НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проєктування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Мобільний додаток для проходження дистанційних навчальних курсів з адаптивною системою формування змісту»

Виконав:
студент IV курсу, групи ТІ-71
Ворвуль Данило Максимович _____

Керівник:
к.т.н доц. Шалденко Олексій Вікторович _____

Рецензент:
к.т.н доц. Шалденко Олексій Вікторович _____

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

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проєктування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

_____
(прізвище, ім’я, по батькові)

1. Тема роботи Мобільний додаток для проходження дистанційних навчальних курсів з адаптивною системою формування змісту
керівник роботи к.т.н. доц. Шалденко Олексій Вікторович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”____” травня 2021р. № _____

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Мови програмування python, фреймворки Django, Django Rest Framework, кроссплатформенный фреймворк для мобільної розробки Flutter на мові Dart, Docker-compose, Docker, javascript, система для створення онлайн курсів та їх проходження
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Ознайомитись з наявними аналогічними програмними системами. Розробити алгоритм програмної системи. Запрограмувати дану систему. Провести тестування та налагодження даної системи. Провести оптимізацію коду системи. Випробувати створену систему.
5. Перелік ілюстративного матеріалу
Титульна сторінка, Актуальність, Огляд інших систем, Огляд вибраних технологій

розробки, Діаграма сутностей, Діаграма класів, Інші сторінки застосунку, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”04” жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____

АНОТАЦІЯ

У роботі яка складається з 60 сторінок представлено діаграми, схеми збудованої системи, також проілюстровано скріншотами користувацький досвід, лістинг коду.

Головною метою цієї роботи було дослідити найкращі практики в онлайн освіті та створити мобільний застосунок для проходження курсів. На сьогоднішній день розвиток - це новий тренд, переважання мобільного трафіку над десктопним - це факт, тому програмний продукт та ідеї які з’явилися під час роботи над дипломною роботою являється актуальним та відображають сьогоденні реалії.

Перед розробкою було проведено аналіз технологій та підібрано найкращу комбінацію під нашу задачу: Django + DRF для вебсерверу, Flutter для розробки кросплатформенного мобільного застосунку, Також було було пропрацьовано інфраструктуру з боку розробника - створено CI/CD систему для автоматичної інтеграції нововведень на веб сервер.

Ключові слова: *ОСВІТА, НАВЧАННЯ, КУРСИ, ДИСТАНЦІЙНА ОСВІТА*.

ABSTRACT

The work, which consists of 60 pages, presents diagrams, diagrams of the built system, also illustrated with screenshots of user experience, code listing.

The main purpose of this work was to explore best practices in online education and create a mobile application for courses. Today, development is a new trend, the predominance of mobile traffic over desktop is a fact, so the software product and ideas that appeared during the work on the thesis are relevant and reflect today's realities.

Before development, we analyzed the technology and selected the best combination for our task: Django + DRF for web server, Flutter for cross-platform mobile application development.

Key words: EDUCATION, TRAINING, COURSES, DISTANCE EDUCATION.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IP - internet protocol.

DRF - Django Rest Framework.

API - Application Programming Interface.

CRUD - операції створення, читання, оновлення, видалення (Create, Read, Update, Delete).

GCP - Google cloud platform.

Інстанс - сервер на GCP.

ВМ - віртуальна машина, також сервер на GCP.

ЗМІСТ

ВСТУП

У цій роботі було розглянуто основні тенденції в онлайн освіті, вплив збільшення мобільного трафіку на освітні процеси та запропоновано варіант як можно зробити освіту у школах більш сучасною та діджиталізованою.

З початку пандемії коронавірусу людство стало мати більше часу, через те, що не потрібно нікуди їхати, все стало дистанційним. Велика кількість людей задумалась про те, що з’явився час на підвищення своєї кваліфікації та розвиток, тому тема дипломної бакалаврської роботи наразі являється актуальною. Зараз спостерігається бум у сфері онлайн освіти та акції таких компаній як Coursera, Udemy злетіли вгору. Але школи та університети не були готові та не змогли швидко підлаштуватись під реалії нового світу, це не є погано, бо такого масштабного карантину в світі ніколи не було.

Робота є актуальною, тому що мобільні застосунки стають з кожним роком популярнішими над звичайним вебом, а також зараз спостерігається бум на саморозвиток, тому багато людей шукає зручний для себе застосунок для підвищення кваліфікації.

У цій роботі було розглянуто:

— як можна покращити освітні процеси під час пандемії, та розроблено таку систему, що не втратить своєї актуальності після виходу з карантину та віддаленого навчання;

— як зробити оцінки в університеті або школі більш вагомими для тих, хто їх отримує та для роботодавців;

— як заохотити на здобування знань більшу кількість людей;

— проаналізовано аналоги та розглянуто чому їх не можна використовувати як готові рішення для всієї сфери освіти в Україні;

— описано розробку системи від початкового аналізу предметної області, побудування діаграм до написання коду.

Метою не є створення застосунку, щоб увесь навчальний процес у школах і університетах був дистанційним, цей застосунок повинен покращити навчальний процес та привнести в нього елемент гейміфікації.

1. ПОСТАНОВКА ЗАДАЧІ З КОНСТРУЮВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ПРОХОДЖЕННЯ ОНЛАЙН КУРСІВ

Мета - створити мобільний застосунок, який надасть можливість проходити навчальні курси, а також може використовуватися для надання освіті нових можливостей, де в кінці кожного курсу буде видана не тільки оцінка, а ще сертифікат.

Мобільний застосунок призначений:

— надавати можливість виконувати завдання з навчальних курсів;

— надавати можливість проходити в рамках курсу текстові лекції – теоретична частина, може містити в собі текст, який редагується за допомогою RichTextEditor (у тексті теорії можуть міститися картинки, відео, розмітка MarkUp).

Цільова аудиторія платформи включає наступні групи людей:

— студенти ВУЗів;

— школярі;

— люди, які хочуть змінити сферу діяльності.

1.1 Структура мобільного застосунку

Сайт повинен складатися з наступних розділів:

Пункти меню:

— Доступні курси

— Мої курси

— Сертифікати

Об’єкти застосунку:

— Адмін панель

— Календар

— Адмін

— Вчитель

— Учень

— Сертифікат

— Курс

— Текстова лекція

Модифікація вмісту розділів повинна здійснюватися за допомогою адміністраторського веб-інтерфейсу (системи управління сайтом), який без застосування спеціальних навичок програмування (без використання програмування і спеціального кодування або форматування) повинен передбачати можливість редагування інформаційного вмісту сторінок сайту. Система управління контентом (адміністративна частина сайту) повинна надавати можливість додавання, редагування та видалення вмісту статичних та динамічних сторінок.

Система управління контентом повинна мати стандартний web-інтерфейс, який відповідає таким вимогам:

— в графічному віконному режимі;

— єдиний стиль оформлення;

— інтуїтивно зрозуміле, зручне призначення елементів інтерфейсу;

— відображення на екрані тільки тих можливостей, які доступні конкретному користувачу;

— для операцій з масового введення інформації повинна бути передбачена мінімізація кількості натискань на клавіатуру для виконання стандартних дій.

2. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Інформаційні технології в освіті та навчанні з кожним днем стають все більш популярними. Інтерес до них підігріває постійне вдосконалення цифрового подання інформації та спрощення доступу до будь-яких даних із застосуванням комп'ютерних мереж. Сьогодні онлайн-навчання все частіше розглядається не просто як зручна форма підвищення кваліфікації, а як цілком серйозна альтернатива традиційному освіти, що дозволяє отримати знання. Доступ до навчального курсу через Інтернет дає можливість в комфортних умовах і в зручний час вивчати матеріал, а значить - і краще його засвоїти. З іншого боку навчається не займає місця в навчальній аудиторії. Таким чином, її можна використовувати паралельно до освіти в традиційному розумінні.

Ще одним плюсом онлайн-курсів є наявність чіткої програми. Користувач вже на початку курсу розуміє які етапи навчання йому належить пройти і
які знання в результаті він отримає. Підтвердженням цього можуть бути різного роду дипломи, сертифікати та атестати.

Дистанційне навчання - це форма отримання освіти, при якій викладач і студент взаємодіють на відстані за допомогою інформаційних технологій. Під час дистанційного навчання студент займається самостійно за розробленою програмою, переглядає записи вебінарів, вирішує завдання, проводить консультації з викладачем в онлайн-чаті і періодично видає йому на перевірку свої роботи. Дистанційне навчання стало популярним з появою інтернету, відкривши нові можливості розвитку для жителів віддалених населених пунктів і ділових людей з щільним робочим графіком. Спочатку дистанційне навчання сприймалося лише як додатковий спосіб придбання знань або підготовки до іспитів. Зараз можна пройти повноцінні дистанційні курси і програми підвищення кваліфікації від престижних університетів, комерційних і некомерційних компаній з різних країн, перебуваючи в будь-якій точці планети.

3. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ОНЛАЙН ОСВІТИ

Завдяки онлайн-курсів сьогодні кожен має можливість віртуально вчитися в передових університетах та бізнес-школах на кшталт MIT і Стенфорда, слухати лекції і проходити майстер-класи у майстрів своєї професії з Америки і Європи, а попутно ще й поднатаскать себе в розмовному і спеціальному англійською. І все це - абсолютно безкоштовно. Онлайн-освіта - дуже приваблива нива, на якій множаться все нові платформи на зразок добре відомих Coursera і Prometeus.

На сьогоднішній день мною було пройдено вісім курсів на Coursera, три курс на Udemy.com, один на SkillShare і ще з десяток курсів на TUTS +. Тому я може проаналізувати плюси й мінуси кожної платформи як користувач.

3.1 Платформа “Coursera”

З усього перерахованого, курси на Coursera найбільше нагадують академічну освіту. Звичайно, їх точно не можна порівняти з аналогами тих курсів, які безпосередньо читають в Stanford або MIT. Але, мені здається, що по наповненню і структурою вони найбільше відповідають університетських курсів. Підхід до викладу дисципліни досить грунтовний, включає і загальне розуміння предмета, передбачає вивчення додаткових матеріалів і практичні завдання.

Те, що пропонує Coursera - це однозначно унікальний продукт, унікальний контент. Так, наприклад, до курсу «Understanding Media by Understanding Google» («Розуміти медіа, розуміючи Google»), крім звичайних лекцій від професора, додавалися також записані ним півгодинні інтерв'ю з авторами тих книг, які були в списку рекомендованої до курсу літератури. І сам курс, і ці інтерв'ю були для мене надзвичайно цікаві. Але в курсі мова йшла не про практичну роботу в Google AdWords, Analytics або Google Tag Manager. Піднімалися загальні питання - питання авторського права, пошуку vs дослідження, як нові продукти впливали на традиційні

медіа і т. Д. Це дало більш ґрунтовне розуміння і сучасного медійного простору, і актуальних законів створення, і поширення контенту. Цей курс не підійде тим, хто хоче отримати практичні знання, але дасть глибоке розуміння сучасного інформаційного простору і пов'язаних з ним проблем і можливостей.

Курс по «Human-Computer Interaction» («Взаємодія людина-комп'ютер») був першим, який я закінчив, і до сих пір залишається найскладнішим. Крім вивчення історії самої дисципліни та унікальної методології проведення UX-досліджень і тестувань на різних стадіях розробки продукту, курс також припускав абсолютно реальне створення працюючого прототипу. Думаю, що саме на цьому етапі відсіювалась основна маса студентів, оскільки розробка прототипу включала в себе вивчення досить непростого софта Axure RP або Justinmind. Я витратив на це мінімум годин 10-15. Треба віддати належне платформі - Coursera надає ліцензійний ключ до всіх необхідних для роботи програм.

Окремо хочу відзначити, що курси на Coursera проходити легше тим, у кого проблеми з самодисципліною. Оскільки наявність чітких дедлайнів дуже дисциплінує і стимулює. Для мене Coursera - це спроба відтворити академічну освіту. І досить благородна мета зробити якісну університетську освіту доступнішою для тих, у кого немає до нього доступу.

3.2 Інші платформи (Udemy, SkillShare, TUTS +)

Якщо говорити про курси на зазначених ресурсах, то їх менше можна віднести до академічної освіти. Це скоріше те, що всі звикли називати туторіали, зовсім інший вид навчання. Величезний плюс цих курсів в тому, що відеолекції готують і записують справжні практики, професійні дизайнери, фронтенд девелопери і тд.

Але є нюанс. Щоб проходити ці курси, потрібна велика самомотивація. Немає ні форумів з інструкціями, ні дедлайнів. У цьому може бути проблема. Сумарно відеоуроки по курсу можуть займати всього пару годин, але проходити їх можна місяць. У мене таке часто бувало.

Udemy, SkillShare або TUTS + варто вибирати, якщо ви хочете розвинути конкретні навички (зрозуміти принципи flat design, розібратися з Less, навчитися коштувати responsive веб-сторінки за допомогою Bootstrap і т. Д.) Або не відставати від світових трендів (нових концепцій, інструментів , софта) і вчитися у представників авангарду професії, трендсеттеров.

3.3 Онлайн-освіта vs. офлайн-освіта

Звичайно, на мою думку, нічого не зрівняється з повноцінним університетською освітою європейського або американського зразка. Онлайн-курси не можуть замінити університетську освіту. Але, можливо, онлайн може стати ще одним каналом отримання класичної освіти. Я не бачу різниці між обговоренням в аудиторії або на сторінках форуму. Тут скоріше справа в контенті, компетентності професорів, структурі і продуманості дисциплін, а не в форматі.

Позитивно оцінюють також поява на Coursera спеціалізацій - кількох курсів суміжного напряму. Можливо, це просто один із способів монетизації майданчика, адже для отримання спеціалізації обов'язково необхідно отримувати платний сертифікат. Але для мене спеціалізації більш зрозумілі і корисні, ніж окремі, не пов'язані між собою курси. Що ж стосується Udemy, SkillShare, TUTS +, то для мене це хороший спосіб скоріше не здобувати освіту, а вдосконалювати навички та дізнаватися новинки.

Домашня обстановка відрізняється від навчальної колосальним чином. Наші нейронні зв'язки з'єднані так, що будинок став синонімом відпочинку, комфорту. Ми повертаємося додому, щоб набратися сил після складного дня. Це поведінка всього людства на протязі історії. Тому це навіть не просто нейронні зв'язки, а щось, закріплене в рептильної мозку, від чого не позбутися по клацанню пальців.

На карантині ми все з'ясували: домашня обстановка не сприяє концентрації на важливих професійних питаннях. Раптом пріоритетом стає зайвий перекус, перекур,

генеральне прибирання, поливання квітів. Все, що завгодно, але тільки не напруга мозку і вже тим більше не освоєння нових знань і навичок.

Атмосфера в аудиторії, навпаки, запрограмована на отримання серйозного контенту, вона навчальна. Так, звичайно, там теж є відволікаючі фактори, але все ж мозок схильний зупиняти нас, через тих же нейронних зв'язків: прийшов сюди вчитися, переміг себе і відстань - вчися.

Нарешті, живе навчання - це про сприйняття контенту завдяки контакту викладача і студента. Коли спікер бачить зацікавлені очі студентів, він відчуває свою відповідальність і значущість і починає вкладатися в свою роботу сильніше: придумує нові формати навчання, ретельніше готується до занять. Студенти, в свою чергу, бачачи викладача безпосередньо перед собою і відчуваючи його живу енергетику і інтерес в тому, що він робить, самі виявляються зацікавленими і максимально концентруються на отриманні інформації. Це підвищує рівень залученості учнів в процес і як наслідок підвищує рівень освоєння матеріалу.

Специфіка онлайн-освіти в тому, що воно дозволяє заходити в процес навчання в більш комфортних умовах. Але тут є і пастка: ці комфортні умови можуть недостатньо розташовувати до навчання, і більше того, відволікати.

Палка з двома кінцями: з одного боку, онлайн-навчання - це можливість вчитися де і коли зручно, з іншого - підвищені вимоги до дисципліни і мотивації учня.

Традиційні лекції складаються з двох частин: під час першої спікер виступає на задану тему, під час другої - експерт / викладач відповідає на питання, що цікавлять студентів питання. Іноді спікер не проти відповідати на питання по ходу виступу, що практично нездійсненно в онлайні. Такий формат найкраще працює в живій навчальної обстановці: інтерес учнів стикається з любов'ю спікера до предмету і його великим досвідом. Один питає, другий ділиться - виходить такий собі обмін енергіями, синергія, швидко створюється коннект. Це мотивує ставити ще більше питань, а значить дізнаватися більше. Крім того, такий формат дозволяє спікеру адаптувати програму під кожну групу, загострити увагу на моментах, які дійсно цікаві студентам. Онлайн-навчання ж часто складається з уже колись

записаних вебінарів. Про яку зв'язку між викладачем і студентом може йти мова, якщо між вами навіть немає зорового контакту?

Повертаючись до офлайну, більшість спікерів відкриті до співпраці. Вони із задоволенням залишають студентам свої контакти, щоб ті мали змогу поставити запитання, які з'являться вже після зустрічі. Тобто навчання виходить за рамки навчальних аудиторій і це дійсно здорово.

Якщо уявити, що онлайн-навчання забезпечується хорошим увагою учня, то онлайн виграє - учень може вчитися у власному темпі, глибше усвідомлювати питання. У цьому сенсі онлайн прогресивніше і якісніше, знову ж таки за умови достатньої уваги з боку учня.

Ковідна пандемія всіляко намагається вплинути на суспільне сприйняття соціалізації. За період масового переходу в онлайн ми помітили, що постійне сидіння вдома перед комп'ютером не може мати довгострокові перспективи. Люди втомилися від ізоляції і онлайну, тому зараз ми бачимо, що попит на оффлайн набуває нового імпульсу. Упевнений, що після загальної вакцинації він буде ще вище.

Всі ми знаємо, що одна з ключових функцій освіти - соціалізація. Важливо розуміти, що зараз багато хто йде вчитися не тільки заради контенту, але і заради нетворкінгу. У наш час це визначальний фактор. В онлайні дістати професійні зв'язки і контакти дуже складно, оскільки люди схильні більше довіряти тим, кого вони знають наживо. Живе спілкування та спільну справу (навчання в одній аудиторії) створює міцний зв'язок між студентами, яка підтримується протягом багатьох років після закінчення курсів.

Тому обов'язок кожного оффлайн освітнього закладу створити атмосферу, що сприяє соціалізації, наприклад, для наших студентів стіни RMA буквально стають другою домівкою, де комфортно кожному.

Крім того студентське середовище, в якій люди діляться своїми враженнями та досвідом, роблять помилки і показують свої успіхи в навчанні, сприяє отриманню знань найсприятливішим чином. Коли студент бачить крутий досвід іншого, він

надихається, мотивується, стає більш дисциплінованим - нікому не хочеться впасти обличчям в бруд перед однокурсниками.

А якщо ще навколо одні однодумці, то створюється відчуття спільності і взаємодопомоги, команди. Не зрозумів щось в темі - підійшов і запитав у тих, хто сидить поруч з тобою в аудиторії: створюється потужний обмін ідеями і знаннями.

Бізнес-освіта - це навчання, побудоване цілком на мотивації учнів і їх бажанні вчитися. Якщо є тема, яка знайома або нецікава, то її цілком можна пропустити - ніхто двійку за відвідування не поставить. Якщо пропустив лекцію не спеціально, то можеш прийти на те же заняття з іншою групою і заповнити прогалини.

Зараз оффлайн бізнес-школи часто підлаштовуються під своїх учнів і під їх бажання вчитися. Дійсно, онлайн-освіта дає велику можливість гнучкого підходу при учнів, і разом з цим висуває більше вимог до його здатності і бажання творчо взяти участь у тюнінгу його освітньої траєкторії.

Об'єктивно, деякі речі з курсів можна пропускати, якщо ти з ними знайомий, але щоб розуміти, чи знайомий ти з цією темою в достатній мірі, потрібно бути самому досить зрілим. Потрібно бути співавтором своєї освітньої траєкторії - це важливо при навчанні. Якщо ти правильно це зробиш, то зекономиш час, і отримаєш більше занурення в матеріал.

3.4 Висновки

Можна зробити висновки, що треба створити застосунок, який надасть змогу проходити курси, в ньому повинні бути чіткі дедлайни, щоб допомогти тим, хто сам не може розподіляти свій час, за курси потрібно видавати сертифікати, бо це наш мозок бачить як досягнення та виділяє дофамін.

Також було доведено, що застосунок не має робити офлайн освіту повністю онлайн, бо очні заняття, наприклад, у Skype, Google meet, Zoom, повинні проводитися. Тобто у застосунку можна подивитися свої сертифікати, оцінки за завдання з курсів, курси в процесі, закінчені курси.

4. ЗАСОБИ РОЗРОБКИ

Для вирішення поставленої проблеми було проведено аналіз сучасних мов програмування та фреймворків, які задовольняють умовам, що будуть перелічені у наступних розділах. Умови будуть розташовані від найпріорітетнішої до найменш пріорітетної.

4.1 Умови, яким повинен відповідати фреймворк для розробки мобільного застосунку

1. Можливість писати на одній мові одразу під дві мобільні платформи - IOS та Android. Це зменшує час на розробку та підтримку програмного продукту майже в два рази та дозволяє навіть одній людині підтримувати та додавати нововведення до застосунку.

2. Фреймворк повинен бути найсучаснішим та отримувати оновлення мінімум раз на рік. Часті оновлення покращують стабільність програмного забезпечення та лагодять небезпечні вразливості.

3. Велика кількість готових віджетів та компонентів UI мобільного застосунку. Готові рішення дозволяють не витрачати час не дебаг коду, а використовувати вже готові та протестовані рішення.

4. Наявність зручного дебаггера HTTP запитів.

5. Наявність функції “Hot reload”. Дана функція дозволяє миттєво відображати зміни в коді на екрані мобільного телефону.

6. Наявність DEV-інструментів, щоб можно було оцінити швидкість роботи застосунку та не допустити витік пам’яті (таке часто зустрічається при помилках на Java).

7. Фреймворк повинен “вміти” працювати з віртуальним девайсом (емуляцією Android/IOS смартфона) під час розробки та дебагу.

8. Статична типізація мови фреймворку. Статична типізація дозволяє не допустити значну кількість помилок на етапі розробки.

9. Можливість вести розробку з Windows або Linux.

4.2 Умови, яким повинен відповідати фреймворк для розробки API для мобільного застосунку

1. Автогенерація адмін-панелі на основі описаних моделей для CRUD (Create, Read, Update, Delete) операцій. Ця можливість економить дуже багато часу, бо не треба додатково розробляти CRUD для адмінів.

2. Кастомізація авто-згенерованої адмін панелі. Кастомізація дозволяє забути про окрему адмін-панель окрім авто-згенерованої, бо можна настроїти все під потреби користувачів системи. Також це дозволяє не витрачати час на дебаг нововведень а використовувати готові рішення.

3. Можливість запустити фреймворк у докері. Запуск у докері буде використовуватися для того, щоб налаштувати CI/CD.

4. Автогенерація та корегування автогенерації SWAGGER документації. Це дозволить менше часу витрачати на написання документації та швидко переходити від написання endpoint-у до розробки фронтенду застосунку.

5. Статична типізація мови програмування, на якій написано фреймворк. Переваги статичної типізації над динамічною було описано у пункті 3.1.8.

4.3 Умови, яким повинен відповідати провайдер для хостингу API та бази даних

1. Можливість швидко змінювати конфігурацію серверів для масштабування застосунку.

2. Наявність безоплатного тестового періоду. За тестовий період є можливість спробувати усі сервіси, які нам потрібні і вже тільки після цього вирішити чи залишатися з цим провайдером чи ні.

3. Наявність load balancer системи. Load balancer систему будемо застосовувати на піках активності користувачів, щоб підвищити потужність серверів на короткий час та знову зменшити до оптимальної задля економії коштів та задовільнення потреб користувачів.

4. Наявність налаштування серверу бази даних через адмін-панель провайдера. Це дозволяє менше уваги приділяти конфігурації бази даних.

5. Підтримка останньої версії PostgreSQL. Було обрано базу даних PostgreSQL, бо вона надає більше можливостей у збереженні даних у порівнянні з класичним MySQL.

6. Можливість оренди у провайдера доменного імені, щоб усі запити до API було захищено SSL сертифікатом.

4.4 Вибір технологій для розробки мобільного застосунку

У сучасній мобільній розробці зараз домінують такі мови програмування:

— Java, Cotlin (Android);

— Swift (IOS);

— React Native, Flutter (IOS and Android).

Java. Cotlin, Swift задовольняють всім вище перечисленим пунктам про мову/фреймворк для розробки мобільних застосунків, окрім, як на мене, найважливішого - ці мови не дають розробляти застосунок одразу на IOS та Android, не можна назвати вибір цих мов для розробки поганим, бо нативна розробка також має свої переваги, але для досягнення нашої цілі найбільш підійде кросплатформенна мова або фреймворк, тому не бачу сенсу роздивлятися їх у подальшому аналізі.

Наступним інструментами розробки, які буде розглянуто, є React Native та Flutter [12]. Для зручності у порівнянні було збудовано таблицю.

Таблиця 1 - Порівняльна характеристика Flutter та React Native

	Flutter	React Native
Що це?	Портативний набір користувальницьких інтерфейсів для створення власних скомпільованих програм на мобільних пристроях, в Інтернеті та для ПК з єдиної кодової бази. [12]	Фреймворк для створення власних додатків за допомогою React.
Офіційний реліз	грудень 2018 року, Google I / O	березень 2015 року, конференція F8
Мова програмування	Dart	JavaScript
Популярність	120 000 зірок на Github (травень 2021)	95 300 зірок на Github (травень 2021)
Hot reload	+	+
Native performance	Добре	Добре
UI	Програми Flutter виглядають на сучасних операційних системах так само добре, як і на старих версіях. [13] Оскільки у них лише одна	Компоненти програми виглядають так само, як і власні (наприклад, кнопка на пристрої iOS виглядає так само, як кнопка власного iOS, і така ж на

	кодова база, програми виглядають і поводяться однаково в iOS та Android - але завдяки віджетам Material Design та Cupertino вони також можуть імітувати сам дизайн платформи. Як це можливо? Flutter містить два набори віджетів, які відповідають певним мовам дизайну: Віджети Material Design реалізують однойменну мову дизайну Google; Віджети Купертіно імітують дизайн Apple від iOS. Це означає, що ваш додаток Flutter виглядатиме та поводитиметься природно на кожній платформі, імітуючи їхні рідні компоненти [15].	Android) [14]. Той факт, що React Native використовує власні компоненти під капотом, повинен надати вам впевненості, що після будь-якого оновлення інтерфейсу ОС компоненти вашого додатка також будуть миттєво оновлені. Однак це може порушити користувальницький інтерфейс програми, але це трапляється дуже рідко. Якщо ви хочете, щоб ваш додаток виглядав майже однаково на всіх платформах, а також у старих версіях операційної системи (як це досягає Flutter), тоді подумайте про використання сторонніх бібліотек (як ця). Вони дозволять вам

		використовувати компоненти Material Design замість рідних.
Sharing code	За допомогою Flutter 2 (анонсованого в березні 2021 р.) Ми можемо використовувати ту саму кодову базу для доставки власних додатків до п’яти операційних систем: iOS, Android, Windows, macOS та Linux; а також веб-досвід, орієнтований на браузери, такі як Firefox, Chrome, Safari або Edge. Метушня навіть може бути вбудована в машини, телевізори та розумну побутову техніку. (джерело) [16]. Мабуть, найбільшим оголошенням у Flutter 2 є підтримка Інтернету якісної продукції. Він може бути використаний для:	iOS та Android - але є вибрані бібліотеки, які дозволяють використовувати той самий код для створення програм для iOS, Android, Інтернету та Windows10. Ви також можете витягти спільний код у мобільних, настільних та веб-програмах в окреме сховище; розглядати це як окремий проект; потім вводять його так само, як іншу залежність. Це дозволяє розробнику зосередитися на написанні коду для певної платформи, не розглядаючи питання сумісності з іншою [17].

	Прогресивні веб-програми (PWA), які поєднують охоплення Інтернету з можливостями настільного додатка, Додатки для однієї сторінки (SPA), які завантажуються один раз і передають дані до та з Інтернет-сервісів. Існуючі мобільні програми - дозволяючи запускати програми Flutter на ПК.	
Час виходу на ринок	Зазвичай набагато швидше, ніж native development.	Можливо, так швидко, як розробка за допомогою Flutter. Однак ... React Native використовує мостові та власні елементи, тому може знадобитися окрема оптимізація для кожної платформи - проблема, з якою Flutter на основі віджетів не стикається. Це

		може продовжити розробку додатків за допомогою React Native.
Конкурентна перевага	• Чудовий зовнішній вигляд завдяки багатим віджетам; • Швидко зростаюче співтовариство та популярність; • Відмінна документація із потужною підтримкою команди Flutter (що полегшує початок розробки за допомогою Flutter); • Покращення Flutter для Інтернету, пропонуючи потенціал для однієї кодової бази на мобільних та веб-платформах • Важко обіграти тривалість виходу на ринок	• Стабільність (5+ років на ринку); • Багато успішних, видатних гравців ринку, що використовують React Native; • Зріла, величезна громада; • Проста у засвоєнні технологія; • Багато навчальних посібників та бібліотек, що дозволяють швидко та легко розвиватися; • Код може бути легко використаний як для веб-програми, так і для розробки настільних додатків.
Коли технологія не	Якщо ...	Якщо …

підійде	• Ваш додаток повинен підтримувати 3D Touch (наразі Flutter не підтримує 3D - але він є в довгостроковій дорожній карті команди Flutter) • Дизайн вашого додатку залежить від платформи [18] • Ваш додаток вимагає багаторазової взаємодії з ОС; або вимагає рідкісних, маловідомих рідних бібліотек [19] • Вам потрібен мінімалістичний інтерфейс, але покла • дайтесь на значне використання апаратного забезпечення телефону (наприклад,	• Ваш додаток повинен виконувати менш поширені або надто конкретні завдання (наприклад, обчислення) у фоновому режимі • Вам потрібен спеціальний зв’язок через Bluetooth (що може бути складно реалізувати за допомогою React Native) [20] • Ви хочете створити програму лише для Android По правді кажучи, якщо ви хочете створити додаток для iOS і ви знаєте JavaScript, розгляньте React Native - але якщо ви хочете лише додаток для Android, швидше за все, побудувати за власним бажанням з іншою

	програма, яка відтворює музику або робить лише фотографії) ● Ви хочете створити миттєвий додаток (невеликий додаток) Якщо ваш додаток схожий на будь-що з перерахованого, можливо, краще вибрати нативну розробку додатка.	командою. Чому? Зараз iOS має кращу підтримку, ніж Android.

На основі таблиці 1 можна зробити висновок, що для нашої задачі найкраще підходить Flutter, бо в нього більше переваг для вирішення проблеми з нашої предметної області та задачі в цілому.

4.5 Вибір технологій для розробки API

Було визначено, що на Dart (мова фреймворку Flutter) можна написати будь що, не є винятком й API сервер. Аналіз найпопулярніших Dart серверних фреймворків показав, що всі вони не задовольняють одній з найважливіших для нашої проблеми умови - вони не мають автогенерованої адмін-панелі з можливістю її кастомізувати, а також не має готових пакетів, які б дозволили згенерувати SWAGGER документацію.

Враховуючи вище перелічені проблеми я вирішив для бекенду використати Django написаний на мові Python. Цей фреймворк відповідає всім вимогам, які були перелічені, але Python - динамічно типізована мова, що, як на мене є більше мінусом, бо треба більше писати тестів за для впевненості у коректності роботи застосунку. Лозунг Django [1] виглядає так: “Django - The web framework for perfectionists with deadlines” тобто його сутність у швидкій розробці застосунку.

Django надає “з коробки” такі корисні можливості:

— генерація адмін панелі та її кастомізація;

— можливість настроїти email сповіщення;

— готова авторизація;

— готова архітектура для веб-застосунку (Model, View, Template);

— велика кількість готових рішень;

— висока зацікавленість цим фреймворком на Stack Overflow, Stack Share, тощо.

Але в чистому Django потрібно провести деяку кількість часу, щоб адаптувати його під зручну розробку API ендпоінтів. Тому було вирішено застосувати Django Rest Framework [2] - пакет, у якому уже проведено роботу по вдосконаленню робочого процесу.

Як базу даних будемо використовувати PostgreSQL, бо у неї дуже багато переваг перед аналогічними технологіями.

4.6 Вибір провайдеру для хостингу

З вибором провайдеру при наших умовах ситуація складається наступна: потрібно обирати між Google Cloud Platform [4] (далі GCP) та Amazon Web Services (далі AWS).

Однозначно кращою ні одну платформу назвати не можна.Але я маю рік досвіду з AWS та на мою думку у цієї платформи дуже багато непотрібного мікроменеджменту і часто ситуація складається так, що коли тобі потрібно щось зробити і нема людини поблизу, яка таке вже зробила, то потрібно довго мандрувати по перехресним посиланням у документації. Тому я обрав GCP як більш привабливішу платформу для цього проекту.

4.7 Вибір бази даних

Перший вибір який необхідно зробити у цьому розділі це вибір між реляційною базою даних та не реляційною. Для нашого застосунку очевидно, що треба використовувати реляційну базу даних, бо у діаграмі сутностей багато зв’язків між таблицями та буде багато пошуку за foreign keys.

Другий вибір - це вибір між існуючими реляційними базами даних. Для цього збудовано порівняльну характеристику в таблиці 2.

Таблиця 2 – Порівняльна характеристика MySQL та PostgreSQL

	MySQL	PostgreSQL
Коротка історія	Розробка MySQL почалася ще в 90-х роках. Перший внутрішній випуск бази даних відбувся в 1995 році. За	Розробка Postrgresql почалася в далекому 1986 році в стінах Каліфорнійського університету Берклі.

	цей час розробкою програми займалися кілька компаній. Розробка була розпочата шведською компанією MySQL AB, яку придбала Sun Microsystems, яка, власне перейшла у власність Oracle. На даний момент, починаючи з 2010 року, розробкою займається Oracle.	Розробка тривала майже вісім років, потім проект розділився на дві частини комерційну базу даних Illustra і повністю вільний проект Postrgesql, який розробляється ентузіастами.
Збереження даних	MySQL - це реляційна база даних, для зберігання даних в таблицях використовуються різні движки, але робота з двигунами захована в самій системі. На синтаксис запитів і їх виконання движок не впливає. Підтримуються такі основні движки MyISAM, InnoDB, MEMORY, Berkeley DB. Вони відрізняються між собою способом запису даних на диск, а також	Postgresql вдає із себе об'єктно реляційну базу даних, яка працює тільки на одному двигуні - storage engine. Всі таблиці представлені у вигляді об'єктів, вони можуть успадковуватися, а всі дії з таблицями виконуються за допомогою об'єктивно орієнтованих функцій. Як і в MySQL всі дані зберігаються на диску, в спеціально відсортованих файлах, але структура цих файлів і записів в них

	методами зчитування.	дуже сильно відрізняється.
Стандарт SQL	MySQL підтримує далеко не всі нові можливості стандарту SQL. Розробники вибрали саме цей шлях розвитку, щоб зберегти MySQL простим для використання. Компанія намагається відповідати стандартам, але не на шкоду простоті. Якщо якась можливість може поліпшити зручність, то розробники можуть реалізувати її у вигляді свого розширення не звертаючи уваги на стандарт.	Postgresql - це проект з відкритим вихідним кодом, він розробляється командою ентузіастів, і розробники намагаються максимально відповідати стандарту SQL і реалізують всі найновіші стандарти. Але все це призводить до збитків простоти. Postgresql дуже складний і через це він не настільки популярний як MySQL.
Можливості обробки	При виконанні запиту MySQL завантажує весь відповідь сервера в пам'ять клієнта, при великих обсягах даних це може бути не зовсім зручно. В основному за функціями Postgresql перевершує Mysql, далі	Postgresql підтримує використання курсорів для переміщення по отриманим даним. Ви отримуєте тільки покажчик, весь відповідь зберігається в пам'яті сервера баз даних. Цей покажчик можна зберігати

	розглянемо в яких саме.	між сеансами. Тут підтримується побудова індексів відразу для декількох стовпців таблиці. Крім того, індекси можуть бути різних типів, крім hash і b-tree доступні GiST і SP-GiST для роботи з містами, GIN для пошуку по тексту, BRIN і Bloom. Postgresql підтримує регулярні вирази в запитах, рекурсивних запитів і успадкування таблиць. Але тут є кілька обмежень, наприклад, ви можете додати нове поле тільки в кінець таблиці.
Продуктивність	У більшості випадків для організації роботи з базою даних в MySQL використовується таблиця InnoDB, ця таблиця представляє з себе B-дерево з індексами. Індекси дозволяють дуже	Вся заголовна інформація таблиць Postgresql знаходиться в оперативній пам'яті. Ви не можете створити таблицю, яка буде не в пам'яті. Записи таблиці упорядковано відповідно до індексу, а

	швидко отримати дані з диска, і для цього буде потрібно менше дискових операцій. Але сканування дерева вимагає знаходження двох індексів, а це вже повільно. Все це означає що MySQL буде швидше Postgresql тільки при використанні первинного ключа.	тому ви можете їх дуже швидко витягти. Для більшої зручності ви можете застосовувати кілька індексів до однієї таблиці.

В цілому PostgreSQL [3] працює швидше, за виключенням використання первинних ключів, тому було обрано як базу сервера PostgreSQL.

ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Кожна програма починається з репозиторія. Для потреб системи та зручності було обрано Gitlab та створено на ньому групу Vertica Platform, а в ній два репозиторія vertica_backend, vertica_mobile.

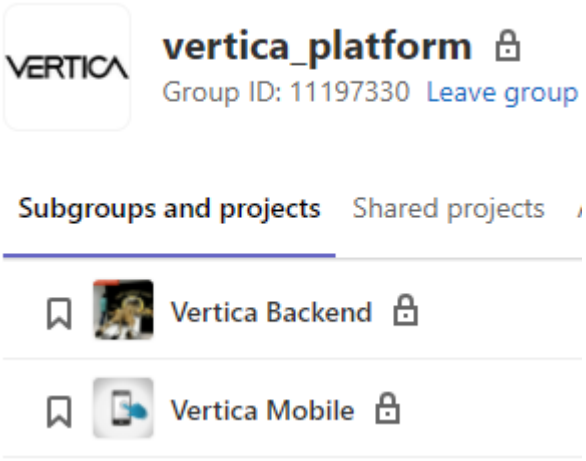

Рисунок 1 — група на Gitlab

5.1 Створення серверу та його налаштування

Перед створенням серверу було розроблено діаграму деплоя усієї системи.

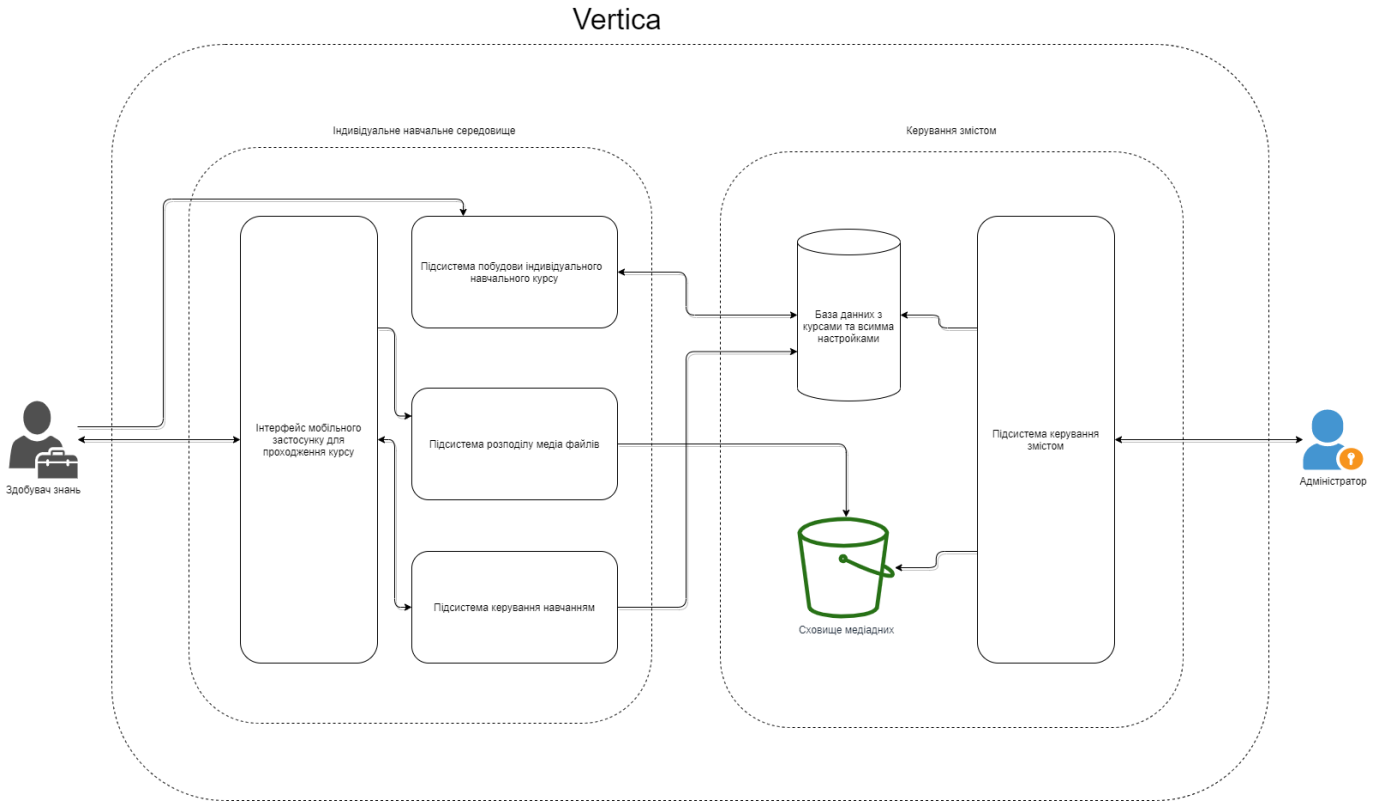

Рисунок 2 — Діаграма деплоя Vertica

З діаграми слідує, що нам потрібно створити одну віртуальну машину [5] на Google Cloud Platform, одну базу даних та один Cloud Storage

Для наших потреб створено сервер на Ubuntu 20.04 та встановлено на нього наступні пакети:

- certbot
- docker
- docker-compose

Certbot використовується для генерації сертифікатів для доменного імені. За допомогою наступної команди буде згенеровано сертифікати для nginx:

```
sudo certbot certonly -d api.vertica.yourcofounder.com.ua --standalone
--email def_fake@yahoo.com
```

Докер - ядро нашого CI/CD pipeline, за допомогою можливості створювати.
Докер композ - технологія, яка дозволяє зручно настроювати декілька контейнерів.
Після цього було створено сервер на GCP

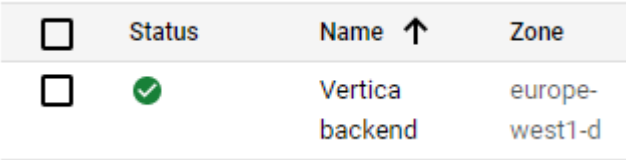

Рисунок 3 — сервер на GCP

Після створення віртуальної машини потрібно підв'язувати її IP адресу з доменом для API.

Рисунок 4 — підв’язка айпі адреси до домену API

Наступним кроком буде налаштування CI/CD системи для того, щоб нововведення після пушу в репозиторій автоматично потрапляли на сервер. Коротко наш CI/CD pipeline можна описати так:

1. Відбувається пуш [9] у гілку qa.
2. Відбувається білд докер імажу [8] де знаходиться Django (Встановлюються всі пайтон пакети, запускаються міграції, з допомогою volumes не втрачаємо завантажені ка)
3. Імаж потряпляє в докер репозиторій [8].
4. Після цього CI/CD заходить на сервер та логіниться у докер репозиторій для пулу попередньо збілдженого імажу.
5. Після пула підіймаються такі контейнери:
    a. vertica_backend - Django.

b. proxy - nginx який працює у режимі реверс-проксі та захищає з’єднання по стантдарту HTTPS.

6. Гітлаб записує статус пайплайну.

Для налаштування CI/CD потрібно встановити докер та докер компоуз слідуючи пунктам з офіційної документації. Після встановлення треба описати Gitlab CI config (де буде описаний алгоритм вище), докер файл (для того щоб встановити потрібні пайтон пакети), докер компоуз файл (для того щоб поряд з сервером джанго піднімати nginx). Усі ці описані конфіги можна знайти в репозиторії в папці deploy.

5.2 Діаграма сутностей (ERD)

Перед переходом до програмування потрібно спочатку розробити архітектуру системи. Першим кроком є CRC-аналіз (CRC-карта (Class-responsibility-collaboration card; з англ. - «карта« Клас-Відповідальність-Кооперація »») - метод мозкового штурму, призначений для проектування об'єктно-орієнтованого програмного забезпечення) предметної області, де було виділено сутності з якими буде працювати бекенд. На основі списку виділених сутностей було побудовано діаграму залежностей на основі якої будуть будуватися Джанго моделі та таблиці в Постгрес.

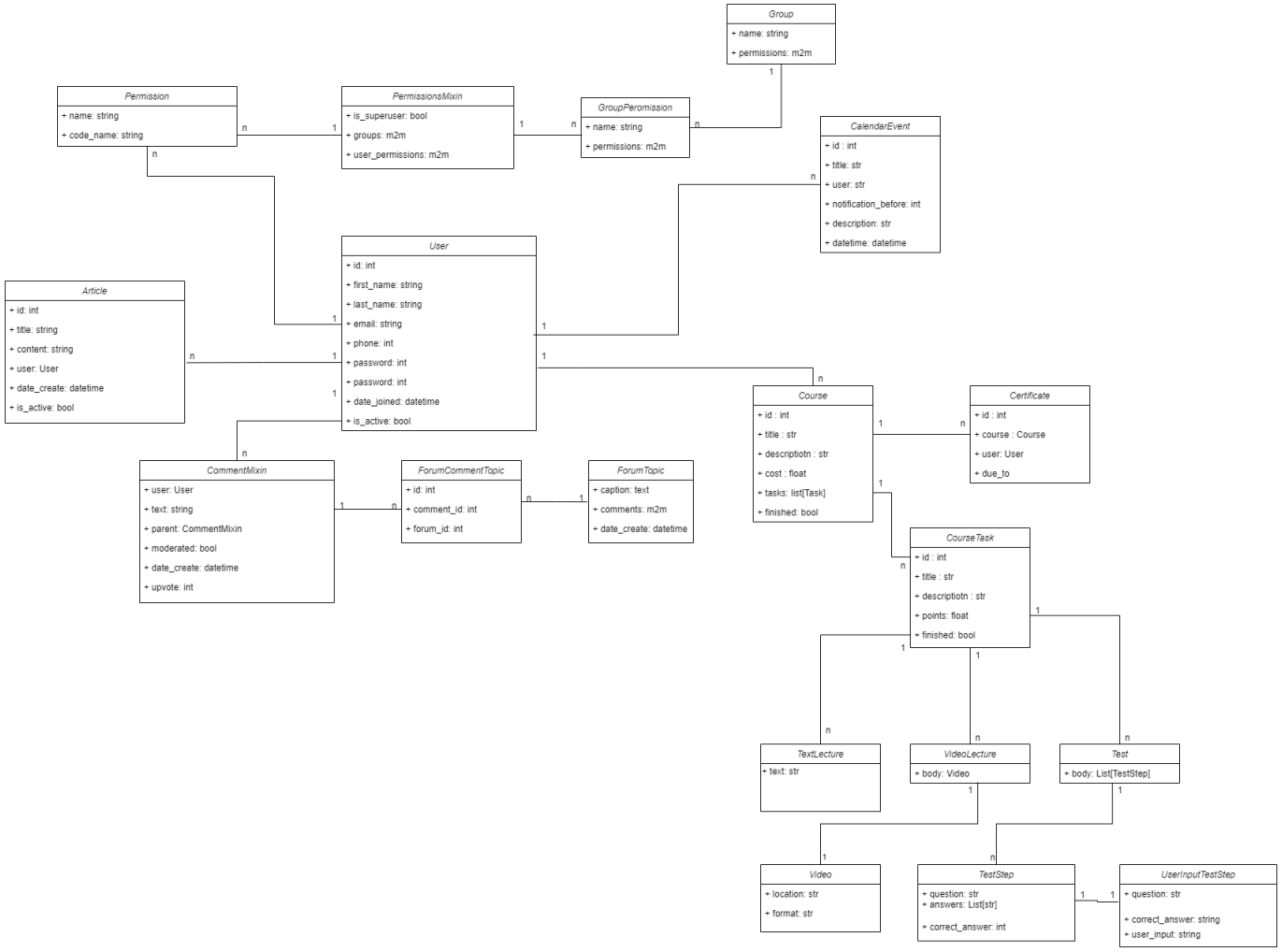

Рисунок 5 – Діаграма залежностей між сутностями системи

На даній діаграмі чітко видно, що у системі є запис про курс, та окремі записи про задачі з нього. Коли юзер запишеться на курс, буде створено окремий запис з

посиланням на курс, на який юзер був записаний, а також повинні автоматично створитись записи про задачі з цього курса.

На основі діаграми можна вже зробити висновки та мати загальне уявлення як працює система.

5.3 Діаграма класів (classes diagram)

Після реалізації Entity relation diagram вже можно приступати до діаграми класів, де розглянемо роботу системи вже більш детально.

Діаграма класів повинна бути побудована так, щоб код міг пропрацьовувати наступні сценарії:

— підписка на курс з створенням запису про сам курс та кожну його задачу, а також у всіх задач та курсу повинен бути статус “Не розпочато”;

— юзер повинен мати можливість змінити статус задачі;

— коли всі задачі в статусі “Зроблено” повинен створюватися сертифікат.

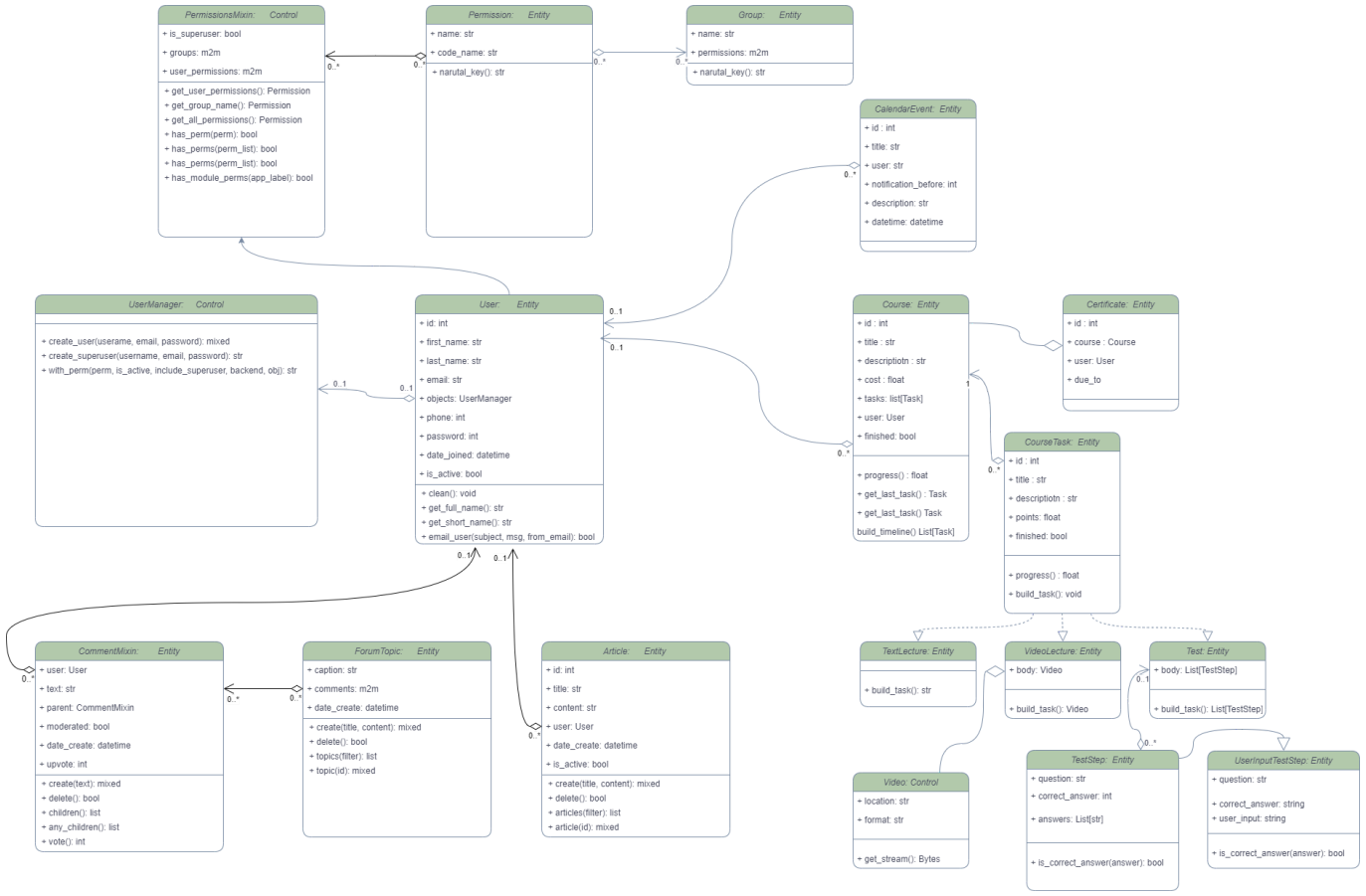

Рисунок 6 — Діаграма класів

Діаграму класів було трохи спрощено, бо буде використано звичайну структуру Джанго та нема сенсу описувати як використовувати ці класи, щоб не повторювати документацію.

Після діаграми класів можна переходити вже до написання коду.

5.4 Підготовка до розробки фронтенду мобільного застосунку

Для мене розробка мобільного застосунку це щось нове, тому я не знаю як це роблять у компаніях. Але в мене є досвід у backend, а також data engineering і точно можна сказати, що перед розробкою треба побудувати діаграми.

Я не знаю які діаграми треба будувати для фронтенду, тому вирішив звернутись до книги Responsive Web Design / Ethan Markot. Автор рекомендує приступати до розробки в декілька етапів [11]:

1. Побудувати будь-яку діаграму сценаріїв. Я дозволю собі знехтувати цим кроком, бо сценаріїв у системі буде не багато.

2. Побудувати UX flow діаграму - просту діаграму, що описує кроки, які користувач повинен зробити з вашим продуктом або послугою для досягнення мети. На цьому кроці я й опишу усі сценарії роботи з застосунком.

3. Побудувати UI flow діаграму - у фронтенд розробці UI flow діаграму використовують як wireframe для наступного кроку [11], де буде вже розмальовуватися детальний дизайн. В цілому цю діаграму можна описати, як зв’язок між екранами, з якого екрана на який юзер може попасти, які елементи повинні бути на екрані.

4. Розробити точний дизайн на намалювати мокапи. Цим кроком я також хочу знехтувати, бо мені не подобається малювати. Я вирішив обрати просту кольорову гамму (жовтий - найрадісніший з кольорів, білий - стандартний вибір, чорній - також стандартний вибір) та розмалювати вже по ходу розробки в ці кольори елементи з UI flow діаграми. Хочу виділити, що хоч я й назвав три кольори в дизайні білий і чорний часто не беруть до уваги коли говорять о кольоровій гамі, тобто в мене є тільки жовтий, а чим менше кольорів, тим простіше розробити непоганий дизайн.

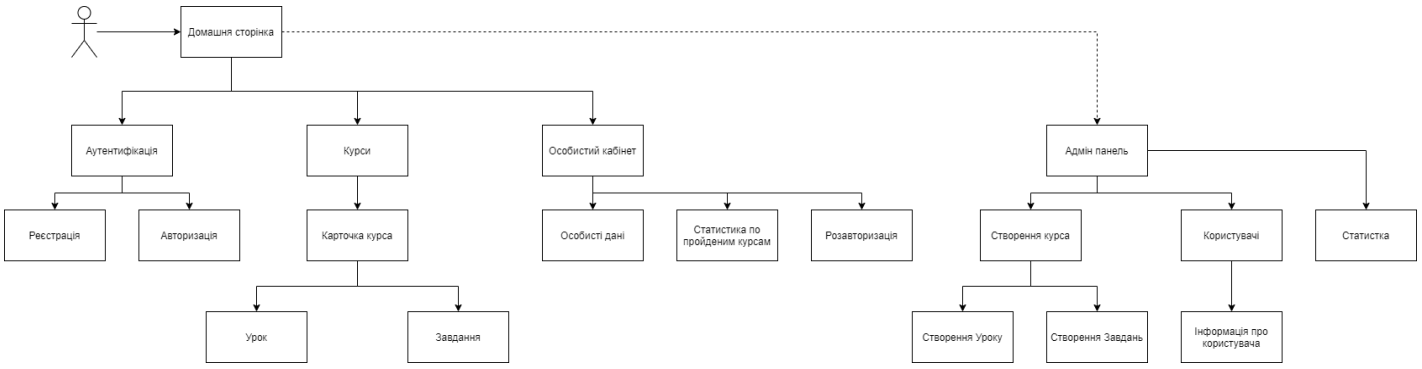

Рисунок 7 – UX flow діаграма

На діаграмі з рисунка 7 також відображено частина робочого процесу для адміністратора, яка буде відбуватися з настроєної адмін панель в Джанго. До описаних раніше сценаріїв додалося ще 3 - логін, реєстрація, логаут.

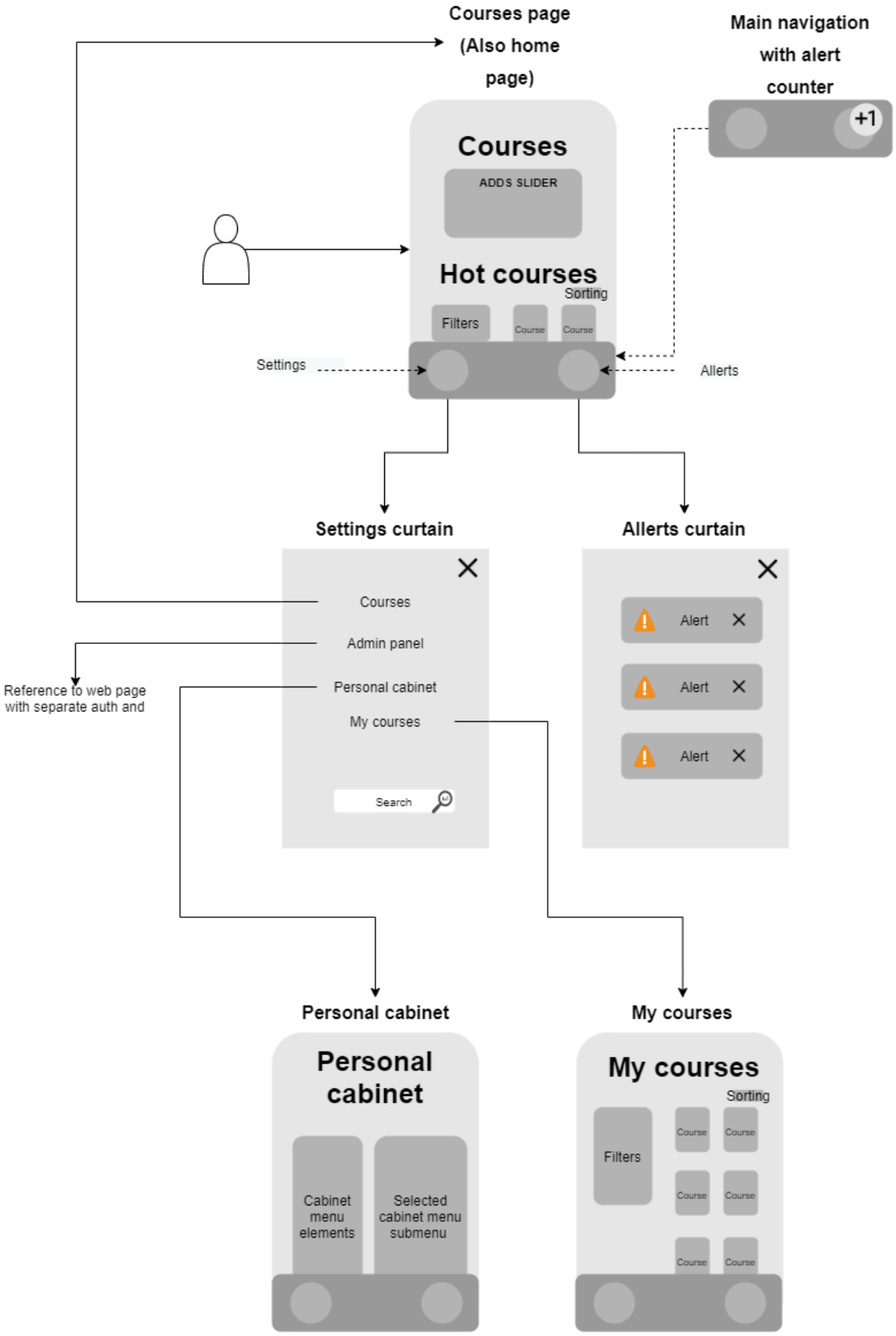

Рисунок 8 — UI flow діаграма

На UI флоу діаграмі видно, що перша версія застосунку буде дуже проста на екранів 6-8.

5.5 Конструювання та реалізація програмного продукту

Написання коду, коли готові всі діаграми готові це діло техніки, але як було написано в книзі “Довершений код”: конструювання - процес з багатьма помилками, бо хороше рішення часто відрізняється від поганого невеликою деталлю [10].

Коротко про етапи написання коду до цього проекту:

1. Описано Джанго моделі [1] по ERD.

2. Описано Джанго рест фреймворк серіалайзери [2] - завдяки їм налаштовується відповідь від сервера на запити. Наприклад, коли потрібно повернути не всі поля моделі, чи коли потрібно повернути тільки айді запису що був створений - вся ця логіка інкапсулюється у серіалайзера.

3. Описано Джанго вью класи - Джанго вью це звичайні контролери з архітектури MVC. Вью класи інкапсулюють логіку виклику функцій моделей тощо на обробку запиту та передають відповідь у серіалайзери, які вже формують HTTP відповідь.

4. Описано роути [1] - використовуються для того щоб сервер зрозумів що від нього хоче користувач та що потрібно викликати, щоб задовольнити його вимогу.

5. Описано Джанго сигнали - використовується патерн “Спостерігач”, завдяки цьому можливо додати автоматичні дії на деякі події. Було створено такі автоматичні події як:

a. Створення сертифікату для юзера коли всі задачі в статусі “Зроблено”.
b. Статус “Завершено” у курса ставиться автоматично, коли пройдено всі задачі.
c. Коли юзер записується на курс автоматично створюються усі задачі з курсу для юзера.
d. Всім новостворениим задачам ставиться статус “Не розпочато”.

e. Коли створюється задача до курс, то вона автоматично додається всім підписаним юзерам та створюється сповіщення.

f. Коли юзер завершує курс створюється статистика по курсу для юзера та для адмінів.

6. Застосовано пакет django-swagger [14] для генерації swagger документації ендпоінтів.

Рисунок 19 - Swagger документація

7. Створено мобільний застосунок на Flutter згідно діаграм та робочих процесів системи.

6. РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Приклади роботи користувача з системою будуть демонструватися на скріншотах емульованого андроїд девайсу Google Pixel 3XL з версією андроїд 11 у IntellijIDEA.

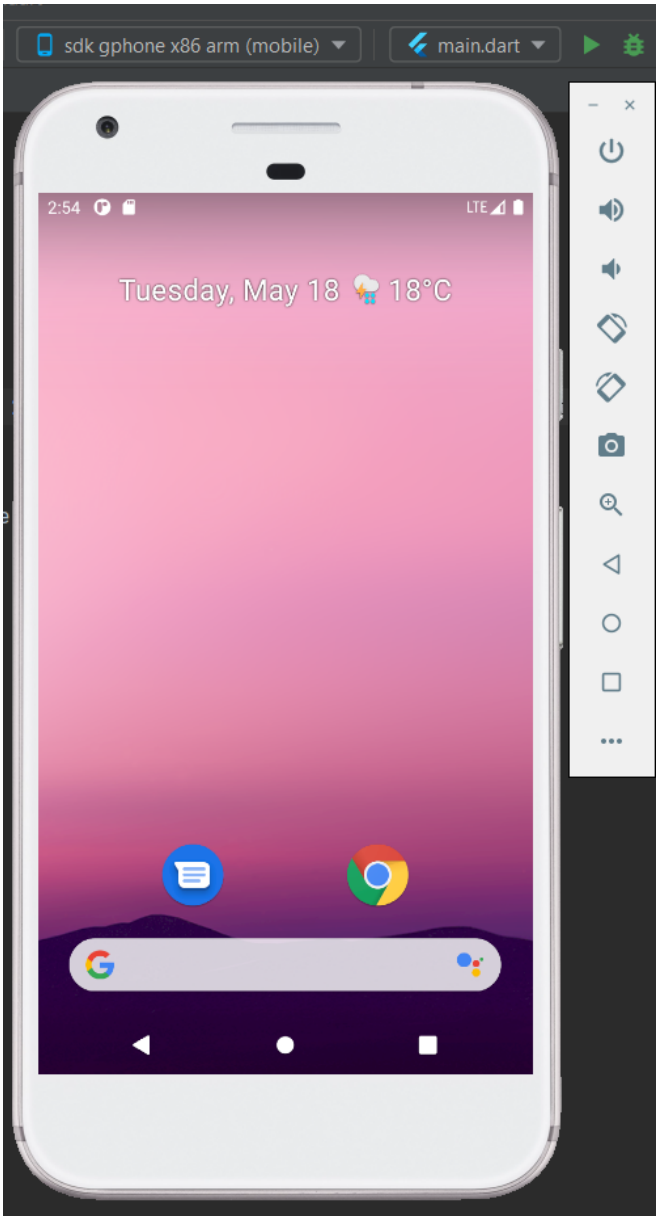

Рисунок 9 — Емульований андроїд девайс

Для того щоб встановити додаток потрібно скачати файл інсталяції на телефон та натиснути на нього. Після інсталяції з’явиться значок застосунку на робочому столі. Юзер натискає на цей значок та потрапляє на сторінку авторизації.

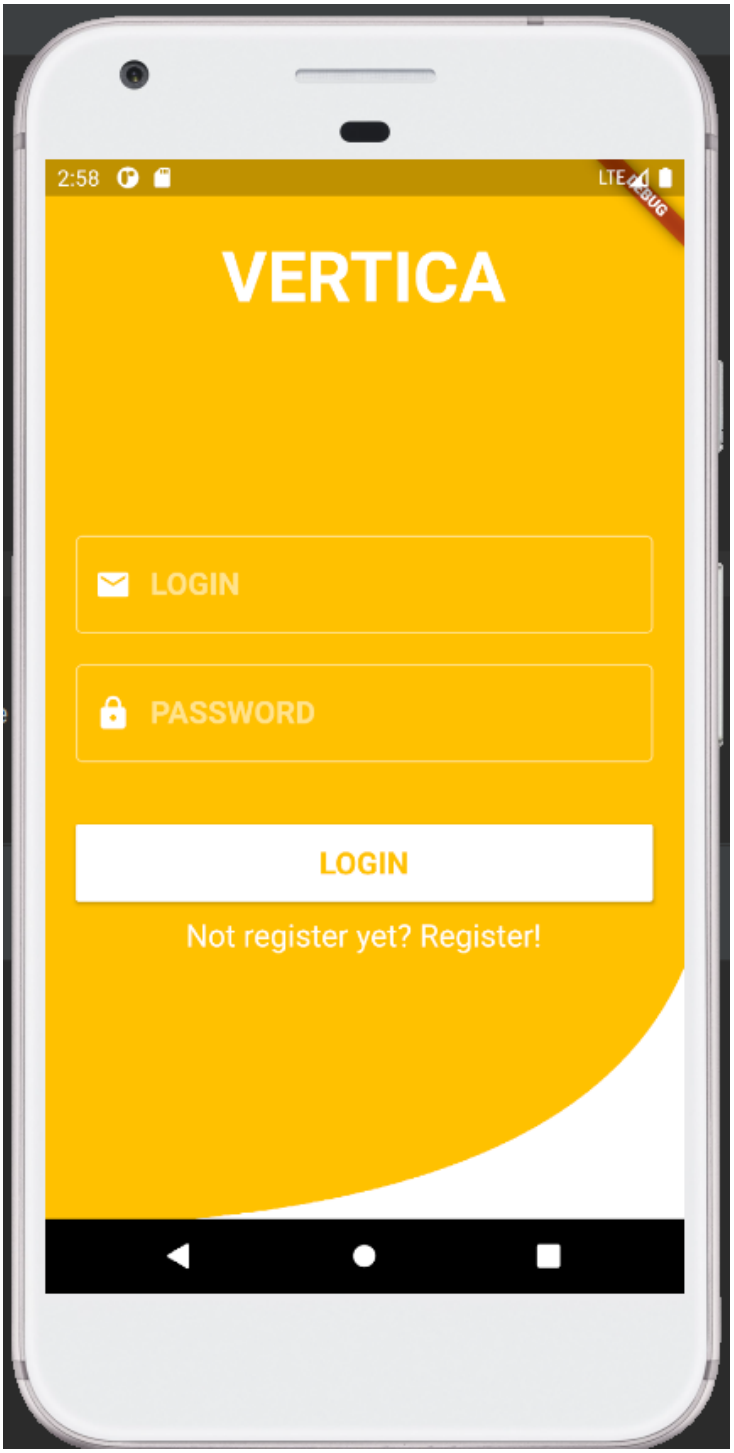

Рисунок 10 – Екран авторизації

Після успішної авторизації юзер потрапляє на сторінку курсів та може обрати курс для запису.

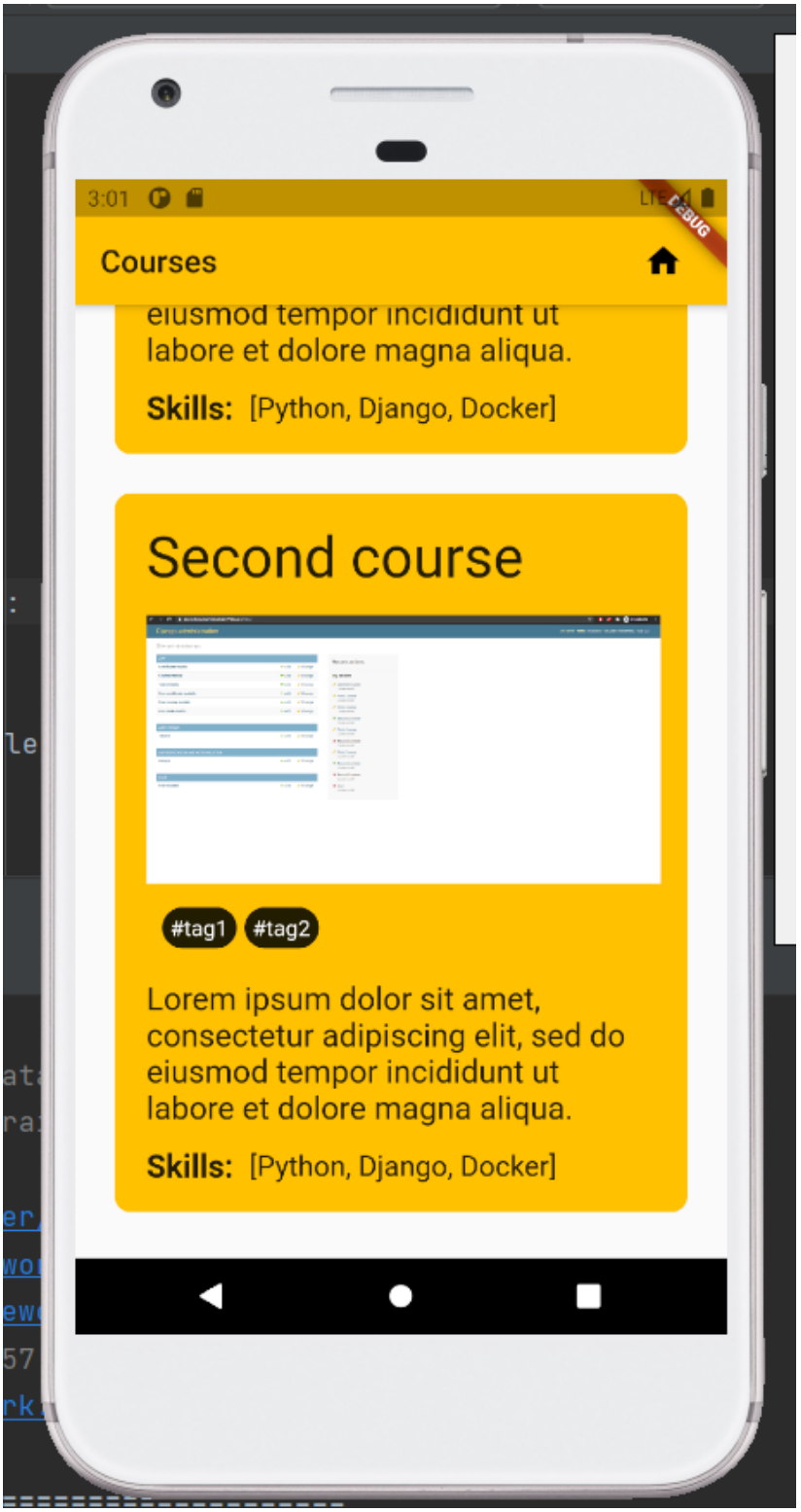

Рисунок 11 – Список курсів

Юзер може натиснути на іконку дому та потрапити до особистого кабінету, де знаходиться меню навігації, кнопка розавторизації та коротка статистика.

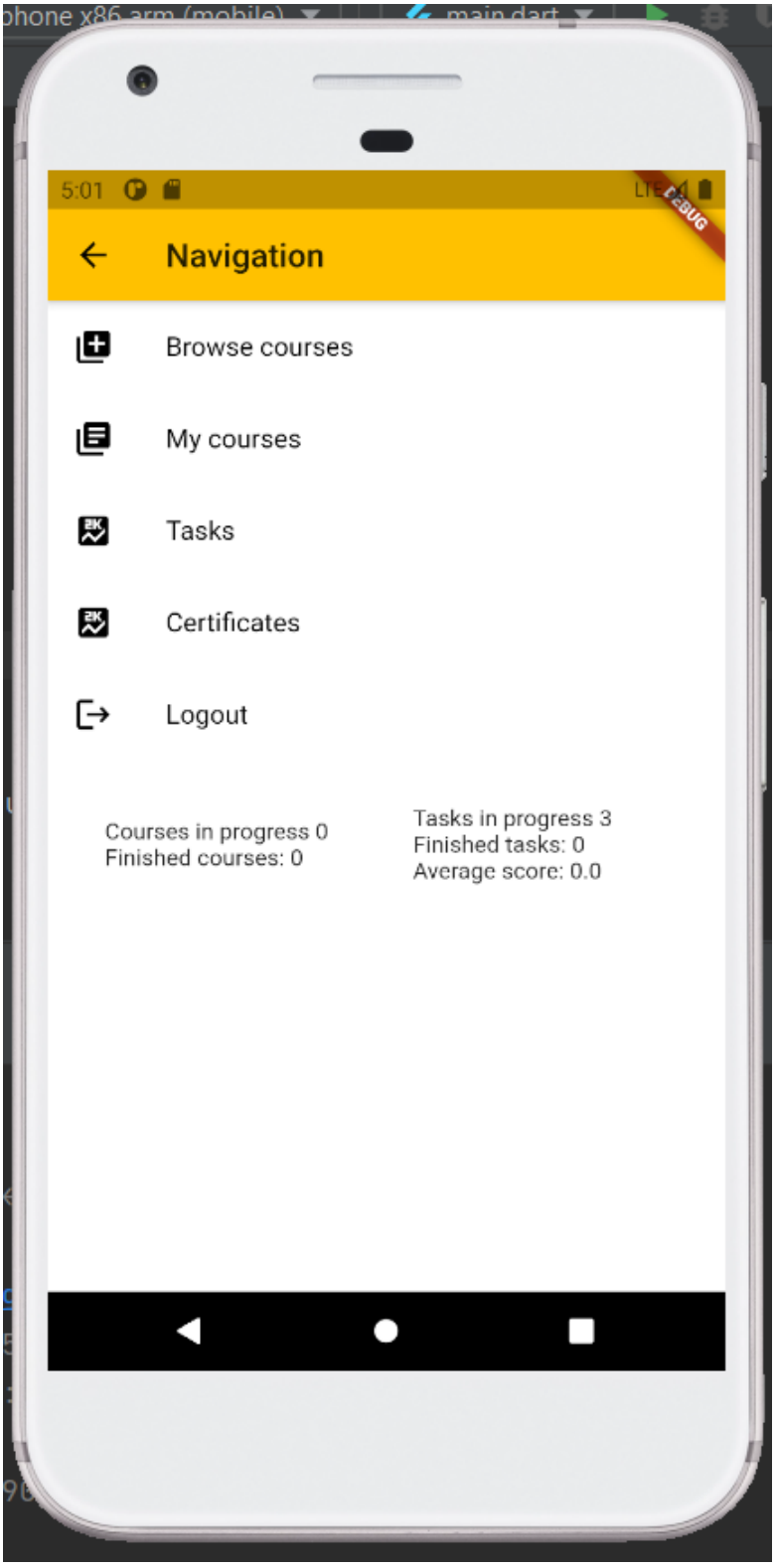

Рисунок 12 – Особистий кабінет

Якщо юзер натискає на курс у списку доступних курсів, то потрапляє на сторінку курсу та може ознайомитися з навичками, які прокачує курс, його описом, та кнопкою “Enroll” (записатися).

Рисунок 13 – Картка доступного курсу

Після запису на курс юзер потрапляє на картку курсу, де видно задачі з цього курса. Всі курси, на які юзер записався можна знайти в особистому кабінеті “My courses”.

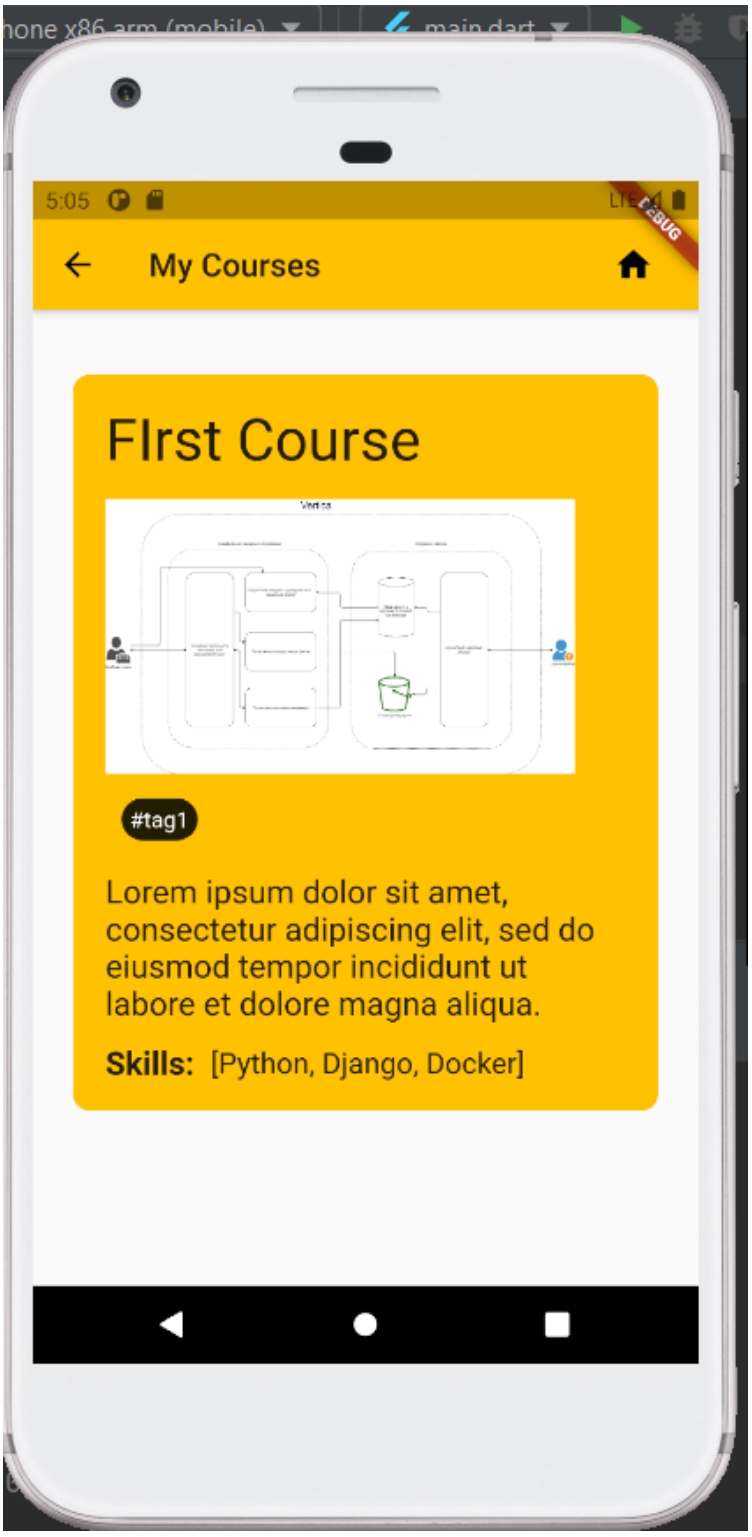

Рисунок 14 – Список курсів, на які юзер записався

6.1 Робота адміністратора з системою

Адмін-панель автоматично генерується на основі моделей Django. Було покращено вигляд автогенерації та додано декілька функцій для нашої задачі.

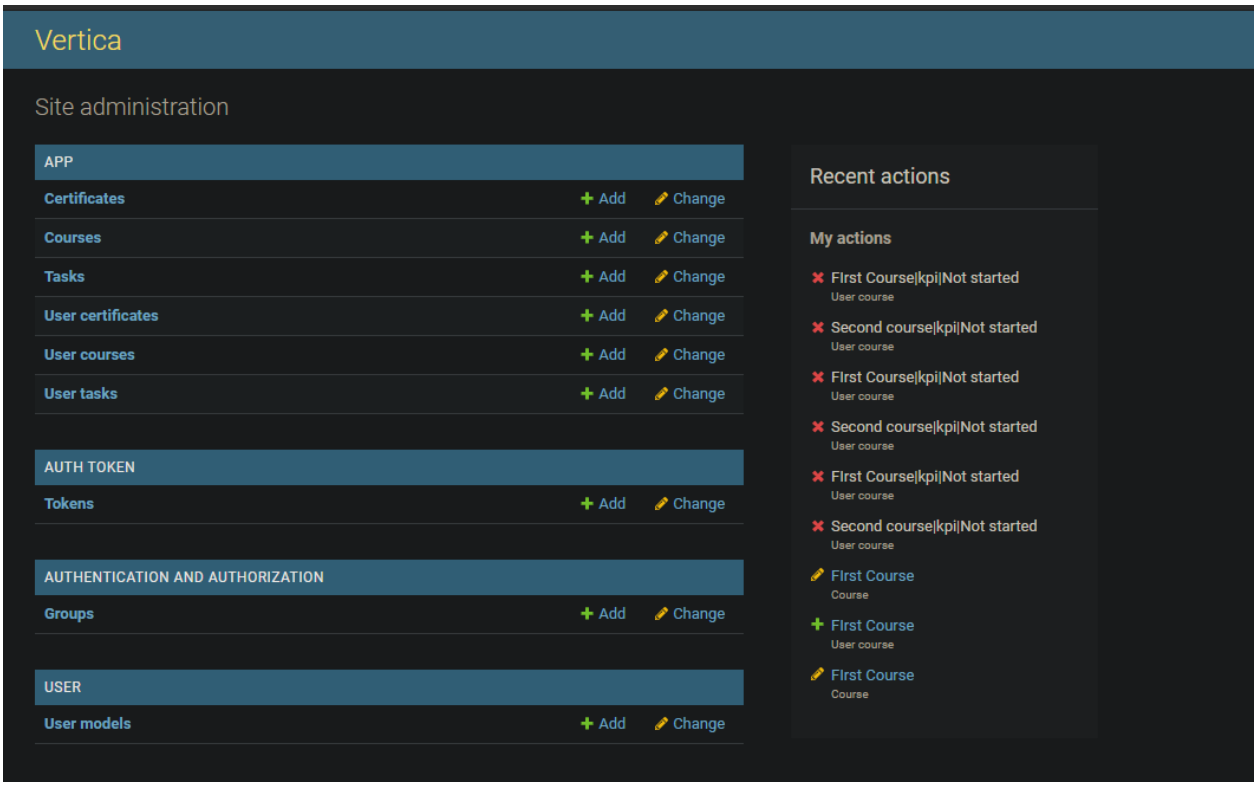

Рисунок 15 – Адмін-панель

В адмін-панелі юзер, який має право заходити в адмін-панель, має доступ до всіх сутностей системи. Юзер може подивитися на створені курси.

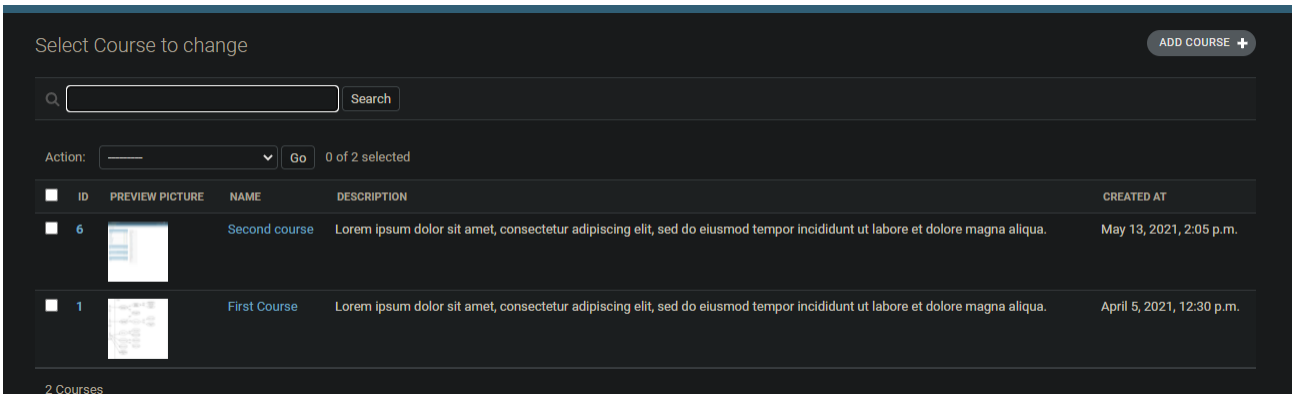

Рисунок 16 – Список курсів

На вкладці списку курсів користувач може обрати курс, для редагування, та змінити опис курсу, список навичок, список тегів, картинку для прев’ю, опис курсу, також було додано RichTextEditor для надання можливостей розмітки як у Google Docs.

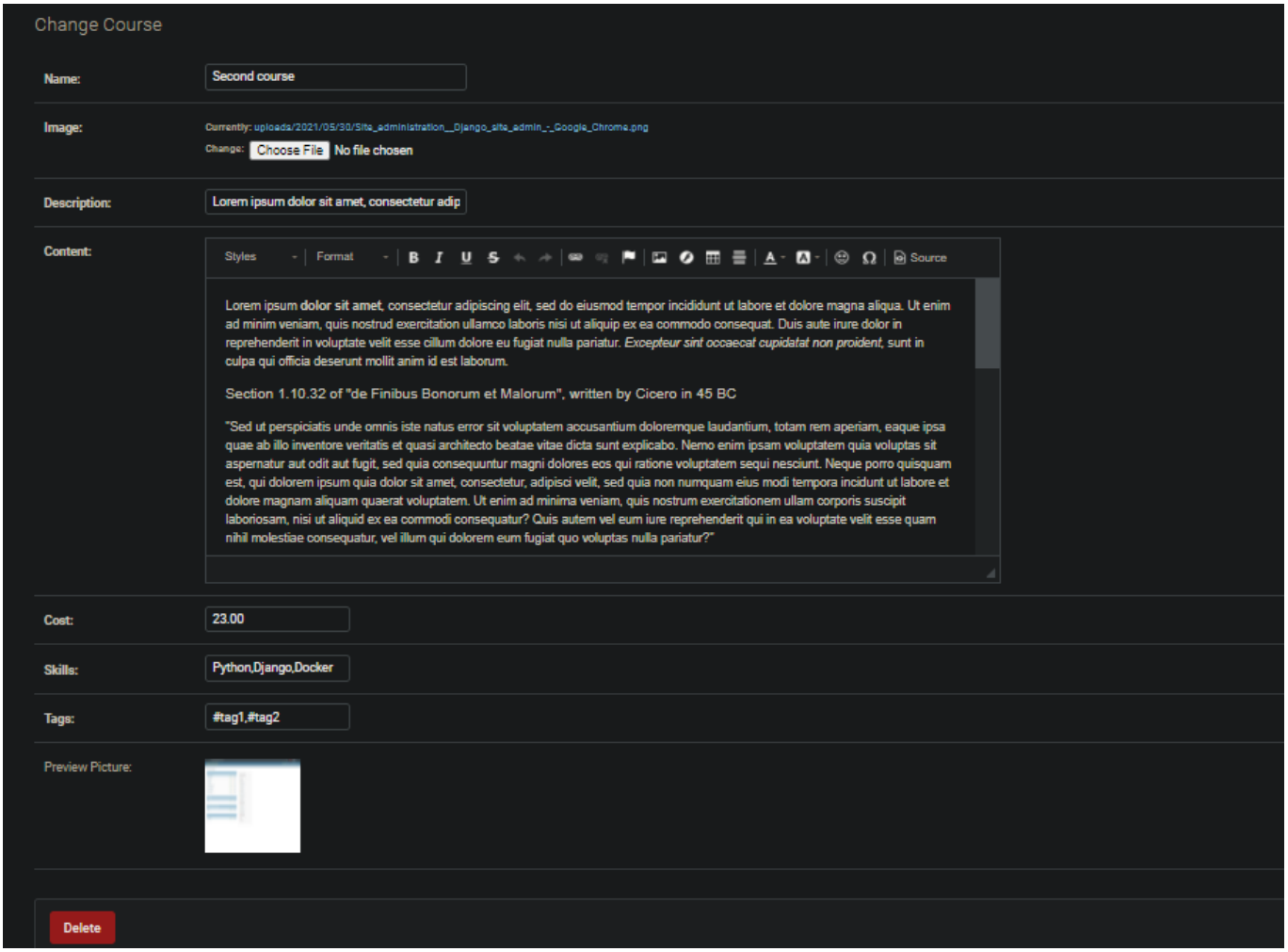

Рисунок 17 – Редагування курсу

2. ВИСНОВКИ

Під час роботи над застосунком було засвоєно багато нової інформації та отримано новий досвід у розробці повного циклу кросплатформенного мобільного застосунку. Було проаналізовано актуальні тренди у мобільний розробці і обрано найкращу технологію під нашу проблему. Обрано оптимальний варіант розробки фронтенду мобільного застосунку, надбано навичок у будуванні UI, UX flow діаграм. В результаті виконання дипломної роботи була вирішена задача розробки мобільного застосунку, який не є заміною офлайн освіти, а доповнює її та дозволяє у екстрених ситуаціях зробити навчання доступнішим (наприклад у ковід). Задача є актуальною, оскільки після введення карантину розпочався бум в освіті, мобільний трафік перевищує десктопний з 2013 року та з кожним роком його частка збільшується.

Було досліджено існуючі підходи які використовують великі онлайн платформи та запроваджено найкращі їх напрацювання ,щоб зробити офлайн розробку ефективною.

В результаті роботи була збудована система CI/CD, яка дозволяє зручно працювати над нововведеннями до API серверу, бо усі коміти миттєво доставляються і розгортаються на сервері, було розроблено API сервер, мобільний застосунок.

Для демонстрації роботи системи було розроблено програмний продукт, що надає наступні можливості:

— авторизуватися;

— ознайомитися з доступними курсами;

— обрати курс, та підписатися нього;

— виконати завдання курсу;

Результати роботи демонструють що система працює та може швидко відповідати на потреби нововведень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Django [Електронний ресурс] – https://docs.djangoproject.com/en/3.2/.

2. Документація Django Rest Framework [Електронний ресурс] – https://www.django-rest-framework.org/.

3. Документація PostgreSQL [Електронний ресурс] – https://www.postgresql.org/docs/.

4. Документація Google Cloud Platform [Електронний ресурс] – https://cloud.google.com/docs.

5. Документація Google Cloud Platform Virtual Machines [Електронний ресурс] – https://cloud.google.com/compute/docs/instances.

6. Документація Google Cloud SQL [Електронний ресурс] – https://cloud.google.com/sql/docs.

7. Документація GCP Firewall [Електронний ресурс] – https://cloud.google.com/vpc/docs/firewalls.

8. Документація Docker [Електронний ресурс] – https://docs.docker.com/get-started/.

9. Документація Gitlab [Електронний ресурс] – https://docs.gitlab.com/.

10. Steve McConnell : Perfect Code / Steve McConnell – Los Santos, USA: GTA 806 с.

11. Ethan Marko : Responsive Web Design / Ethan Markot, 2011– 455 с.

12. Flutter Complete Reference : Create beautiful, fast and native apps for any device / Alberto Miola, 2020 – 766 с.

13. Burd B. Flutter For Dummies / Barry Burd. – New York, United States: John Wiley & Sons Inc, 2020. – 384 с.

14. Alessandria S. A practical, project-based guide to building real-world cross-platform mobile applications / Simone Alessandria. – Birmingham, United Kingdom: Packt Publishing Limited, 2020. – 490 с..

15. Matthes E. Python Crash Course (2nd Edition) : A Hands-On, Project-Based Introduction to Programming / Eric Matthes. – San Francisco, United States: No Starch Press, US, 9. – 544 c. – (2nd Edition).

16. Martin R. Clean Code : A Handbook of Agile Software Craftsmanship / Robert Martin. – Upper Saddle River, NJ, United States: Pearson Education (US), 2009. – 464 c.

17. Orosz G. Building Mobile Apps at Scale : 39 Engineering Challenges / Gergely Orosz. – Netherlands: Pragmatic Engineer B.V, 2021. – 238 c.

18. Martin R. Clean Architecture : A Craftsman's Guide to Software Structure and Design / Robert Martin. – United States: Pearson Education (US), 2017. – 432 c.

19. Thomas D. The Pragmatic Programmer : your journey to mastery, 20th Anniversary Edition / D. Thomas, A. Hunt. – Boston, United States: Pearson Education (US), 2020. – 352 c.

20. McConnell S. Code Complete / Steve McConnell. – Redmond, United States: Microsoft Press,U.S., 2004. – 960 c.

ДОДАТОК А

Мобільний додаток для проходження дистанційних навчальних курсів з адаптивною системою формування змісту

Специфікація

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ71096_21А 1

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ71103_20Б 2	Записка Ворвуль Д. М. ТІ-71.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ71103_20Б 2-1	models.py	Модуль для роботи за базою даних
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ71103_20Б 2-2	admin.py	Модуль для роботи з адмін-панеллю

ДОДАТОК Б

Мобільний додаток для проходження дистанційних навчальних курсів з адаптивною системою формування змісту

Лістинг програми

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ71096_21Б

Аркушів 10

Київ 2021

Код адмін панелі.

```
from django.contrib import admin
from django.utils.safestring import mark_safe

from .models import (
    CourseModel,
    TaskModel,
    CertificateModel,
    UserCourseModel,
    UserTaskModel,
    UserCertificateModel,
)


class PreviewPictureShow:
    def get_preview_picture(self, obj):
        if obj.image:
            return mark_safe(f'<img src="{obj.image.url}" height="60",
width="60">')
        else:
            return "No image"

    def get_preview_picture_edit(self, obj):
        if obj.image:
            return mark_safe(f'<img src="{obj.image.url}"
height="100", width="100">')
        else:
            return "No image"

    get_preview_picture.short_description = "Preview Picture"
    get_preview_picture_edit.short_description = "Preview Picture"


@admin.register(CourseModel)
class CourseAdmin(admin.ModelAdmin, PreviewPictureShow):
    list_display = (
        "id",
        "get_preview_picture",
        "name",
        "description",
        "created_at",
    )
    list_display_links = ("name", "id")
    readonly_fields = ("get_preview_picture_edit",)
    search_fields = ("name",)


@admin.register(TaskModel)
class TaskAdmin(admin.ModelAdmin):
    list_display = (
        "id",
        "title",
```

```
        "course",
        "created_at",
    )
    list_display_links = ("title", "id")
    list_filter = ("course",)
    search_fields = ("title",)


@admin.register(CertificateModel)
class CertificateAdmin(admin.ModelAdmin):
    pass


@admin.register(UserCourseModel)
class UserCourseAdmin(admin.ModelAdmin):
    list_display = (
        "id",
        "user",
        "course",
        "status",
        "created_at",
    )
    list_display_links = ("user", "id")
    list_filter = ("course", "status", "user")
    search_fields = ("course", "user")


@admin.register(UserTaskModel)
class UserTaskAdmin(admin.ModelAdmin):
    list_display = (
        "id",
        "task",
        "user_course",
        "status",
    )
    list_display_links = ("user_course", "id")
    list_filter = ("user_course", "status", "task")
    search_fields = ("user_course", "task")


@admin.register(UserCertificateModel)
class UserCertificateAdmin(admin.ModelAdmin):
    pass


admin.site.site_title = "Vertica"
admin.site.site_header = "Vertica"
```

Код моделей веб серверу.

```
from ckeditor_uploader.fields import RichTextUploadingField
from django.db import models
from django.contrib.auth import get_user
from django.utils.translation import gettext_lazy as _
from app.user.models import UserModel
from django.contrib.postgres.fields import ArrayField


def empty_list():
    return []


class CourseModel(models.Model):
    name = models.CharField(max_length=250)
    image = models.ImageField(upload_to="uploads/%Y/%m/%d/")
    description = models.CharField(max_length=1000)
    content = RichTextUploadingField()
    cost = models.DecimalField(max_digits=6, decimal_places=2)
    skills = ArrayField(
        models.CharField(max_length=15, blank=True), size=10,
default=empty_list
    )
    tags = ArrayField(
        models.CharField(max_length=15, blank=True), size=10,
default=empty_list
    )
    updated_at = models.DateTimeField(auto_now=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.name

    @property
    def image_path(self):
        try:
            return self.image.url
        except:
            ""

    class Meta:
        verbose_name = "Course"
        verbose_name_plural = "Courses"


class TaskModel(models.Model):
    LECTURE = 1
    VIDEO_LECTURE = 2
    FREE_TASK = 3

    TYPES = (
        (LECTURE, _("Lecture")),
        (VIDEO_LECTURE, _("Video lecture")),
        (FREE_TASK, _("Task")),
```

```
    )

    course = models.ForeignKey(
        CourseModel, on_delete=models.CASCADE, related_name="tasks"
    )
    type = models.IntegerField(choices=TYPES, default=LECTURE)
    title = models.CharField(max_length=250)
    description = models.CharField(max_length=1000)
    content = RichTextUploadingField()
    max_points = models.IntegerField(default=10)
    video_link = models.URLField(blank=True, null=True)
    file = models.FileField(upload_to="uploads/%Y/%m/%d/", blank=True,
null=True)
    due_date = models.DateTimeField(blank=True, null=True)
    updated_at = models.DateTimeField(auto_now=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = "Task"
        verbose_name_plural = "Tasks"


class CertificateModel(models.Model):
    course = models.OneToOneField(
        CourseModel, on_delete=models.CASCADE,
related_name="certificate"
    )
    congratulations = models.CharField(max_length=1000)

    def __str__(self):
        return self.course.name

    class Meta:
        verbose_name = "Certificate"
        verbose_name_plural = "Certificates"


class UserCourseModel(models.Model):
    FINISHED = 0
    IN_PROGRESS = 1
    NOT_STARTED = 2

    STATUSES = (
        (FINISHED, _("Finished")),
        (IN_PROGRESS, _("In progress")),
        (NOT_STARTED, _("Not started")),
    )
    user = models.ForeignKey(
        UserModel, on_delete=models.CASCADE, related_name="courses"
    )
```

```
    course = models.ForeignKey(
        CourseModel, on_delete=models.CASCADE,
related_name="usercourses"
    )
    status = models.IntegerField(choices=STATUSES,
default=NOT_STARTED)
    updated_at = models.DateTimeField(auto_now=True)
    created_at = models.DateTimeField(auto_now_add=True)

    @property
    def full_name(self):
        return f"{self.user.first_name} {self.user.last_name}"

    def __str__(self):
        return " | ".join(
            [self.course.name, self.user.username,
str(self.STATUSES[self.status][1])]
        )

    class Meta:
        verbose_name = "User course"
        verbose_name_plural = "User courses"


class UserTaskModel(models.Model):
    FINISHED = 0
    IN_PROGRESS = 1
    NOT_STARTED = 2

    STATUSES = (
        (FINISHED, _("Finished")),
        (IN_PROGRESS, _("In progress")),
        (NOT_STARTED, _("Not started")),
    )

    user_course = models.ForeignKey(
        UserCourseModel, on_delete=models.CASCADE,
related_name="user_tasks"
    )
    task = models.ForeignKey(
        TaskModel, on_delete=models.CASCADE,
related_name="users_tasks"
    )
    status = models.IntegerField(choices=STATUSES,
default=NOT_STARTED)
    points = models.IntegerField(default=0)

    @property
    def full_name(self):
        return f"{self.user_course.user.first_name}
{self.user_course.user.last_name}"

    @property
```

```
    def course_name(self):
        return self.user_course.course.name

    def __str__(self):
        return self.task.title

    class Meta:
        verbose_name = "User task"
        verbose_name_plural = "User tasks"


class UserCertificateModel(models.Model):
    certificate = models.ForeignKey(
        CertificateModel, on_delete=models.CASCADE,
related_name="certificate"
    )
    user_course = models.ForeignKey(
        UserCourseModel, on_delete=models.CASCADE,
related_name="user_certificates"
    )
    file = models.FileField(upload_to='certificates/')
    uuid = models.CharField(max_length=35)
    created_at = models.DateTimeField(auto_now_add=True)

    @property
    def full_name(self):
        return f"{self.user_course.user.first_name}
{self.user_course.user.last_name}"

    @property
    def congratulations(self):
        return self.certificate.congratulations

    @property
    def course_name(self):
        return self.user_course.course.name

    def __str__(self):
        return self.congratulations

    class Meta:
        verbose_name = "User certificate"
        verbose_name_plural = "User certificates"
```

Код сигналів для автоматичних подій, наприклад автоматичного створення задач для підписників курсу, коли було додано нове завдання.

```
from uuid import uuid4
from django.db.models.signals import post_save, pre_save
```

```
from .models import (
    UserCourseModel,
    TaskModel,
    UserTaskModel,
    UserCertificateModel,
    CertificateModel,
)


def save_user_course(sender, instance: UserCourseModel, created,
**kwargs):
    if created:
        for task in instance.course.tasks.all():
            ut = UserTaskModel(
                user_course=instance,
                task=task,
                status=UserTaskModel.NOT_STARTED,
                points=0,
            )
            ut.save()

    if (
            instance.status == UserCourseModel.FINISHED
            and not instance.user_certificates.first()
    ):
        user_cert = UserCertificateModel(

certificate=CertificateModel.objects.get(course=instance.course),
            user_course=instance,
            uuid=uuid4().hex,
        )
        user_cert.save()


def save_task(sender, instance: TaskModel, created, **kwargs):
    if created:
        for uc in instance.course.usercourses.all():
            ut = UserTaskModel(
                user_course=uc,
                task=instance,
                status=UserTaskModel.NOT_STARTED,
                points=0,
            )
            ut.save()


def save_user_task(sender, instance: UserTaskModel, created,
**kwargs):
    if not created:
        if instance.user_course.status == 2 and any(
                [user_task.status == 1 for user_task in
instance.user_course.user_tasks.all()]):
```

```
UserCourseModel.objects.filter(id=instance.user_course.id).update(stat
us=1)
        if instance.user_course.status == 1 and all(
                [user_task.status == 0 for user_task in
instance.user_course.user_tasks.all()]):

UserCourseModel.objects.filter(id=instance.user_course.id).update(stat
us=0)


post_save.connect(save_user_course, UserCourseModel)
post_save.connect(save_task, TaskModel)
post_save.connect(save_user_task, UserTaskModel)
```

ДОДАТОК В

Мобільний додаток для проходження дистанційних навчальних курсів з адаптивною системою формування змісту

Опис програмного коду

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ71096_21В 3

Аркушів 11

Київ – 2021

АНОТАЦІЯ

У роботі яка складається з 60 сторінок представлено діаграми, схеми збудованої системи, також проілюстровано скріншотами користувацький досвід, лістинг коду.

Головною метою цієї роботи було дослідити найкращі практики в онлайн освіті та створити мобільний застосунок для проходження курсів. На сьогоднішній день розвиток - це новий тренд, переважання мобільного трафіку над десктопним - це факт, тому програмний продукт та ідеї які з’явилися під час роботи над дипломною роботою являється актуальним та відображають сьогоденні реалії.

Перед розробкою було проведено аналіз технологій та підібрано найкращу комбінацію під нашу задачу: Django + DRF для вебсерверу, Flutter для розробки кросплатформенного мобільного застосунку, Також було було пропрацьовано інфраструктуру з боку розробника - створено CI/CD систему для автоматичної інтеграції нововведень на веб сервер.

Ключові слова: *ОСВІТА, НАВЧАННЯ, КУРСИ, ДИСТАНЦІЙНА ОСВІТА*.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку А міститься специфікація програмного забезпечення.

У додатку Б міститься лістинг коду найголовніших елементів розробленого програмного забезпечення.

У додатку В описано основні класи розробленого мобільного застосунку.

Програма може бути запущена на емуляції Android/IOS девайсу в будь якій IDE, яка підтримує Flutter розробку. Також можлива установка через файл інсталяції. Для успішного користування застосунком треба лише доступ в інтернет. Застосунок розроблено з використанням кросплатформеного фреймворка Flutter.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Мобільний застосунок для проходження курсів має наступні функції:

— створення курсу;

— створення задачі до курсу;

— редагування курсу;

— редагування задачі курсу;

— запис на курс;

— автоматичне створення сертифікату коли всі задачі курса завершено;

— автоматичне додавання задач учаснику курса;

— відображення помилок у разі якщо виникли якісь неполадки при відправленні запитів;

— авторизація та реєстрація;

— можливість використовувати RichTextEditor при редагуванні опису курсу;

— пройти лекцію;

— подивитися список курсів;

— подивитися картку задачі;

— подивитися картку курсу;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Архітектурно система влаштована з трьох підсистем:

— веб сервер;

— адмін-панель;

— мобільний застосунок;

Веб сервер опрацьовує запити користувачів з мобільних застосунків та запити з адмін-панелі. В адмін-панелі проводяться всі роботи з наповнення контентом курсів. Мобільний застосунок використовується для запису, проходження курсів, отримання сертифікатів за пройденний курс.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програма потребує для запуску смартфон на Android/IOS або емулятор цього девайсу та IDE з підтримкою Flutter. Також потрібен доступ до інтернету.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма потребує для запуску смартфон на Android/IOS або емулятор цього девайсу та IDE з підтримкою Flutter. Також потрібен доступ до інтернету.

ВХІДНІ ДАНІ

Вхідними даними для розробленої системи є курси у вигляді лекцій.

ВИХІДНІ ДАНІ

Вихідними даними роботи системи є сертифікат про успішне проходження курсу.