], "tags",
                                     page_number=page_number, has_next=has_next),
                            content_type=CONTENT_TYPE_JSON)


def courses(request, pk=None):
    try:
        with_ids = request.GET.get(WITH_IDS_PARAM, "False")
        assert with_ids.lower() == "false" or with_ids.lower() == "true"
        with_ids = with_ids.lower() == "true"
    except AssertionError:
        return HttpResponse(json.dumps({"Error": "Invalid {} parameter value".format(WITH_IDS_PARAM)}))

    if pk:
        try:
            course = Course.objects.get(pk=pk)
        except Course.DoesNotExist:
            course = None
        return HttpResponse(paginate([course] if course else [], "courses", 1, False, **{WITH_IDS_PARAM: with_ids}
                                     ), content_type=CONTENT_TYPE_JSON)
    else:
```

```
try:
    page_number = int(request.GET.get("page", 1))
    assert page_number >= 1
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid page parameter value"}))
from_obj = (page_number - 1) * PAGE_SIZE
to_obj = page_number * PAGE_SIZE


result_query = Course.objects


# filtering
filter_by_platform = request.GET.get(courses_filter_by_platform, None)
if filter_by_platform:
    result_query = result_query.filter(platform__name__in=filter_by_platform.split(","))


filter_by_tag = request.GET.get(courses_filter_by_tag, None)
if filter_by_tag:
    result_query = result_query.filter(tags__tag__in=filter_by_tag.split(","))


filter_by_category = request.GET.get(courses_filter_by_category, None)
if filter_by_category:
    result_query = result_query.filter(category__en__in=filter_by_category.split(","))


filter_by_title = request.GET.get(courses_filter_by_title, None)
if filter_by_title:
    result_query = result_query.filter(title__contains=filter_by_title)
```

```
filter_by_language = request.GET.get(courses_filter_by_language, None)
if filter_by_language:
    result_query = result_query.filter(language__short_name__in=filter_by_language.split(","))

filter_by_rating = request.GET.get(courses_filter_by_rating, None)
if filter_by_rating:
    result_query = result_query.filter(rating__gte=filter_by_rating)

filter_by_search = request.GET.get(courses_filter_by_search, None)
if filter_by_search:
    result_query = result_query.filter(
        preprocessed_description__contains=" ".join(stemming(filter_by_search.split(" "))))

# sorting
try:
    sort_by = request.GET.get(SORT_BY_PARAM, None)
    assert (sort_by in courses_sort_by) or not sort_by
except (ValueError, AssertionError) as e:
    return HttpResponse(json.dumps({"Error": "Invalid sort_by parameter value"}))

if sort_by == courses_sort_by_added:
    result_query = result_query.order_by("-added_date")
elif sort_by == courses_sort_by_title:
    result_query = result_query.order_by("title")
elif sort_by == courses_sort_by_rating:
    result_query = result_query.order_by("-rating")
```

```
result_query = result_query.all()


try:
    next_obj = result_query[to_obj]
    has_next = True
except IndexError:
    has_next = False


return HttpResponse(paginate(result_query[from_obj: to_obj], "courses",
                             page_number=page_number, has_next=has_next, **{WITH_IDS_PARAM: with_ids}),
                    content_type=CONTENT_TYPE_JSON)
```

Android-додаток:

“CoursesObserverAPI.java”

```
package com.vladnamik.developer.coursesobserver.api;



import com.raizlabs.android.dbflow.annotation.NotNull;

import com.vladnamik.developer.coursesobserver.entities.Platform;


import retrofit2.Call;

import retrofit2.http.GET;

import retrofit2.http.Path;

import retrofit2.http.Query;


public interface CoursesObserverAPI {

    String DATETIME_FORMAT_PATTERN = "yyyy-MM-dd HH:mm:ssZ";

    String BASE_URL = "http://77.47.204.127:16522/";


    @GET("platforms")

    Call<SearchPage> getPlatforms(@Query("page")Integer page,

                                  @Query("sort_by")String sortBy,

                                  @Query("name") String fullName);


    @GET("platforms/{id}")

    Call<SearchPage> getPlatform(@NotNull@Path("id")Long id);


    @GET("languages")
```

```
Call<SearchPage> getLanguages(@Query("page")Integer page,
                        @Query("sort_by")String sortBy,
                        @Query("full_name")String fullName);

@GET("languages/{id}")
Call<SearchPage> getLanguage(@NotNull@Path("id")Long id);

@GET("categories")
Call<SearchPage> getCategories(@Query("page")Integer page,
                         @Query("sort_by")String sortBy);

@GET("categories/{id}")
Call<SearchPage> getCategory(@NotNull@Path("id")Long id);

@GET("tags")
Call<SearchPage> getTags(@Query("page")Integer page,
                    @Query("sort_by")String sortBy,
                    @Query("tag")String tag);

@GET("tags/{id}")
Call<SearchPage> getTag(@NotNull@Path("id")Long id);

@GET("courses")
Call<SearchPage> getCourses(@Query("page")Integer page,
                       @Query("title")String title,
                       @Query("sort_by")String sortBy,
                       @Query("platform")String platform,
```

```
                    @Query("category")String category,
                    @Query("tag")String tag,
                    @Query("language")String language,
                    @Query("rating")Integer rating,
                    @Query("with_ids")Boolean withIds,
                    @Query("search")String search);

    @GET("courses/{id}")
    Call<SearchPage> getCourse(@NotNull@Path("id")Long id,
                    @Query("with_ids")Boolean withIds);
}
```

"APIHelper.java"

```
package com.vladnamik.developer.coursesobserver.api;

import com.vladnamik.developer.coursesobserver.components.Application;
import com.vladnamik.developer.coursesobserver.entities.Category;
import com.vladnamik.developer.coursesobserver.entities.Course;
import com.vladnamik.developer.coursesobserver.entities.Language;
import com.vladnamik.developer.coursesobserver.entities.Platform;
import com.vladnamik.developer.coursesobserver.entities.Tag;

import org.androidannotations.annotations.Background;
import org.androidannotations.annotations.EBean;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
```

```
import java.util.List;


public class APIHelper {
    private CoursesObserverAPI api;


    public APIHelper(CoursesObserverAPI api) {
        this.api = api;
    }


    public List<Course> getCourses(CourseSearchParameters parameters) throws IOException{
        SearchPage searchPage;
        List<Course> courses = new ArrayList<>();
        int pageNumber = 1;
        do {
            searchPage = getCoursesPage(parameters, pageNumber);
            courses.addAll(searchPage.getCourses());
            pageNumber++;
        } while (searchPage.getMeta().hasNext());
        return courses;
    }


    public SearchPage getCoursesPage(CourseSearchParameters parameters, int pageNumber) throws IOException{
        SearchPage searchPage;
        List<Course> courses;
        searchPage = api.getCourses(pageNumber, parameters.getTitle(), parameters.getSortBy(),
                toEnumeration(parameters.getPlatforms()), toEnumeration(parameters.getCategories()),
                toEnumeration(parameters.getTags()), toEnumeration(parameters.getLanguages()),
```

```
            parameters.getRating(), parameters.getWithIds(), parameters.getSearch())
            .execute().body();
    return searchPage;
}

public Course getCourse(Long id) throws IOException {
    List<Course> courses = api.getCourse(id, null).execute().body().getCourses();
    if (courses == null || courses.size() == 0) {
        return null;
    } else {
        return courses.get(0);
    }
}

public List<Tag> getAllTags() throws IOException{
    SearchPage searchPage;
    List<Tag> tags = new ArrayList<>();
    int pageNumber = 1;
    do {
        searchPage = api.getTags(pageNumber, null, null).execute().body();
        tags.addAll(searchPage.getTags());
        pageNumber++;
    } while (searchPage.getMeta().hasNext());
    return tags;
}

public SearchPage getTagsPage(Integer page, String tag, String sortBy) throws IOException{
```

```
    return api.getTags(page, sortBy, tag).execute().body();
}

public Tag getTag(Long id) throws IOException {
    List<Tag> tags = api.getTag(id).execute().body().getTags();
    if (tags == null || tags.size() == 0) {
        return null;
    } else {
        return tags.get(0);
    }
}

public List<Category> getAllCategories() throws IOException{
    SearchPage searchPage;
    List<Category> categories = new ArrayList<>();
    int pageNumber = 1;
    do {
        searchPage = api.getCategories(pageNumber, null).execute().body();
        categories.addAll(searchPage.getCategories());
        pageNumber++;
    } while (searchPage.getMeta().hasNext());
    return categories;
}

public Category getCategory(Long id) throws IOException {
    List<Category> categories = api.getCategory(id).execute().body().getCategories();
    if (categories == null || categories.size() == 0) {
```

```
        return null;
    } else {
        return categories.get(0);
    }
}

public List<Language> getAllLanguages() throws IOException{
    SearchPage searchPage;
    List<Language> languages = new ArrayList<>();
    int pageNumber = 1;
    do {
        searchPage = api.getLanguages(pageNumber, null, null).execute().body();
        languages.addAll(searchPage.getLanguages());
        pageNumber++;
    } while (searchPage.getMeta().hasNext());
    return languages;
}

public Language getLanguage(Long id) throws IOException {
    List<Language> languages = api.getLanguage(id).execute().body().getLanguages();
    if (languages == null || languages.size() == 0) {
        return null;
    } else {
        return languages.get(0);
    }
}
```

```
public List<Platform> getAllPlatforms() throws IOException{
    SearchPage searchPage;
    List<Platform> platforms = new ArrayList<>();
    int pageNumber = 1;
    do {
        searchPage = api.getPlatforms(pageNumber, null, null).execute().body();
        platforms.addAll(searchPage.getPlatforms());
        pageNumber++;
    } while (searchPage.getMeta().hasNext());
    return platforms;
}

public Platform getPlatform(Long id) throws IOException {
    List<Platform> platforms = api.getPlatform(id).execute().body().getPlatforms();
    if (platforms == null || platforms.size() == 0) {
        return null;
    } else {
        return platforms.get(0);
    }
}

private String toEnumeration(List<String> elements) {
    if (elements.size() == 0) {
        return null;
    }
    StringBuilder stringBuilder = new StringBuilder();
    for (String element: elements) {
```

```
            stringBuilder.append(element);
            stringBuilder.append(",");
        }
        stringBuilder.deleteCharAt(stringBuilder.length() - 1);
        return stringBuilder.toString();
    }
}
```

"BasicDataLoader.java"

```
package com.vladnamik.developer.coursesobserver.utils;

import com.vladnamik.developer.coursesobserver.api.CourseSearchParameters;
import com.vladnamik.developer.coursesobserver.components.Application;
import com.vladnamik.developer.coursesobserver.database.DBHelper;
import com.vladnamik.developer.coursesobserver.entities.Category;
import com.vladnamik.developer.coursesobserver.entities.Language;
import com.vladnamik.developer.coursesobserver.entities.Platform;
import com.vladnamik.developer.coursesobserver.entities.Tag;

import java.io.IOException;
import java.util.List;

public class BasicDataLoader implements DataLoader{

    private Application application;
    private DBHelper dbHelper = new DBHelper();
```

```
    public BasicDataLoader(Application application) {
        this.application = application;
    }

    @Override
    public List<Platform> getAllPlatforms() {
        updateDBInfoIfNeeded();
        return dbHelper.getAllPlatforms();
    }

    @Override
    public List<Category> getAllCategories() {
        updateDBInfoIfNeeded();
        return dbHelper.getAllCategories();
    }

    @Override
    public List<Language> getAllLanguages() {
        updateDBInfoIfNeeded();
        return dbHelper.getAllLanguages();
    }

//    @Override
//    public List<Tag> getAllTags() {
//        updateDBInfoIfNeeded();
//        return dbHelper.getAllTags();
//    }
```

```
@Override
public List<Tag> getTags(String tag) throws IOException{
   return application.getApiHelper().getTagsPage(1, tag, null).getTags();
}

public void updateDBInfo(final List<Platform> platforms, final List<Category> categories,
                  final List<Language> languages) {
   dbHelper.deleteAllData();
   dbHelper.savePlatforms(platforms);
   dbHelper.saveCategories(categories);
   dbHelper.saveLanguages(languages);
}

public void updateDBInfoIfNeeded() {
   new Thread(new Runnable() {
      @Override
      public void run() {
         if (!Utils.isDBRelevant(application)) {
            try {
               List<Platform> platforms = application.getApiHelper().getAllPlatforms();
               List<Category> categories = application.getApiHelper().getAllCategories();
               List<Language> languages = application.getApiHelper().getAllLanguages();

               updateDBInfo(platforms, categories, languages);

               Utils.setDBChecked(application);
```

```
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }).start();

}

@Override
public CoursesIterator getCoursesIterator(CourseSearchParameters parameters) {
    return new CoursesIterator(parameters, application.getApiHelper());
}

}
```