МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Проектування освітнього ресурсу для вивчення ІТ-технологій

Виконав студент групи К-18
спеціальності 122 «Комп'ютерні науки»
Джанмурадов Сердар _____

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант _____
_____

Рецензент к.техн.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2022

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІС – інформаційна система

ОС – операційна система

ПЗ – програмне забезпечення

СУБД –система управління базами даних

API – Application Programming Interface – програмний інтерфейс програми

IDE – Integrated Development Environment – інтегроване середовище розробки

SDK – Software Development Kit – набір засобів розробки

ВСТУП

Останнім часом молодь все більше уваги приділяє своєї освіті. Це може бути як класична університетська освіта так і додаткові курси, вивчення іноземних мов чи спеціальні курси, що надають додаткові навички, наприклад вивчення правил дорожнього руху. Все більше навчального матеріалу зявляється в мережі Інтернет. Особливості епідеміологічної ситуації в світі в останні роки привело до того, що дистанційна освіта стала майже головною формою навчання. Тому все більше зявляється веб-ресурсів, які надають можливість вивчати різні курси. Освітні ресурси можуть розміщувати різний контент: відео лекції, презентації, тести різних типів. Збірати та аналізувати статистику результатів навчання. Особливо актуальними подібні веб-ресурси є для сучасних вищих навчальних закладів України.

Метою кваліфікаційної роботи бакалавра є проектування освітнього ресурсу для вивчення ІТ-технологій.

Продукт призначається для всіх, хто бажає отримати нові знання в сфері ІТ або зміцнити їх, для тих, у кого є бажання вивчати щось нове, але немає грошей на платні курси або недостатньо часу для їх відвідування.

Для досягнення мети необхідно вирішити ряд завдань:

- Обгрунтувати вибір інструментів та технології для реалізації освітнього ресурсу;
- Виконати проектування інформаційної системи, створити основні діаграми, що її описують: діаграма прецедентів, даграма станів системи, діграма логічного представлення інформаційної системи, діаграма класів та ін.;
- Реалізувати систему;
- Провести тестування системи.

Структура кваліфікаційної роботи складається з вступу, трьох розділів, висновків, переліку посилань на 18 найменувань. Повний обсяг роботи становить 66 сторінок і містить 28 рисунків та 3 таблиці.

1 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ ОСВІТНЬОГО РЕСУРСУ

1.1Опис стеку технологій

При розробці усіх модулів ресурсу будуть використані такі технології:

− мова програмування Python [1], яка була обрана для використання через швидкість написання програмного коду та його зрозумілість при подальшій роботі, а також через значну кількість доступних до використання бібліотек, які реалізують різноманітну функціональність та позбавляють користувачів від необхідності створення власних програмних реалізацій;

− фреймворк Django [2], який дозволяє проводити створення веб-сайтів найбільш ефективно та швидко за рахунок можливість одразу виконувати додавання контенту на сайт за допомогою автоматично генерованої панелі адміністратору, що дозволяє з самого початку розробки побачити як контент відображається на кожній сторінці сайту;

− технологія віртуалізації Docker [3], яка дозволяє виконувати симуляцію незв’язності серверів для різноманітних СУБД та головного серверу ресурсу в окремих контейнерах, які у подальшому можна використовувати на справжніх серверах.

Розробка модулю користувальницького інтерфейсу планується проводити з використанням наступних технологій:

− мова гіпертекстової розмітки HTML [4], яка використовується для надання загальної структури веб-сторінок ресурсу;

− мова каскадних стилів CSS [5], яка використовується для визначення стилів різних елементів структури веб-сторінок;

− мова програмування JavaScript [6], яка використовується для реалізації частини функціоналу ресурсу, наприклад, використання технології AJAX для уникнення необхідності перезавантаження сторінки при незначній зміні даних.

Розробка модулю контролеру, який забезпечує взаємодію з усіма іншими компонентами ресурсу, буде проводитися за допомогою:

− модуль мови програмування Python django_postgres [7], який використовувався для доступу до даних у PostgreSQL та маніпуляції з ними, забезпечення найбільшої зручності використання разом з фреймворком Django;

− модуль Python elasticsearch_dsl [8], використання якого дає можливість доступу до даних у Elasticsearch та базової маніпуляції з ними;

− модуль мови програмування Python django_redis [9], який використовується для доступу до кешу сторінок у Redis та для маніпуляції з ним, а також забезпечення найбільшої зручності використання разом з фреймворком Django;

− модуль мови програмування Python redis [10], який використовується окремо від фреймворку Django для можливості самостійного доступу та маніпуляції з даними у базі;

− бібліотека мови програмування Python Scrapy [11], яка використовувалася для створення парсеру вакансій за рахунок максимальної зручності та простоти у використанні.

Розробка модулів зберігання даних проводилась з використанням наступних СУБД:

− СУБД PostgreSQL [12], яка використовується для зберігання пов’язаних між собою даних та даних, які потребують найбільшої безпеки при збереженні та маніпуляції;

− СУБД Elasticsearch [13], яка використовується для забезпечення повнотекстового пошуку за словами запиту через значно більшу швидкість роботи цієї функції, ніж у інших СУБД;

− СУБД Redis [14], яка використовувалась для зберігання кешу сторінок та даних, які потребувалися лише тимчасово або була важлива їх швидка обробка за рахунок збереження усієї інформації у оперативну пам'ять.

1.2 Вибір архітектурного шаблона та шаблонів проектування

Архітектурний шаблон – це шаблони програмного забезпечення (ПЗ), що являють собою звіт «належних практик» вирішення архітектурних проблем розробки програмного забезпечення. Розглянемо їх детальніше.

N-ярусний архітектурний шаблон допомагає провести структуризацію ПЗ, шляхом його розбиття на певну кількість абстрактних рівнів (ярусів), які у свою чергу виконують характерну для кожного рівня підмножину задач. Кількість ярусів необмежена, але зазвичай інформаційні системи в бізнес-застосунках часто використовують двоярусну архітектуру – більш відому під назвою клієнт-серверна архітектура (рис. 1.1).

Даний шаблон має такі переваги: повторне використання ярусів, підтримка стандартизації, залежності знаходяться локально, замінність. Недоліки: каскадне змінювання поведінки (зміна в одному ярусі приводить до зміни інших), низька ефективність, дублювання сервісів.

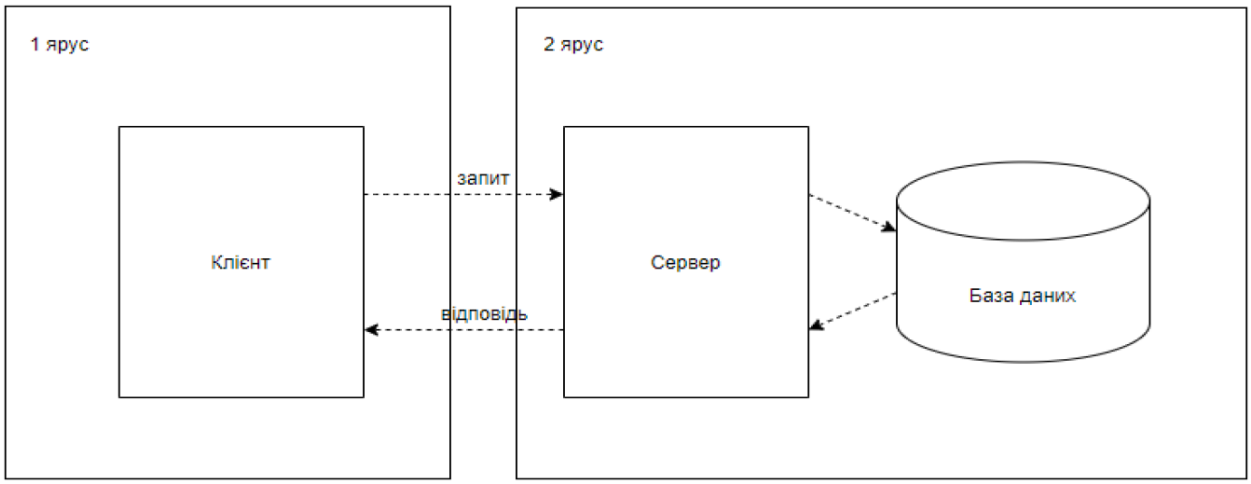

Рисунок 1.1 – Структура двоярусної архітектури

Архітектурний шаблон MVC передбачає поділ системи на три складові частини (рис. 1.2). Модель реалізує основні функціональні можливості та містить дані. Вид відображає інформацію для користувачів. Контролер

обробляє введені користувачем данні. Вид і контролер разом складають інтерфейс користувача, а механізм розповсюдження змін забезпечує узгодженість між інтерфейсом користувача та моделлю [15].

Переваги шаблона: синхронізованість даних, більшість фреймворків реалізують та використовують даний шаблон для створення ПЗ, багаточисленні види для однієї моделі. Недоліки шаблона: тісний зв'язок між видом і контролером, труднощі використання MVC з сучасними інструментами інтерфейсами користувача.

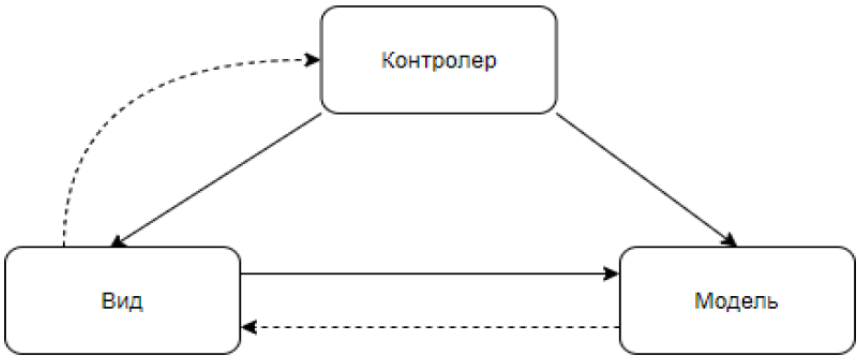

Рисунок 1.2 – Структура MVC

Архітектурний шаблон MVVM – це шаблон, який полегшує відокремлення розробки інтерфейсу користувача (UI) від бізнес-логіки (рис. 1.3). Як і в класичному MVC, Модель містить дані, а Вид представляє інтерфейс користувача і підписується на події зміни значень властивостей і команд, що надаються Моделлю представленням. Сутність Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. В ній реалізується бізнес-логіка застосунка. Також містить у собі команди за допомогою яких Представлення може впливати на Модель.

Переваги шаблона: легко зрозуміти логіку Представлення, легше тестувати, менше коду. Серед недоліків можна виділити один – даний шаблон підходить для великих проектів, у яких велика кількість складних UI.

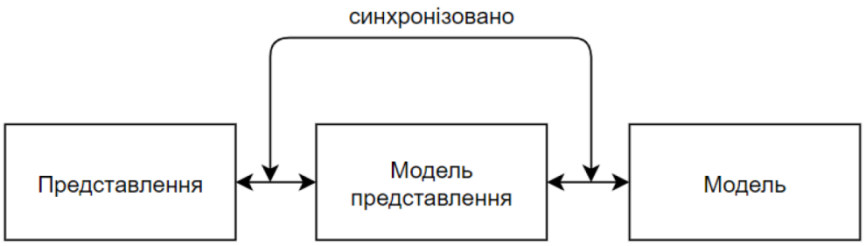

Рисунок 1.3 – Структура MVVM

Вище було сказано, що зазвичай інформаційна система складається з більш ніж одного архітектурного шаблона. Тому описувана в даній роботі інформаційна система не є винятком. Вона поєднує вище описані шаблони певним чином. За основу взято клієнт-серверну архітектуру, тобто ПЗ буде розділене на 2 яруси: клієнт та сервер з базою даних. Взаємодія між клієнтом і сервером буде за допомогою протоколу HTTPS, тобто REST взаємодія. Сервер в свою чергу буде використовувати шаблон MVC, але оскільки передбачається REST взаємодія, тоді шар Представлення буде реалізовано на клієнті у вигляді окремого односторінкового web-додатка. Оскільки клієнт це SPA, то він буде реалізовувати шаблон MVVM, оскільки завдяки ньому з легкістю можна будувати UI різної складності. Детально архітектуру зображено на рис.1.4, однак слід зауважити, що на рисунку також позначені application server та web server. Це означає, що написаний сервер буде виконуватися на сервері додатків (англ. application server) [16].

Для полегшення розробки було обрано 3 найбільш доцільні для даної системи шаблони проектування.

Декоратор – це структурний шаблон проектування, який дозволяє динамічно додавати об’єктам нову функціональність, шлях їх загортання в корисні «обгортки». Даний шаблон корисний в випадках, коли неможливо або небажано використовувати наслідування для розширення функціоналу. Також плюсом виступає те, що об’єктам не тільки можна давати обов’язки (новий функціонал) але й знятих їх при непотрібності.

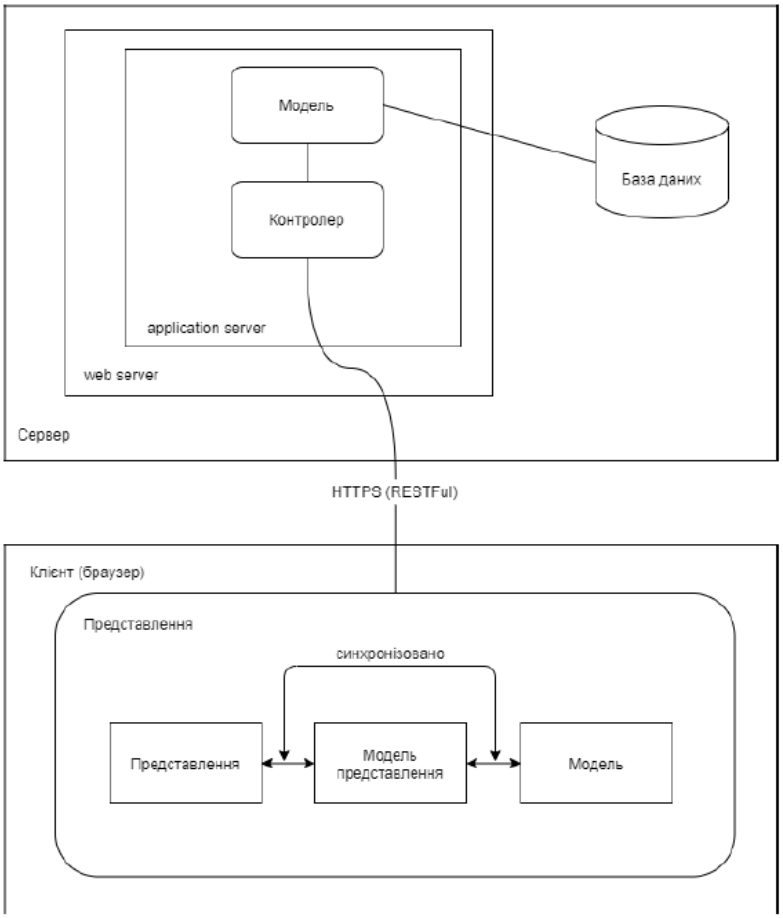

Рисунок 1.4 – Архітектура проекту

Ітератор – це шаблон проектування, який належить до класу шаблонів поведінки. Він дозволяє послідовно обходити елементи складених об’єктів, не розкриваючи їх внутрішнього представлення.

Шаблонний метод – це шаблон проектування, який належить до класу шаблонів поведінки. Він визначає скелет алгоритму, перекладаючи відповідальність за деякі його кроки на підкласи. Шаблон дозволяє підкласам перевизначити кроки алгоритму, не змінюючи його загальної структури.

1.3 Вибір мови програмування

Спочатку визначимось з мовою програмування на клієнті. Оскільки клієнтський додаток буде працювати у браузері в якості web-додатка, то вибір мови програмування стає очевидним – це буде JavaScript або мова, яка

транслюється в нативний JavaScript. Серед мов, що транслюються в JavaScript, розглянемо найбільш відомі TypeScript та CoffeeScript та порівняємо їх з JavaScript. На основі цих порівнянь виберемо найбільш придатну мову для розробки клієнта.

JavaScript – це динамічна, інтерпретована мова програмування високого рівня з підтримкою декількох парадигм програмування. JavaScript це мова програмування web-мережі. Переважна більшість web-сайтів використовує JavaScript, а всі сучасні web-браузери – в персональних комп'ютерах (ПК), планшетах та смартфонах – містять у собі інтерпретатор JavaScript, роблячи її найбільш розповсюдженою мовою програмування за всю історію [6].

JavaScript має динамічну і слабку типізацію, автоматичне керування пам'яттю та прототипне наслідування. Значна особливість JavaScript – функції виступають у якості об'єктів першого роду, тобто функція є об'єктом над яким можна проводити усі операції, що і над іншими об'єктами: передавати функцію у якості аргумента в іншу функцію, повертати функцію в якості результата з іншої функції, присвоювати функції атрибути.

В JavaScript відсутнє паралельне виконання коду з використанням потоків або процесів, але це не проблема, оскільки в JavaScript відмінно реалізована асинхронна обробка даних без якої важко уявити написання більш-менш складного проекту.

Одна з багатьох переваг JavaScript полягає у великій спільноті розробників, а також наявність великої кількість готових плагінів, інструментів та готових рішень для будь-якої задачі.

CoffeeScript – це невеличка мова програмування, яка компілюється в JavaScript. CoffeeScript – це спроба простим способом розкрити усі переваги JavaScript. Код написаний на CoffeeScript компілюється один в один в еквівалентний код на JavaScript, і не підтримує інтерпретації під час виконання. Скомпільований код працює так само швидко, а іноді навіть швидше, ніж еквівалентний код на JavaScript. Також можна використовувати будь-яку існуючу бібліотеку JavaScript із CoffeeScript (і навпаки) [17]. Безумовно пере-

вага CoffeeScript над JavaScript у більш компактному, лаконічному та зрозумілому синтаксисі.

TypeScript – це мова програмування з відкритим кодом, яка базується на JavaScript, додаючи статичну типізацію. Типи надають спосіб описання форми об’єктів, забезпечуючи кращу документацію та дозволяючи TypeScript перевірити чи правильно працює написаний код на етапі компіляції [18]. Код TypeScript перетворюється в код JavaScript за допомогою компілятора TypeScript або Babel. Цей JavaScript код представляє собою чистий та простий код, який запускається всюди, де працює JavaScript, наприклад в браузері. TypeScript дуже економить час на знаходження помилок та їх виправлення.

У табл. 1.1 узагальнено вище сказане про кожну мову програмування на основі порівняння їх можливостей. Звідси бачимо, що незважаючи на свою ефективність, зручність у роботі та інших перевагах, TypeScript та CoffeeScript не можуть зрівнятися з JavaScript. Тому розборка клієнтської частини буде проходити з використанням JavaScript в якості основної мови програмування.

Таблиця 1.1 – Порівняння мов програмування для написання клієнта

	JavaScript	TypeScript	CoffeeScript
Компактний та лаконічний синтаксис	–	–	+
Висока продуктивність	+	+/–	+/–
Статична типізація	–	+	–
Велика кількість готових рішень	+	–	–
Велика спільнота	+	+/–	–
Висока швидкість розробки	+/–	+/–	+

Далі виберемо мову програмування для написання серверної частини. Для цього розглянемо 3 найбільш уживаних для серверної розробки мов програмування. Із рейтингу мов програмування на сайті dou.ua можна побачити,

що за 2021 рік найбільш популярні мови для розробки серверної частини є Java, C# та Python. Отже, порівняємо ці мови та виберемо найбільш доцільну для вирішення поставлених задач.

Java – це об’єктно-орієнтована мова програмування високого рівня, яка має простий але багатослівний синтаксис. Мова має широкий спектр використання, але здебільшого її використовують в ентерпрайз розробці (англ. enterprise development) та для написання мобільних додатків на Android. Програми написанні на Java компілюються в бай-код та інтерпретується віртуальною машиною (JVM). Код на Java переносимий, не залежить від вибору операційної системи (ОС) та має високу швидкість виконання завдяки JIT-компіляції, яка перетворює найбільш уживані частини коду в нативний код ОС.

Java позиціонується як безпечна мова програмування завдяки відсутності вказівників, автоматичному управлінні пам’яті та збирачу сміття. Також мова має чудову реалізацію паралельного виконання коду та статичну типізацію.

За всі роки існування Java об’єднала навколо себе велику спільноту розробників. Завдяки цьому існує багато готових рішень для різних задач на Java.

C# – це об’єктно-орієнтована мова програмування високого рівня з статичною типізацією для платформи .NET. Мова використовується здебільшого для ентерпрайз розробки. В свій час C# зазнала впливу з боку Java, тому велика кількість можливостей, що реалізована в Java також наявна в C#. Однак, C# має ряд особливостей, такі як підтримка динамічного виведення типів, вказівників та перезавантаження операторів. Ці особливості C# ввібрав від мови програмування C++. Тому й не дивно, що синтаксис C# являє собою комбінацію синтаксистів C++ та Java.

Як і Java, за роки свого існування C# зібрав навколо себе вражаючу кількість розробників.

Python – це динамічно типізована, об’єктно-орієнтована мова програмування з відкритим кодам. Python має різні сфери використання, але здебільшого його використовують в web-розробці та Data Science. Він має лаконічний, легко читаємий та приємний синтаксис. Підходить для початківців, оскільки дуже легкий у вивченні. Python можна запустити під управлінням будь-якої операційної системи, тому код на Python є переносимим.

Python має динамічну типізацію, а також усе в ньому є об’єктом. Функції виступають у якості об’єктів першого роду. Також для Python притаманні качина типізація, множинне наслідування, замикання та елегантно реалізоване перевантаження операторів. Python з перших кроків навчає гарному стилю програмування, оскільки в його реалізації входять такі шаблони проектування, як декоратор, ітератор, проксі. Дана мова програмування підтримує декілька парадигм програмування.

Python має потужні елементи функціонального стилю програмування. Навіть такі речі, як наслідування (відноситься до ООП), реалізовано в функціональній манері через дескриптори та розширення функціоналу оператора доступу до атрибутів об’єкта (крапки “.”).

Одним з недоліків Python вважається його повільна робота у порівнянні з мовами зі статичною типізацією. Однак, є декілька способів вирішення цієї проблеми і найлегший с них – використовувати у якості інтерпретатора PyPy. З ним швидкість виконання коду зростає майже в 100 разів і наближається до показників C та C++.

Python має велику спільноту розробників, а отже – й велику кількість фреймворків, готових бібліотек та рішень для різних задач.

Отже, у табл. 1.2 порівняно вище розглянуті мови програмування для написання серверної частини. Звідси можна побачити, що найкращий варіант – це Python.

Таблиця 1.2 – Порівняння мов програмування для написання серверної частини

	Java	C#	Python
Статична типізація	+	+	–
Висока швидкість розробки	–	–	+
Компактний та лаконічний синтаксис	–	–	+
Висока швидкість виконання коду	+	+	+/–
Велика спільнота	+	+	+
Велика кількість готових рішень	+	+	+
Переносимість програм	+	+	+
Безпечне управління пам’яттю	+	+	+

1.4 Вибір технологій для розробки інформаційної системи

Оскільки Python одна з найпопулярніших мов для написання backend, він має велику кількість фреймворків, як великих за розміром та функціоналом, так і малих, як популярних, так і набираючих популярність, які в тій чи іншій мірі полегшають даний процес. Тому, розглянемо найбільш популярні феймворки та оберемо серед них той, що задовільнить усім умовам, а саме: гнучкість, швидкість виконання, наявність повної та якісно написаної документації, наявність готових архітектурних рішень.

Із офіційної документації, Django – це високорівневий web-фреймворк на Python, який сприяє швидкій розробці та чистому, прагматичному дизайну. Він бере на себе більшу частину клопіт під час створення продукту, тому можна зосередитись на вирішенні конкретних бізнес-задач, не вдаючись у написанні повсякденних задач, таких як обробка запитів або обробка URL адресів. Також фреймворк дозволяє в найближчий час перейти від ідеї до конкретного прототипу.

Django робить акцент на безпеку, гнучкість та масштабування. Також він намагається дати усе із коробки, тобто надати розробнику усі необхідні інструменти, такі як ORM, шаблонізатор, роутинг, сигнали, тестовий клієнт та інше. Фреймворк має вбудований WSGI, який можна використовувати під

час розробки і тестування. Однак він не підходить для реального використання, тому необхідно використовувати один з виробничих серверів додатків, наприклад Gunicorn або uWSGI.

Django підтримує концепцію повторного використання додатків – це стало причиною існування великої кількісті готових модулів та бібліотек для даного фреймворка.

Django побудований з використанням великої кількості шаблонів проектування, як архітектурних, так і GoF. Наприклад, Django використовує архітектурний шаблон MVT, що являє собою переосмислений MVC. Як і в MVC, літера M означає модель або model. Однак за контролер (літера C у MVC) відповідає в'ю або view (V у MVT), а представлення (V у MVC) – шаблон або template (T у MVT).

Django заохочує використання різних принципів написання якісного коду, наприклад DRY, KISS.

Діла розглянемо наступний менш популярний у порівнянні з Django, але достатньо популярний фреймворк Twisted.

Згідно офіційній документації Twisted – це подійно-орієнтований (асинхронний) мережевий web-фреймворк на Python. Він підтримує велику кількість протоколів передачі даних та дозволяє з легкістю розгорнути сервер на будь-якому з них. Також фреймворк відомий своєю продуктивністю та швидким виконанням коду завдяки асинхронному виконанні програм.

Однією з важливих можливостей Twisted виступає використання event loop не тільки написаних на Python, а і на інших мовах програмування.

Як і Django, Twisted розроблено з використанням різних шаблонів проектування, наприклад реалізація шаблонів фабричний метод та абстрактна фабрика, які допомагають використовувати в коді різні типи протоколів.

Ще у часи, коли Python не підтримував асинхронність та не мав модуля asyncio, розробники Twisted взялися за розробку власних механізмів асинхронного виконання коду. Тому популярність Twisted зумовлена наявністю асинхронності, а саме відсутності її у Python в минулому. Однак це не

означає, що Twisted погано реалізує асинхронність і скоро кане в Лету. Не зважаючи на те, що фреймворк був одним із перших, він на рівні змагається з більш новими асинхронними фреймворками, а в деяких випадках випереджає їх як у швидкості та продуктивності, так і в частоті появі нових можливостей та функцій.

Отже, зваживши усі переваги та недоліки описаних фреймворків, було вирішено використовувати Django.

Для розробки необхідна бібліотека Django Rest Framework (DRF). DRF – це бібліотека, яка призначена для полегшення розробки API. Вона надає безліч інструментів та дозволяє створити сервер довжиною в декілька строчок коду.

1.5 Вибір бази даних

Важливим аспектом, без якого не можливе існування складних систем, є збереження інформації. Найбільш придатними виступають бази даних. Зараз виділяють 2 основні види баз даних: SQL та NoSQL.

SQL бази даних – це різновид баз даних, які використовують декларативну мову програмування SQL. Зазвичай їх використовують, коли між сутностями є тісні зв’язки. Перевагами SQL баз даних є незалежність від конкретної системи керування баз даних (СКБД), наявність стандартів та опис намірів (декларативність), тобто програміст описує тільки ті дані, які необхідно дістати або змінити. Серед недоліків можна зазначити, що SQL бази даних мають складності в роботі з ієрархічними структурами даних.

NoSQL бази даних – це різновид баз даних, які не використовують SQL. По суті, NoSQL дані не є реляційними, а NoSQL бази даних зазвичай не мають схем і вони мають більш узгоджену модель, ніж наявні в традиційних реляційних БД. Перевагами NoSQL баз даних виступають легкість у масштабуванні та висока продуктивність. Серед недоліків NoSQL баз даних

можна зазначити низьку надійність, великий об’єм даних та залежність від СКБД.

Зважаючи на усі переваги та недоліки вище описаних типів баз даних, для написання проекту було обрано SQL бази даних, оскільки для проекту важливі структура та зв’язки між сутностями. Далі порівняємо та виберемо для проекту базу даних.

Oracle Database – об’єктно-реляційня система керування базами даних компанії Oracle. Була розроблена для корпоративних розподілених обчислень. Перевагами даної СКБД є висока швидкість обробки транзакцій, сумісність з ACID принципами, обробка великих даних, популярність у розробників, простота у використанні, зрозуміла документація, підтримка довгих найменувань, JSON, покращений тег списку і Oracle Cloud. Найбільший недолік Oracle – її висока вартість.

PostgreSQL – масштабна об'єктно-реляційна СКБД, що працює на Linux, Windows, OSX і деяких інших системах. У PostgreSQL є такі функції, як логічна реплікація, декларативне розбиття таблиць, поліпшені паралельні запити, більш безпечна автентифікація по паролю на основі SCRAM-SHA-256. Також у PostgreSQL є підтримка табличних просторів, збережених процедур, об'єднань, представлень і тригерів та асинхронна реплікація. У порівнянні с Oracle, Postgres має менший функціонал та продуктивність, але в той же час Postgres має найбільшу перевагу – безкоштовність.

Безкоштовність та функціонал PostgreSQL цілком задовольняє цілям проекту, тому розробка буде проходити з її використанням.

2 ПРОЕКТУВАННЯ ОСВІТНЬОГО РЕСУРСУ

2.1 Мета і задачі інформаційної системи

Мета інформаційної системи: надання користувачам можливості проходження курсів та інших видів навчання у сфері інформаційних технологій для покращення свої знань або набуття нових; зробити навчання якомога більш доступним та швидким для широкого колу людей.

Веб-сайт є безкоштовним освітнім ресурсом для навчання різним мовам програмування і технологій, що використовуються при розробці. На ресурсі присутні курси для початківців, тих, хто має деякі знання мов програмування та розробки в цілому, і розробників з достатнім досвідом роботи. Навчання проводиться в ігровій системі, що істотно збільшує інтерес користувачів до навчання. Інформація подається за допомогою відео, після кожного уроку користувач проходить тест, після проходження якого, можна говорити про засвоєння даної інформації. Для успішного завершення курсу, необхідно пройти модуль, який відкривається після проходження всіх уроків.

Важливою частиною функціонування освітнього ресурсу є роздільне зберігання використовуваної на ресурсі інформації в різних базах даних, які дозволяють краще і швидше працювати саме з певною інформацією. Це істотно прискорює роботу ресурсу і дозволяє робити завантаження сторінки більш швидкої, що сприятливо впливає на бажання користувача займатися навчанням саме в розроблюваному ресурсі.

Отже можна виділити наступні задачі системи:

– навчання користувачів за допомогою курсів за різними спеціальностями та рівнями підготовки;

– надавання додаткових джерел інформації у вигляді вебінарів та можливість поставити питання у коментаріях;

– перевірка знань користувачів після кожного уроку для визначення засвоєності навчального предмету;

– збільшення швидкості завантаження сторінок ресурсу та загальної швидкості його роботи за рахунок використання багатоваріантної персистентності.

Вхідні потоки: запити користувача до ресурсу.

Вихідні потоки: корисна для людини інформація у відповідь на запити.

На рисунку 2.1 представлена контекстна діаграма, на якій відображені вхідні і вихідні потоки освітнього ресурсу.

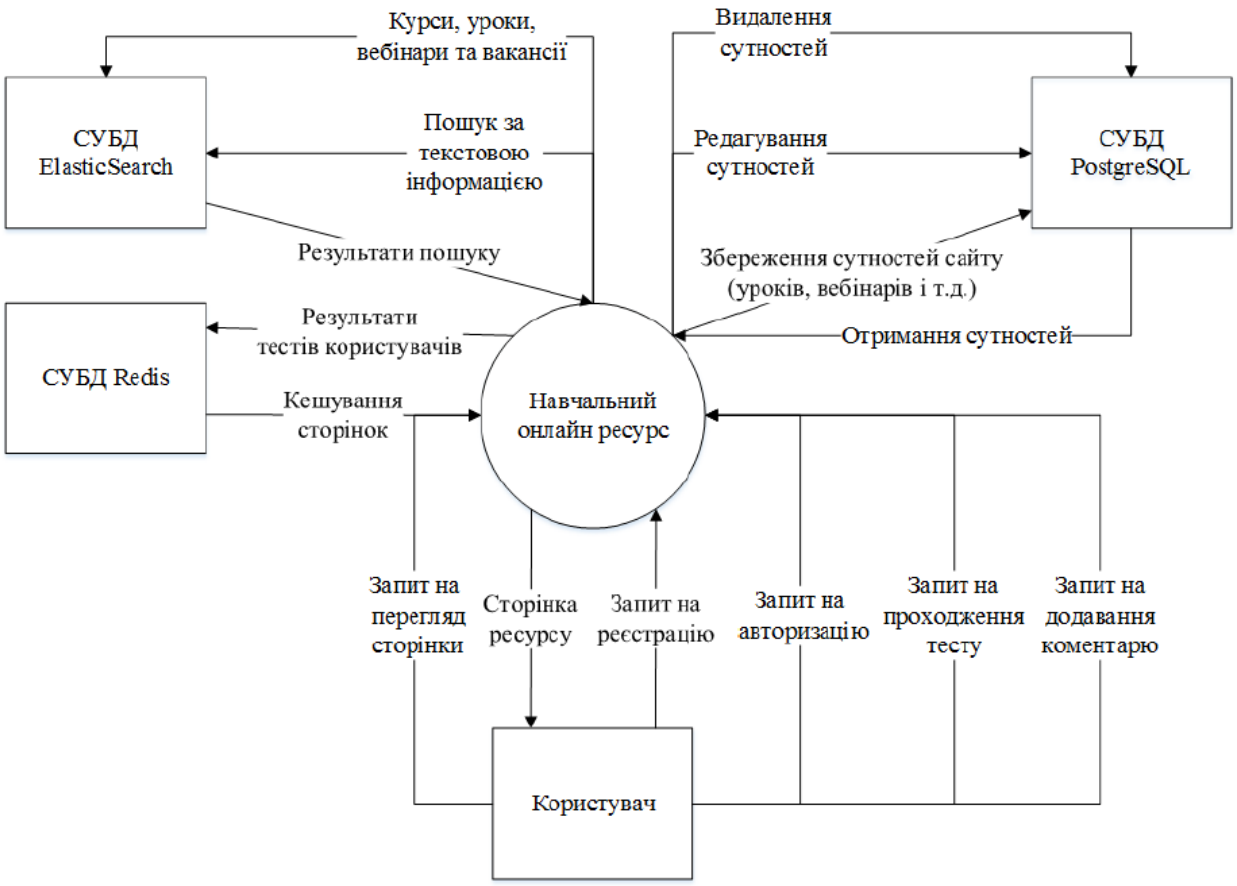

Рисунок 2.1 – Діаграма вхідних та вихідних інформаційних потоків

2.2 Типи користувачів

Гість – початковий користувач, який ще не пройшов реєстрацію або ще не увійшов до свого аккаунту. Може відвідувати такі сторінки ресурсу: головна, список курсів, список вебінарів, список вакансій, сторінка інформації про ресурс, сторінка реєстрації, сторінка входу до облікового запису.

Звичайний користувач – користувач, який пройшов реєстрацію і увійшов до свого аккаунту. Додатково до можливостей, які надаються гостю, він також може заходити на такі сторінки: сторінки окремих курсів і уроків, тестування після кожного уроку і проходження модуля, сторінку певного вебінару після реєстрації на нього, профіль і його редагування, а також у нього є можливість залишати коментарі до уроків і вебінарів.

Адміністратор має всі можливості звичайного користувача, а також може заходити на сторінку адміністрування, де має можливість видаляти будь–які з доданих елементів в базі даних, тобто курси, вебінари, спеціальності, коментарі і т.д.

Після визначення користувачів було побудовано діаграми прецедентів. На рисунку 2.2 представлена діаграма прецедентів для актора «Гість».

Рисунок 2.2 – Діаграма прецедентів для актора «Гість»

Дії з ресурсом, можливі до виконання після авторизації до акаунту були виділені в окрему частину діаграми для забезпечення можливості їх подальшого використання у декількох інших діаграмах без необхідності повторення (рис. 2.3).

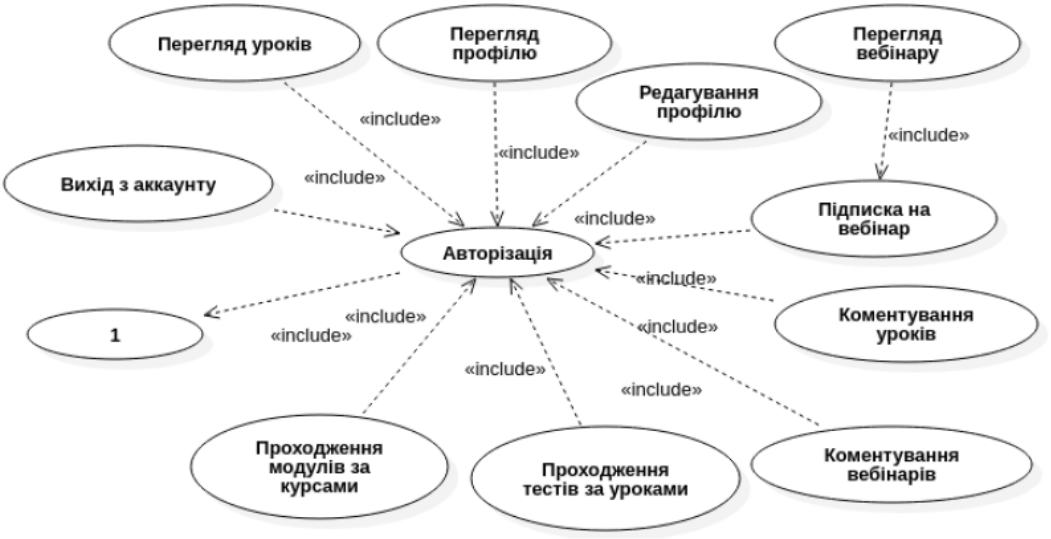

Рисунок 2.3 – Прецеденти, які доступні для користувача після авторизації

На рис. 2.4 представлена діаграма варіантів використання ресурсу для актора «Звичайний користувач».

Рисунок 2.4 – Діаграма прецедентів для актора «Звичайний користувач»

На рис. 2.5 представлена діаграма прецедентів для «Адміністратора».

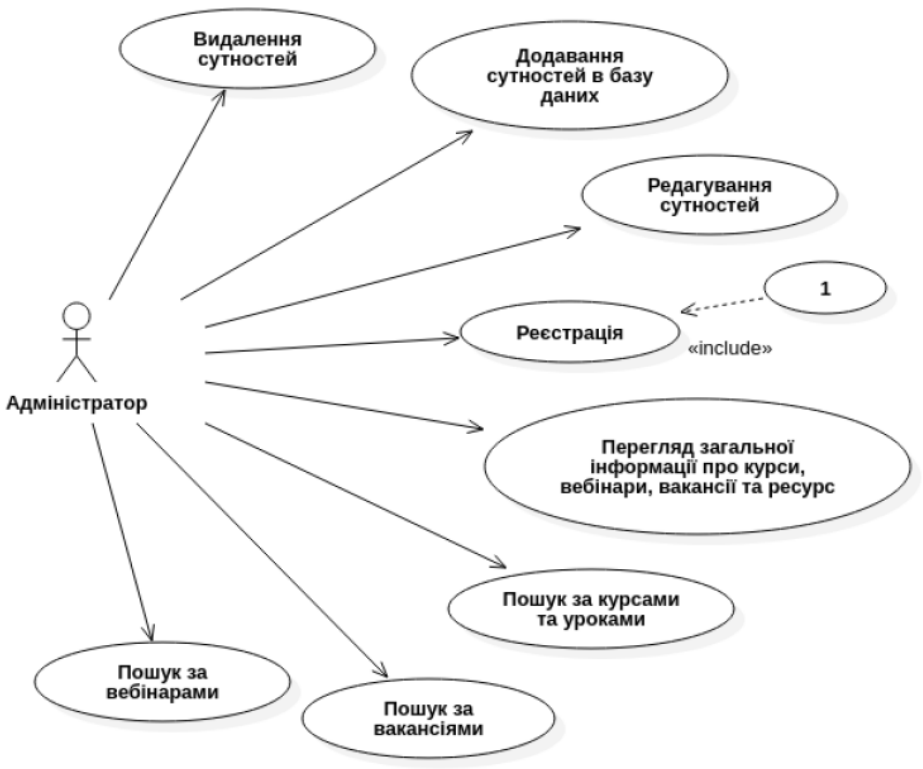

Рисунок 2.5 – Діаграма прецедентів для актора «Адміністратор»

З рисунку можна побачити, що адміністратору, окрім базової функціональності, а також повної функціональності, яка відкривається після авторизації, доступні також маніпуляції з усіма сутностями у ресурсі.

2.3 Представлення інтерфейсу користувача (UI View)

З будь–якої сторінки ресурсу можлива навігація по кнопках в головному меню, яке знаходиться вгорі кожної сторінки. Там присутні кнопки JustStart, яка виконує перехід на головну сторінку, Webinars (перехід на список вебінарів), Courses (перехід на список курсів), IT specialties (перехід на список вакансій), About us (перехід на інформацію про ресурс – про нас). Також правіше головного меню, в залежності від того авторизовані користувач,

можуть відображатися або три кнопки (коли користувач авторизовані), зліва направо –

перехід на профіль користувача, перехід в редагування профілю, вихід з облікового запису. Якщо користувач не авторизовані, замість трьох кнопок буде відображатися одна кнопка Login (перехід на сторінку авторизації – логін). На рис. Б.1 представлена головна сторінка ресурсу.

На головній сторінці представлена базова інформація про сайт. Крім переходу по меню, на головній сторінці можна виконати перехід по кнопці Just Start в центрі екрану, яка переадресує користувача на сторінку реєстрації.

На сторінці реєстрації розгорнута форма реєстрування нового аккаунта користувача, яка зображена на рис. Б.2. У формі присутні такі поля: ім'я користувача (*), адреса електронної пошти (*), посилання на Github аккаунт (*), країна проживання, пароль (*) і малюнок або фото, що представляє із себе характерне зображення користувача в профілі. Значком * відмічені обов'язкові поля у формі.

Якщо користувач натисне на кнопку Cancel, його буде переадресовано назад на головну сторінку. Якщо він заповнить всі необхідні поля правильно і натисне на кнопку Sign up, ресурс перенаправить його на сторінку профілю. Однак, якщо якісь поля будуть заповнені неправильно або ті поля, які є обов'язковими будуть незаповнені, йому буде видано попередження про це і буде очікуватися виправлення введеної інформації та повторна відправка форми реєстрації.

Сторінка логіна дозволяє здійснити вхід в акаунт (виконати аутентифікацію користувача) за допомогою імені користувача та пароля. Вона зображена на рис. Б.3.

Кнопка Login переадресує користувача на сторінку профілю при успішної аутентифікації. Кнопка Sign Up перенаправляє на сторінку реєстрації, де користувач може створити новий обліковий запис. Кнопка Cancel повертає відвідувача на головну сторінку.

На сторінці "Про нас" представлена коротка інформація про ресурс, роботі сайту, а також цілі створення даного ресурсу. Ця сторінка зображена на рис. Б.4. Навігація з неї можлива тільки по головному меню і посиланнях на соціальні мережі.

Сторінка списку з вакансіями (списку спеціальностей), яка зображена на рис. Б.5, представляє з себе список з актуальними пропозиціями роботи в різних містах України в сфері IT. Актуальність досягається оновленням всього списку за допомогою створеного парсеру вакансій, який включається раз в день та повністю оновлює список вакансій.

Сторінка списку курсів містить всі наявні на сайті курси з різних мов програмування. Вона представлена на рис. Б.6. Можна зробити пошук по курсам і уроків за допомогою поля для введення запиту. Сам пошук виконується після натиснення кнопки Search, яка зробить перенаправлення на таку ж сторінку зі знайденими курсами, які відповідають запиту. Пошук виконується за назвами, ключовими словами і описами уроків і курсів. Користувач перенаправляється на сторінку з вступом для курсу при натисканні на обраний курс.

На рис. Б.7 представлена сторінка вступу курсу зі списком уроків. На початковій сторінці курсу знаходиться його короткий опис і посилання на уроки, а також посилання на модуль. Модуль доступний для проходження тільки після успішного виконання всіх тестів уроків. Посилання перенаправляють на відповідні сторінки уроків.

Сторінка уроку складається з опису, посилання на відео, секції з коментарями, поля для додавання коментарів і посилання на тест. Вона представлена на рис. Б.8. Також на сторінці присутні посилання на інші уроки. Для додавання коментарів необхідно ввести текст в поле введення і натиснути на кнопку Send. Кнопка Pass Test виконує пересилку користувача на сторінку тесту. Після кожного уроку на ресурсі присутній тест. Проходження кожного тесту безпосередньо впливає на рівень користувача і прогрес по курсу, який відображається в профілі в секції Skills.

Сторінка зі списком вебінарів, яка представлена на рис. Б.9, дозволяє побачити всі доступні на сайті вебінари, а також провести пошук по ним. Пошук працює абсолютно таким же чином, як і для курсів. При натисканні на обраний вебінар, відкривається сторінка з введенням для вебінару (сторінкою підписки на вебінар).

Сторінка вступу вебінару, яка зображена на рис. Б.10, складається з короткої інформації про нього, кількості підписаних на нього користувачів і кнопки Join з назвою самого вебінару, яка дозволяє підписатися на нього і перенаправляє на сторінку вебінару.

Зовнішній вигляд сторінки вебінару представлений на рис. Б.11. На ній присутні: його короткий опис, відео вебінару, форма для додавання коментарів і список коментарів до поточного вебінару. Для додавання коментарів необхідно ввести текст в поле і натиснути на кнопку Send.

Сторінка з тестами представляє собою кілька тестів, зазвичай близько двох-трьох. Приклад сторінки з тестом зображений на рис. Б.12. Кожен тест знаходиться на своєму слайді, перемикання між ними відбувається по кнопці Next. Якщо не було дано відповіді на поточний тест, перехід на наступний буде неможливий. Коли користувач пройде всі тести на сторінці, йому буде показана кнопка Pass Test, розташована на місці кнопки Next. Питання в тестах бувають двох видів – відкритого типу, в якому є поле для введення і необхідно дати письмову відповідь і тест з варіантами відповідей, де потрібно вибрати один варіант з кількох можливих. Після виконання тесту користувача перенаправить на сторінку показу результату тесту.

Сторінка результату тесту дає користувачеві можливість побачити зміг він пройти поточний тест чи ні. Один з варіантів сторінки результату тестування зображений на рис. Б.13. Тест вважається пройденим, якщо відповіді на всі питання були вірними, інакше тест не зараховується. Якщо тест виконаний правильно, до прогресу користувача за поточним курсом додається бали, як і до прогресу по набору рівня.

На сторінці профілю (рис. Б.14) відображена інформація про введені ним дані при реєстрації, а також про курси, за якими він проходив тести, в секції Skills і відображається набраний їм рівень відповідно до рівню досвіду за проходження уроків і курсів.

Сторінка редагування профілю зображена на рис. Б.15. На ній можна виконати зміну інформації про користувача, а саме тієї, що була введена їм при реєстрації. При натисканні кнопки Save відбувається збереження даних і

перенаправлення користувача на сторінку профілю за умови, що всі дані, що змінюються були введені вірно. Кнопка Cancel також виконує перенаправлення на сторінку профілю, але без збереження введених даних.

Виходячи зі створених макетів сторінок освітнього ресурсу, була побудована діаграма переходів станів, що показує можливі переходи з одного стану в інший, тобто з однієї сторінки на іншу (рис. 2.6).

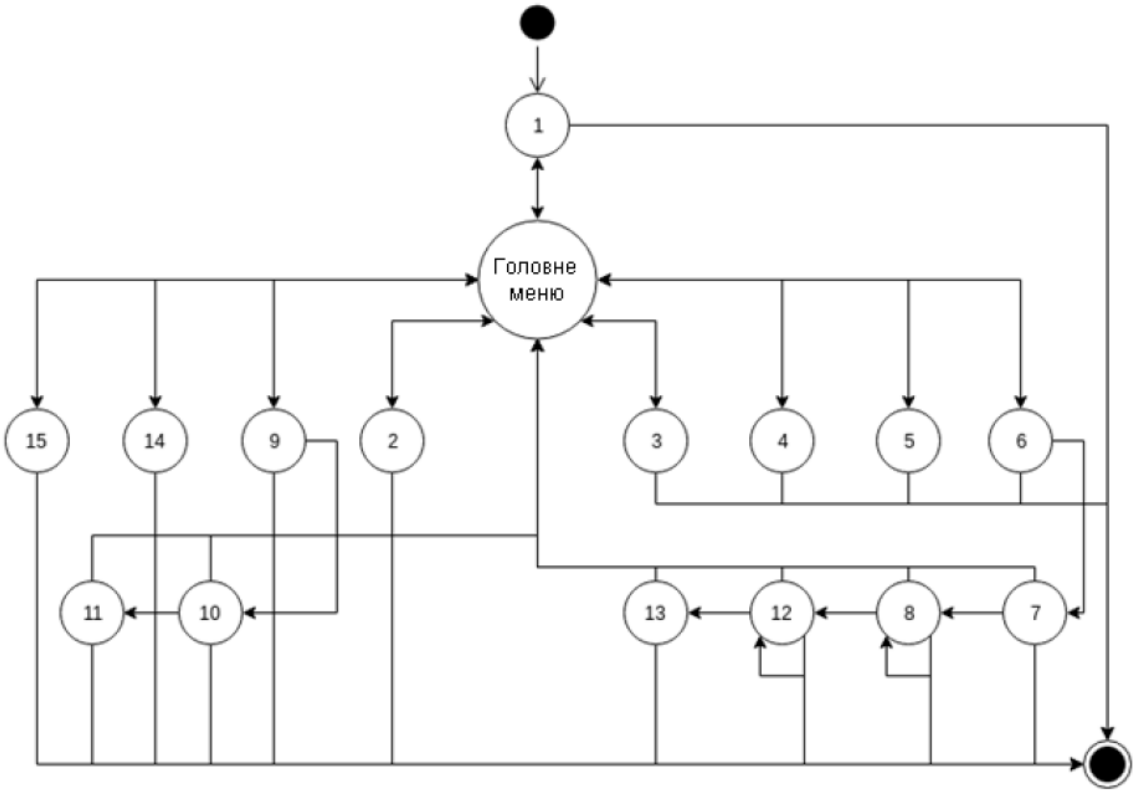

Рисунок 2.6 – Діаграма станів навчального ресурсу

З представленої діаграми можна побачити, що взаємодія з ресурсом відбувається за допомогою головного меню. Також, через те, що навчальний ресурс є веб-сайтом, закінчення роботи з ним можливе з будь-якої сторінки.

2.4 Логічне представлення ІС (Logical View)

Логічне представлення навчального ресурсу, а саме його структурні елементи, зв'язок між ними та приклади типів даних, що зберігаються у різних моделях БД, представлені на рисунку 2.7. Ділянка, що представлена на діаграмі, як middleware є основним компонентом ресурсу, який забезпечує зв'язок між усіма іншими компонентами сайту та виконує керування за усією логікою та функціональністю системи. Саме в ньому виконується реалізація концепції багатоваріантної персистентності, яка є основою функціонування ресурсу та забезпечує збільшення швидкості його роботи.

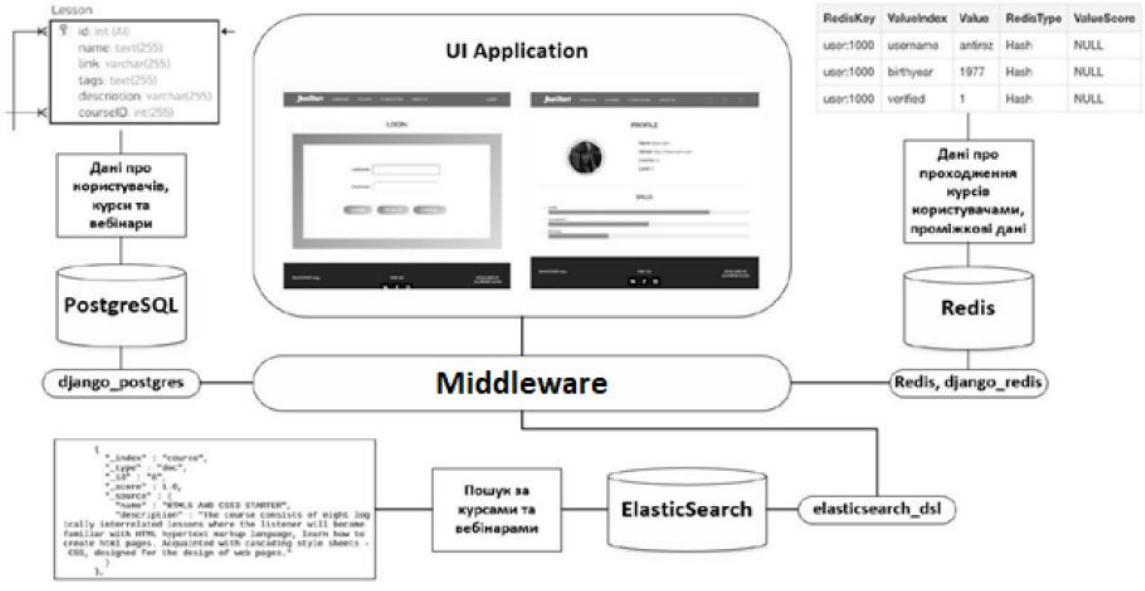

Рисунок 2.7 – Діаграма логічного представлення ресурсу

Графічний інтерфейс представляє собою інтерфейс для взаємодії з усім навчальним ресурсом та забезпечує доступ до всієї функціональності сайту, відповідно до типу користувача, який виконує взаємодію.

Middleware представляє собою головний сервер, який є проміжковим для усіх інших серверів для керування та передачі даних між ними. Керування здійснюється за допомогою взаємодії з різноманітними модулями роботи з БД для формування запитів або отримання даних, як результат цих запитів. При цьому формування запиту може бути не лише для отримання даних, але наприклад і редагування або видалення їх. Сервер працює у відповіді на дії користувача у графічному інтерфейсі.

Головний сервер частково використовує фреймворк django для взаємодії з базами даних. Робота з СУБД PostgreSQL відбувається за допомогою модуля django_postgres та використання об’єктно-реляційного відображення сутностей у базі та взаємодії з ними.

Робота з СУБД Redis виконується частково з використання цього фреймворку за допомогою модулю django_redis для мови програмування Python. Він дозволяє зберігати у цій БД кешоване представлення сторінок, які часто використовуються. Іншим використанням СУБД Redis є збереження проміжкової інформації, оновлення якої проводиться швидко або яка потрібна лише на деякий короткий час. Взаємодія з базою при такому сценарії ведеться за допомогою модулю redis.

Робота з СУБД ElasticSearch відбувається за допомогою модулю elasticsearch_dsl для забезпечення зберігання індексованих текстових даних, за якими користувач може виконувати пошук, та подальшого отримання даних, слова у яких співпадають зі словами пошукового запиту.

СУБД PostgreSQL використовується у навчальному ресурсі, як RDBMS – реляційна база даних, для збереження даних про користувачів, запис їх на курси та вебінари, зв’язки між курсами, уроками, тестуванням та іншими сутностями. Також у ньому зберігаються дані, для яких необхідно забезпечити найбільшу безпеку збереження та отримання.

СУБД ElasticSearch використовується у ресурсі для забезпечення пошуку за курсами, уроками та вебінарами за рахунок дуже швидкого доступу до текстових даних через використання хешованих значень кожного слова.

СУБД Redis використовується для збереження кешу сторінок та ресурсів, які змінюються не часто та не потребують виконання перезавантаження після оновлення сторінки. Це дозволяє значно збільшити швидкість завантаження сторінок та інших додаткових ресурсів. Також ця технологія використовується для збереження проміжкової інформації, наприклад для збереження відповідей користувача на тестування для перевірки їх на сервері та переадресації користувача на відповідну до результату сторінку.

2.5 Представлення розгортання ІС (Deployment View)

Діаграма розгортання застосовується для уявлення загальної конфігурації і топології розподіленої програмної системи і містить зображення розміщення компонентів по окремих вузлах системи. Крім того, діаграма розгортання показує наявність фізичних з'єднань – маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи.

ІС складається з веб-серверу, баз даних та комп’ютера користувача, що зображено на діаграмі представлення розгортання (рис 2.8).

Користувач переходить за адресою сайту, після чого взаємодіє з компонентами сайту. Для обробки дій користувача та передачі даних використовується протокол HTTP. Дані відправляються на сервер.

Сервер Django WSGI Server розташований у компоненті програмного забезпечення Docker. Django-проект має зв’язки з різними БД, такими як PostgreSQL, Elasticsearch, Redis.

Компонент PostgreSQL використовується для зберігання даних про користувачів, запис їх на курси та вебінари, зв’язки між курсами, уроками, тестуванням та іншими сутностями. Компонент серверу взаємодіє з ним за допомогою бібліотеки django_postgres.

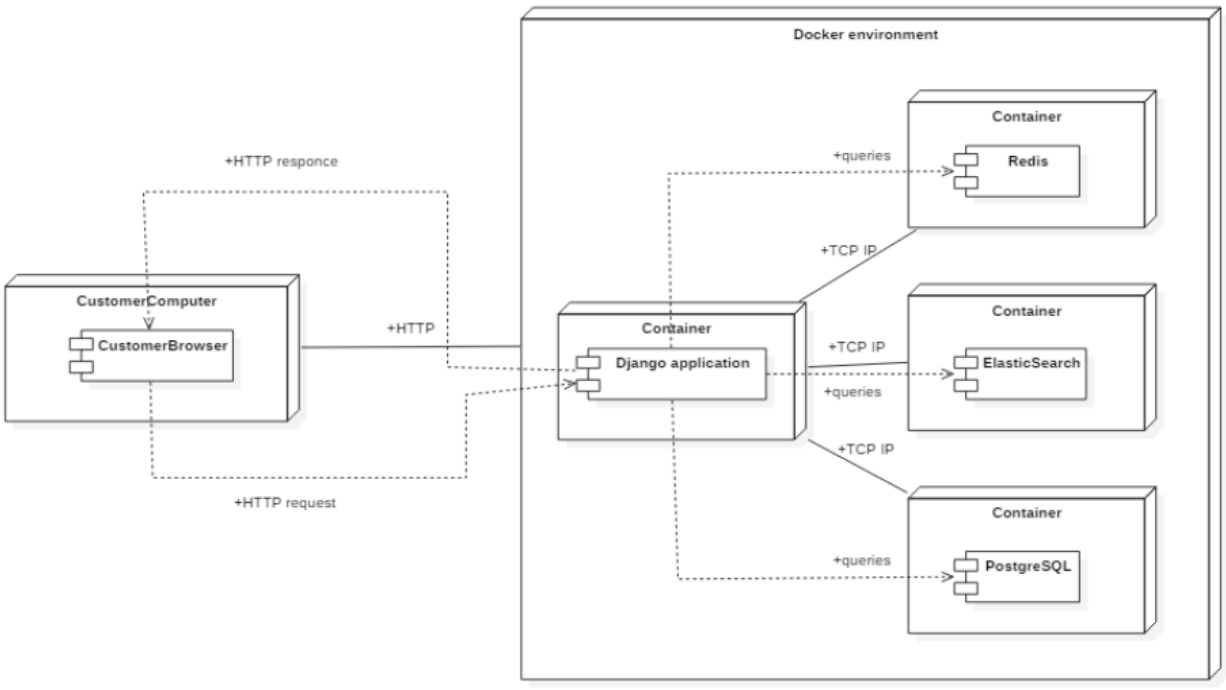

Рисунок 2.8 – Діаграма розгортання ІС

Компонент Elasticsearch використовується для забезпечення пошуку за курсами, уроками та вебінарами за рахунок дуже швидкого доступу до текстових даних через використання хешованих значень кожного слова. Компонент серверу взаємодіє з ним за допомогою бібліотеки elasticsearch_dsl.

Компонент Redis використовується для збереження кешу сторінок та ресурсів, які змінюються не часто. Компонент серверу взаємодіє з ним за допомогою бібліотеки django_redis.

2.6 Представлення даних ІС (Data View)

Система потребує багато ресурсів та оперує великою кількістю різних за типом даних, тому для зменшення навантаження на сервер та збільшення швидкості завантаження сторінок, а також збільшення швидкості роботи ресурсу було використано багатоваріантну персистентність з використанням трьох баз даних, які функціонують окремо одна від одної.

За допомогою СУБД ElasticSearch у навчальному ресурсі проводиться текстовий пошук введених користувачем слів. Ця технологія не може забезпечити достатньої простоти для обробки зв’язних даних, що відбувається у SQL базах даних за рахунок побудови структури таблиць. Тобто вона буде потребувати ручного видалення відповідних значень у базі даних, при видаленні пов’язаних з ним інших даних. Але при цьому за допомогою індексації, тобто збереження хешу від кожного зі слів тексту окремо, можна проводити пошук певних слів набагато швидше, ніж при використанні інших СУБД для цих цілей.

У СУБД ElasticSearch дані зберігаються у виді, так званих, індексів, які є аналогом таблиць у реляційних базах даних. Внутрішня структура індексів схожа за видом на тип даних «словник» у мові програмування Python, а також на тип json файлів. Тобто вона складається з деякої кількості об’єктів типу ключ-значення.

У навчальному ресурсі використовується чотири типа індексів для ElasticSearch для зберігання даних з таблиць курсів, уроків, вебінарів і вакансій, що зображено на рис. 2.9.

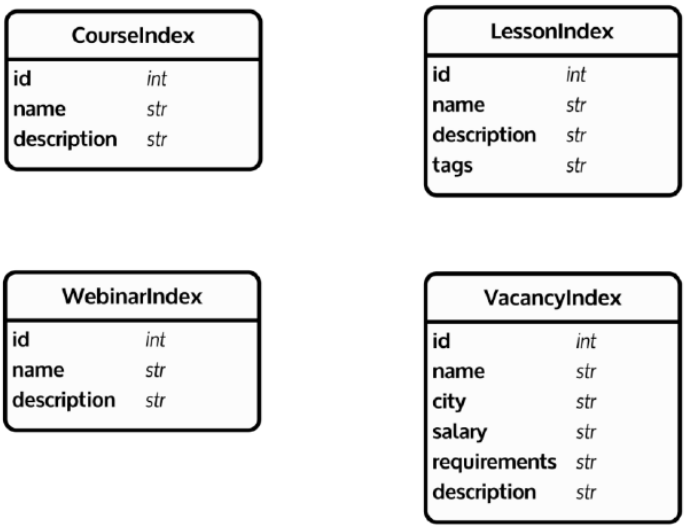

Рисунок 2.9 – Представлення індексів у СУБД ElasticSearch

У курсах індексується назва і опис; в уроках – назва, опис і ключові слова; в вебінарах – теж назву і опис; у вакансіях – назва, місто, для якого виконується пошук, вимоги та опис.

СУБД Redis використовується у ресурсі для зберігання кеша даних сторінки користувача для прискорення доступу до неї, інших часто використовуваних сторінок, дані на яких змінюються досить нечасто, а також для тимчасового зберігання даних, які потребують швидкої обробки або потрібні лише на короткий проміжок часу, наприклад, тимчасове зберігання відповідей на питання при проходженні тесту користувачем для їх подальшої перевірки та визначенні чи пройшов користувач тестування.

Дані в цій СУБД мають формат пар ключ-значення. Доступ до них отримується дуже швидко, за рахунок того, що вони зберігаються в оперативній пам’яті пристрою користувача, але це значно обмежує використання цієї технології, бо величина цих даних не може перевищувати величину оперативної пам’яті пристрою.

У кожного файлу кеша є свій ключ, який вказує на нього. Кеш примусово оновлюється тільки в двох випадках: користувач або адміністратор редагував дані, які присутні на сторінці або, наприклад, користувач вийшов з облікового запису, якщо мова йде про кеш профілю користувача. В такому випадку видаляються всі ключі зі старими даними на машині користувача і оновлюється кеш.

СУБД PostgreSQL є реляційною БД, що забезпечує гарну роботу з даними, пов’язаними між собою значним чином. Але при цьому ця технологія не є кращим варіантом для зберігання даних, які необхідно використовувати часто або виконувати швидку обробку великої кількості даних, через значне зростання необхідного для виконання обробки запиту часу, при збільшенні об’єму даних.

На рисунку 2.10 зображено діаграму сутностей системи, що показує організацію та взаємодію сутностей, збережених у СУБД PostgreSQL. Ці

сутності були створені за допомогою об’єкт-реляційної моделі фреймворку django.

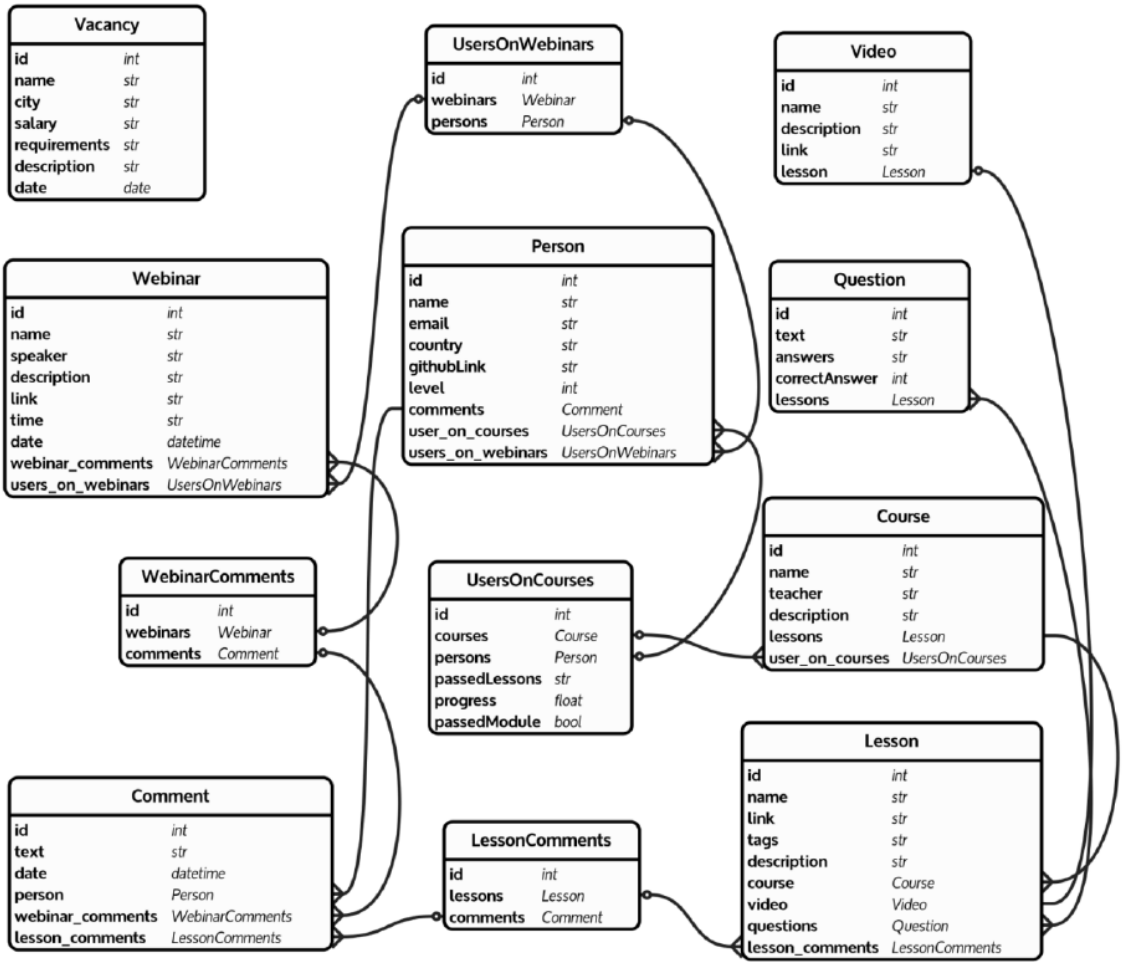

Рисунок 2.10 – Діаграма сутностей у СУБД PostgreSQL

Опис представлених в діаграмі сутностей:

− сутність Course є окремим курсом навчання, у якого є опис, який відображається у вступі до курсу і його назва, а також поле викладача.

− сутність Lesson є окремим уроком курсу. Кожен урок має назву, опис, посилання на відео в уроці, ключові слова і посилання на курс, якому належить цей урок.

− сутність Webinar є вебінаром, що проводиться одним викладачем. У нього є назва, опис, поле викладача, посилання на відео вебінару, його тривалість і дата проведення.

− сутність Vacancy являє собою вакансію на роботу. У неї є такі поля: найменування необхідного фахівця, місто, для якого актуальна вакансія, пропонована заробітна плата, вимоги до фахівця, опис вакансії, дата додавання вакансії.

− сутність Person являє собою користувача ресурсу. У сутності є такі поля: ім'я користувача (логін), посилання на github аккаунт, країна проживання, пароль, адресу електронної пошти і рівень, якого досяг користувач, проходячи тести з курсів.

− сутність Comment являє собою коментар користувача, в якому є посилання на коментатора, текст коментаря і дата його додавання.

− сутність Question представляє собою окремий тест, який може бути двох видів – закритого (з декількома варіантами відповіді) і відкритого (з одним варіантом відповіді). У нього є такі поля: питання, відповідь (правильна), і варіанти відповідей (або прочерк, якщо питання відкритого типу), а також посилання на урок, якому належить цей тест.

− сутність WebinarComments являє собою зв'язок між окремим вебінаром і коментарями до нього.

− сутність LessonComments являє собою зв'язок між окремим уроком і коментарями до нього.

− сутність UsersOnCourses являє собою зв'язок між користувачами і курсами, які вони проходять. У кожного користувача є певний рівень прогресу по розпочатому їм курсу.

− сутність UsersOnWebinars являє собою зв'язок між користувачами і вебінаром, на які вони записані.

Зв'язок між даними з усіх БД представлений на рисунку 2.11. При цьому усі зв’язки між СУБД відбуваються лише через сервер та різноманітні модулі для отримання та маніпуляції з даними у бази.

Зв'язок модулів СУБД PostgreSQL і СУБД Redis (для збереження кешу сторінок) з проектом проводиться за допомогою файлу settings.py, що здійснює налаштування стандартними засобами фреймворка django. Також

доступ до СУБД Redis для збереження та отримання потрібних лише тимчасово даних проводиться за допомогою модулю redis. Зв'язок СУБД ElasticSearch з проектом проводиться за допомогою модулів мови Python – elasticsearch і elasticsearch_dsl, а також файлу search.py, що забезпечує інтеграцію з іншими компонентами фреймворка django.

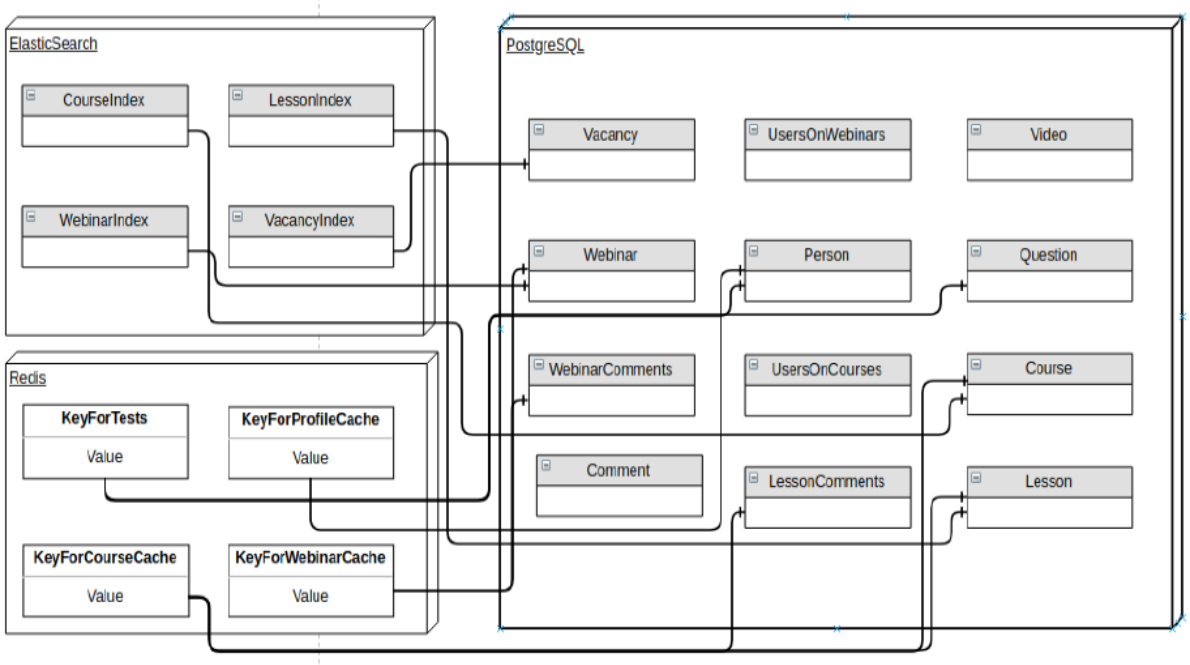

Рисунок 2.11 – Представлення зв’язку між даними у різних БД

Індекси у СУБД ElasticSearch створюються та оновлюються разом з відповідними сутностями у СУБД PostgreSQL за допомогою, за допомогою, так званих, «сигналів» фреймворку django, що активуються саме в потрібні моменти.

Кешування сторінок у СУБД Redis робиться автоматично за допомогою засобів django, але оновлення кешу при редагуванні сутностей у СУБД PostgreSQL, потребувало додаткової інтеграції також за допомогою «сигналів».

Тимчасові дані, наприклад, для зберігання відповідей користувача на тести для уроку, видаляються одразу ж після закінчення їх використання. Та-

ким чином, менше навантажується сервер PostgreSQL, що дає змогу йому швидше виконувати обробку більш важливих запитів від серверу ресурсу.

3 РЕАЛІЗАЦІЯ ОСВІТНЬОГО РЕСУРСУ

3.1 Уявлення про структуру проекту

Розробка здійснювалася за спроектованими схемами та діаграмами, дотримуючись основних вимог до освітнього ресурсу та його швидкодії. Увага також приділялась особливостям використання та розробки на мові програмування Python, веб-фреймворку Django та використаним у кваліфікаційній роботі СУБД.

На рис. 3.1 зображена структура проекту, яка складається з базового пакету «composer» та пакету «education», який розширює його.

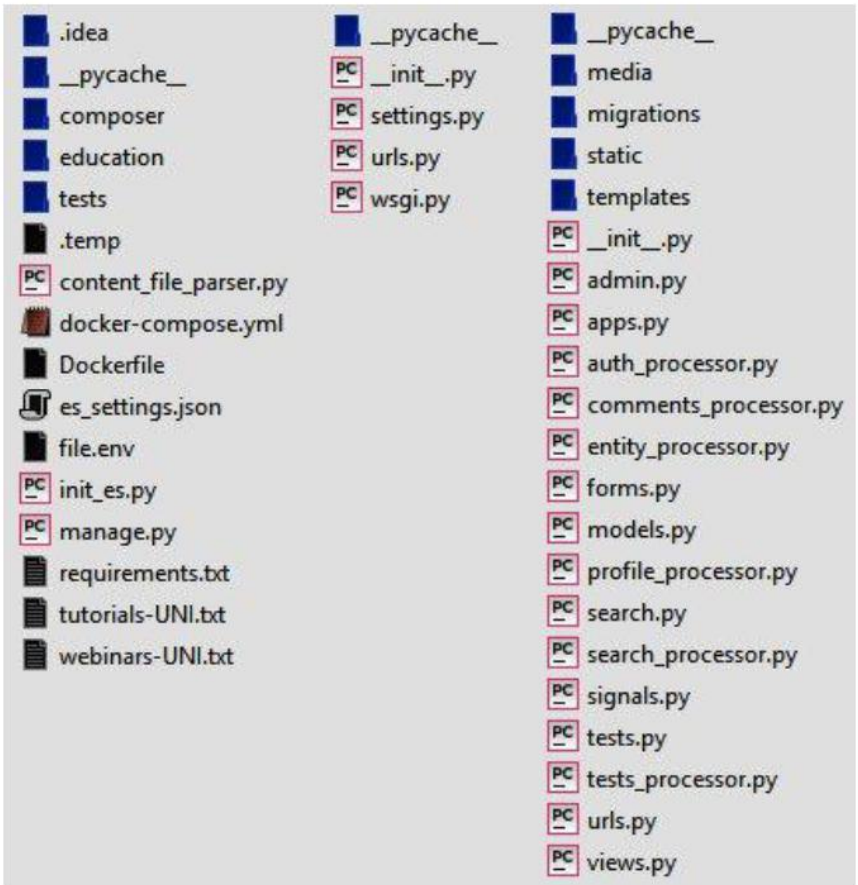

Рисунок 3.1 – Структура проекту веб-додатку

Пакет «composer» містить файли «_init_.py», «settings.py», «urls.py», «wsgi.py». Файл «settings.py» містить налаштування проекту, файл «urls.py» - базову адресацію за посиланнями, а саме перехід на головну сторінку та на сторінку адміністратора.

Пакет «education» являє собою модуль проекту, який містить файли «_init_.py», у якому відбувається ініціалізація початкової конфігурації проекту, «admin.py», де відбувається підключення моделей проекту (тобто таблиць у СУБД PostgreSQL) до можливості додавання та редагування сутностей цих моделей у панелі адміністратора. Файл «forms.py», де визначені усі форми та їх поля, що використовуються у ресурсі. Файл «models.py», що визначає моделі проекту з усіма їх полями, які використовуються для більш простої взаємодії з СУБД PostgreSQL та являють собою кожну з таблиць у базі. Файл «search.py», за допомогою якого відбувається більша частина роботи з СУБД ElasticSearch. Файл «signals.py», де визначаються функції, викликання яких відбувається при певних умовах (наприклад, при зміні об’єкту під час його збереження або при настанні певної події). Файл «urls.py», у якому визначена вся адресація проекту, тобто перехід за кожним з посилань у строці адресу та «views.py», у якому виконується обробка усіх даних, що відправляються та отримуються з серверу. При цьому частину цього функціоналу виконуються в інших файлах, згрупованих за типом використання (наприклад, для роботи з тестами або коментуванню). Файл «auth_processor.py» відповідає за обробку реєстарції та авторизації користувачів, файл «comments_processor.py» відповідає за створення нових коментарів, файл «entity_processor.py» допомагає виконувати обробку даних для сторінок сутностей (тобто курсу, уроку, вебінару тощо), за допомогою файлу «profile_processor.py» відбувається уся взаємодія з профілем користувача, файл «search_processor.py» відповідає за роботу пошуку за різними сутностями на ресурсі, файл «tests_processor.py» допомагає в обробці та показі тестів після кожного з уроків, а також модулів після закінчення курсу. У файлах обробки

сутностей та профілю також відбувається регулювання та налаштування кешу сторінок у СУБД Redis.

3.2 Уявлення про класи ІС

Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

На рис. 3.2 зображено діаграму класів системи, що показує організацію та взаємодію сутностей бази даних у СУБД PostgreSQL.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Усі моделі, окрім моделі користувача (Person), успадковуються від базового класу Django Models, а модель користувача успадковується від класу User, при цьому об’єкт класу Person не може бути адміністратором системи, на відміну від об’єкту класу User.

3.3 Інструкція користувача

Реєстрація нового користувача (рис. 3.3). У головному меню ресурсу натиснути кнопку "Just Start". Заповнити усі поля, дотримуючись формату кожного поточного поля. Натиснути на кнопку "Sign up". Якщо усі поля заповнені коректно, користувач буде перенаправлений на сторінку показу профілю. Якщо якісь поля заповнені некоректно, поряд з ними з'явиться

повідомлення про те, що необхідно змінити для вдалого завершення реєстрації. Повторити пункт 3 після виконання змін даних.

Авторизація у ресурсі (рис. 3.4).

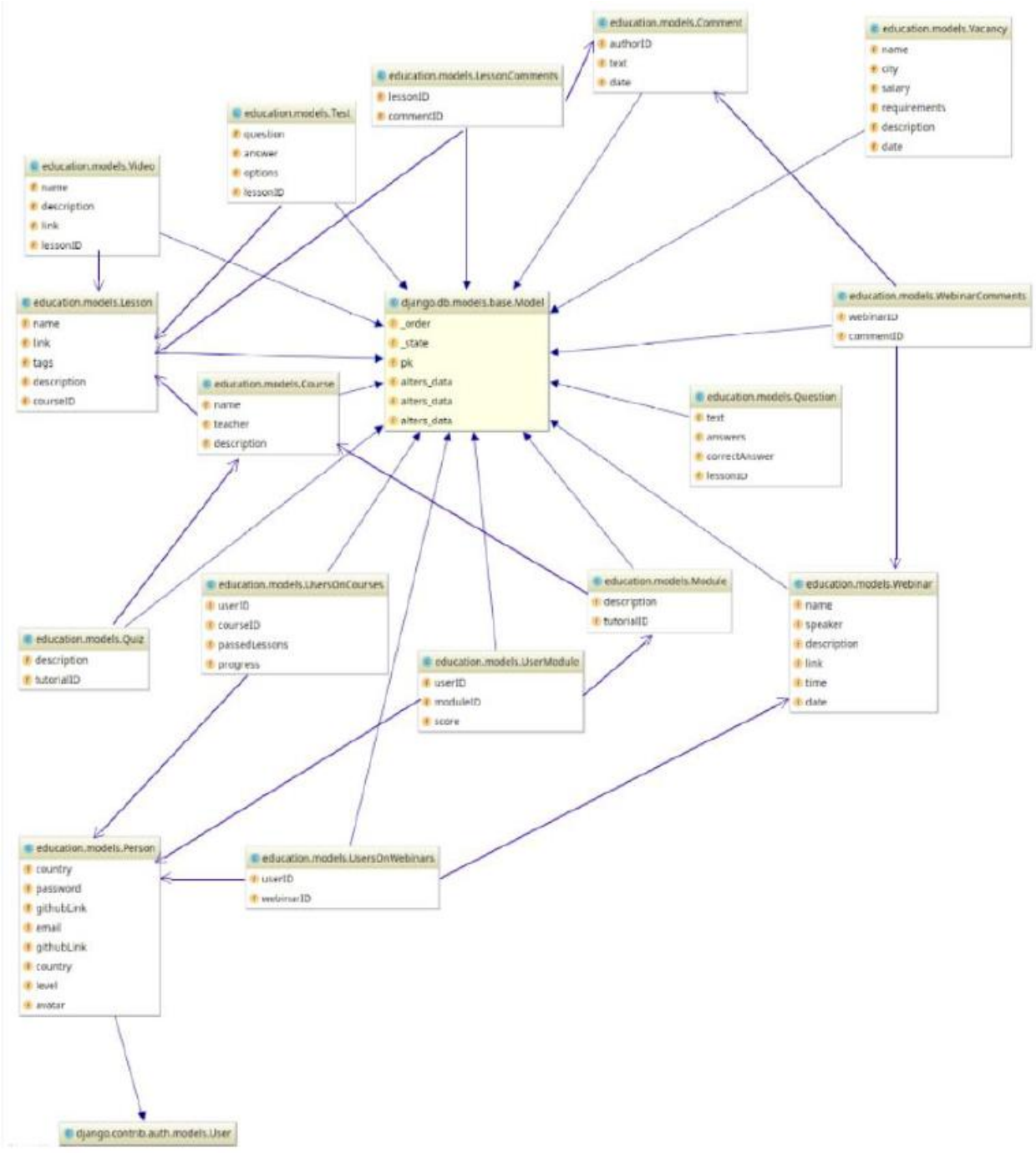

Рисунок 3.2 – Діаграма класів

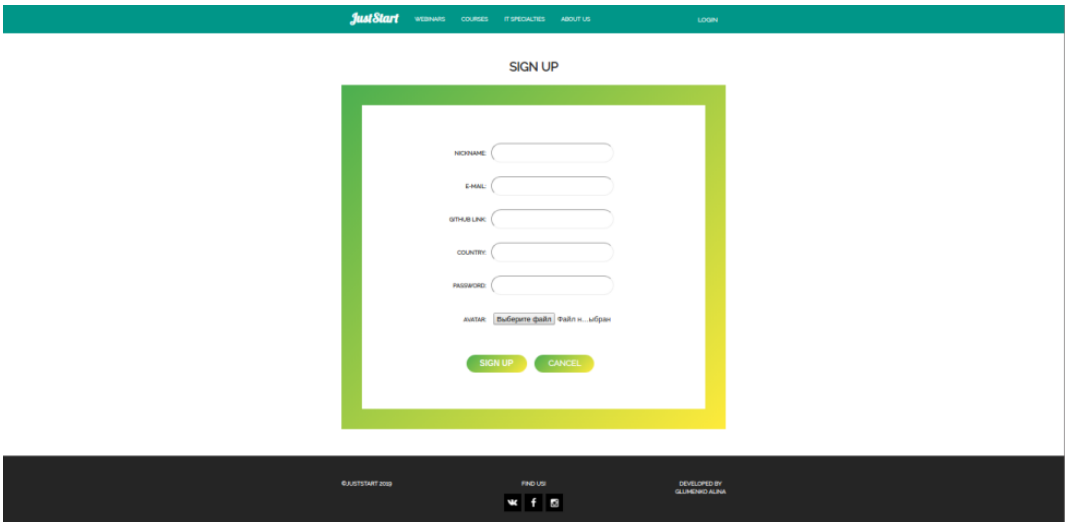

Рисунок 3.3 – Сторінка реєстрації

Натиснути на кнопку "Login" на панелі меню з правої сторони. У формі, що відкрилася заповнити поля "Username" та "Password". Натиснути на кнопку "Sign in". Якщо з'явиться попередження, про те, що пароль не є вірним, необхідно ввести дані ще раз та виконати їх перевірку. Якщо з'явиться попередження, про те, що такого користувача не існує, необхідно перевірити введені дані або нажати на кнопку "Sign up" та зареєструвати нового користувача. Якщо усі поля були заповнені вірно, користувач буде перенаправлений на сторінку показу профілю.

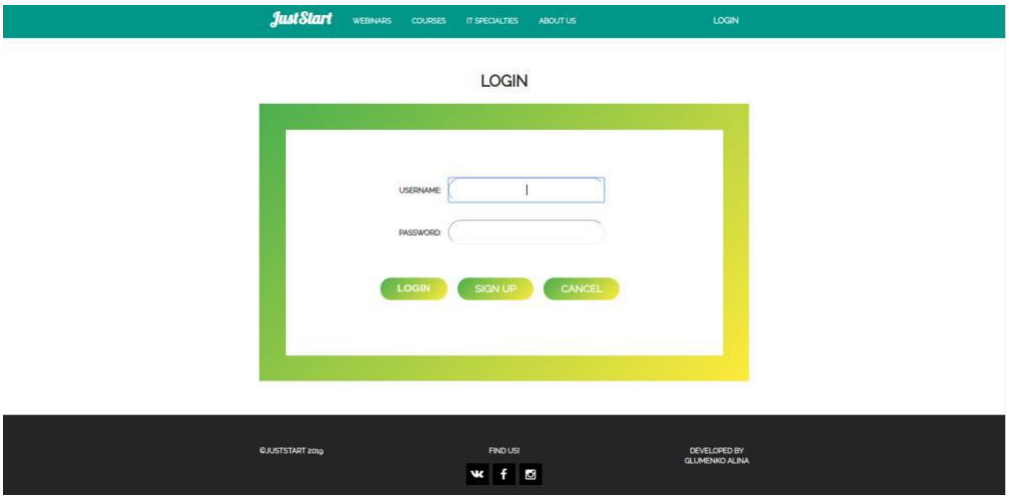

Рисунок 3.4 – Сторінка авторизації

Редагування профілю користувача (рис. 3.5). Якщо на панелі меню з правої сторони є кнопка "Login", користувачу необхідно виконати авторизацію або реєстрацію. В іншому випадку необхідно натиснути на другу кнопку правої частини панелі меню. У формі, що відкрилася, необхідно змінити лише ті поля, які потрібно, а усі інші залишити без змін. Поле паролю відображається пустим, але якщо ввести туди якісь дані, вони будуть збережені у якості паролю. При цьому необхідно дотримуватися формату даних у кожному з полів, щоб уникнути необхідності редагування введених даних. Натиснути на кнопку "Save". Якщо усі поля заповнені коректно, користувач буде перенаправлений на сторінку показу профілю. Якщо якісь поля заповнені некоректно, поряд з ними з'явиться повідомлення про те, що необхідно змінити для вдалого завершення редагування профілю. Повторити пункт 3 після виконання редагування даних.

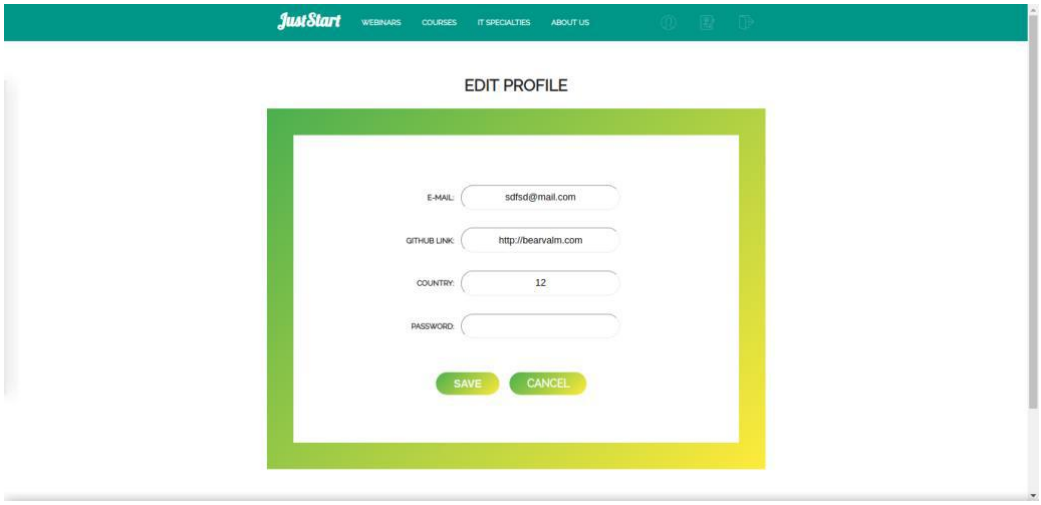

Рисунок 3.5 – Сторінка редагування профілю

Виконання пошуку за курсами та уроками. Перейти на сторінку показу списку курсів. У полі пошуку ввести необхідний запит. Натиснути на кнопку "Search". Якщо курси або уроки знайдені за описом чи назвою, то буде показаний список курсів, якому відповідають ці уроки. Якщо нічого не було знайдено, користувачу буде видане повідомлення про неуспішний пошук.

Рисунок 3.6 – Сторінка пошуку за курсами та уроками

3.4 Керівництво адміністратора ІС

Основні команди роботи з ресурсом:

– «python manage.py migrate» команда, що допомогає ініціалізувати базу даних на сервері;

– «python manage.py collectstatic» команда каже Django про усі статичні файли на сервері, які йому потрібні;

– «python manage.py runserver» команда запуску сервера;

– «CONTROL-C» комбінація клавіш які зупинять сервер.

Важливими файлами налаштувань в проекті являються «settings.py» та «docker-compose.yml». «docker-compose.yml» – файл налаштувань Docker. Це інструмент для визначення і запуску багатоконтейнерних додатків Docker. З Compose використовується файл YAML для налаштування служб програми. Потім за допомогою однієї команди створюються і використовуються всі служби з вашої конфігурації. Compose працює у всіх середовищах: виробництво, підготовка, розробка, тестування, а також робочі процеси CI. Використання Compose в основному складається з трьох етапів:

– визначення середи застосування, Dockerfile щоб воно могло бути відтворено де завгодно.

– визначення сервісів, з яких складається застосунок, docker-compose.yml щоб вони могли працювати разом в ізольованому середовищі.

– запуск docker-compose та починає і Compose запускає всю програму.

Compose має команди для управління всім життєвим циклом застосування:

– запуск, зупинка і перебудовування служби;

– перегляд статусу запущених сервісів;

– потік виведення журналу запущених служб;

– запуск одноразової команди в службі.

Файл «settings.py» – файл налаштувань Django містить повну конфігурацію встановленого проекту. По суті, це просто модуль Python зі змінними модуля.

При використанні Django необхідно вказати які налаштування використовуються. Для цього використовується змінна оточення DJANGO_SETTINGS_MODULE у файлі «manage.py». Значення DJANGO_SETTINGS_MODULE повинно містити шлях імпорту Python, наприклад composer.settings. Проект Django не зобов'язаний визначати будь-які настройки, якщо в цьому немає необхідності. Кожна настройка містить значення за замовчуванням. Ці значння можна знайти в модулі django/conf/global_settings.py.

Django використовує наступний алгоритм для завантаження налаштувань:

– завантажує настройки з global_settings.py;

– завантажує настройки із зазначеного файлу налаштувань, переписуючи значення за замовчуванням.

Є простий спосіб дізнатися, які налаштування проекту відрізняються від налаштувань за замовчуванням. Команда «python manage.py diffsettings» показує різницю між поточними настройками і настройками за замовчуванням Django.

У проекті використовувалися наступні змінні:

– змінна «DEBUG» вмикає/вимикає режим налагодження. За замовчуванням: False. Встановлено: True.

– змінна «ALLOWED_HOSTS» – список хостів/доменів, для яких може працювати поточний сайт. Це зроблено для безпеки, щоб убезпечити від впровадження в куки або листи для скидання пароля посилань на сторонній сайт підмінивши HTTP заголовок Host, що можливо при багатьох, здавалося б безпечних, конфігураціях сервера. За замовчуванням: [ ] (Порожній список).

– змінна «INSTALLED_APPS» – список рядків, який вказують не всі програми Django, використовувані в проекті. Кожен рядок повинен бути повним Python шляхом до класу налаштування програми (рекомендується), або пакету з додатком. За замовчуванням: [ ] (Порожній список).

– змінна «MIDDLEWARE» – список, який складається з шляхів для імпорту використовуваних опцій шарів.

– змінна «ROOT_URLCONF» – шлях для імпорту Python-модуля з головною зміною URL-ів. Може бути перевизначена для конкретного запиту зміною атрибута urlconf об'єкта HttpRequest.

– змінна «TEMPLATES» – список налаштувань для шаблонізаторів, які використовуються Django. Кожен елемент – це словник з параметрами налаштування шаблонізатора.

– змінна «WSGI_APPLICATION» – повний Python шлях до об'єкта WSGI додатку, яке буде використовувати вбудований сервер Django (наприклад runserver). Команда django-admin startproject створить простий wsgi.py файл з функцією application, і встановить значення цієї настройки на цей об'єкт application.

– змінна «DATABASES» – словник містить настройки для всіх баз даних, які будуть використовуватися Django. Словник містить псевдоніми використовуваних баз даних і словник з настройками для кожної. Налаштування DATABASES повинні визначати базу даних з псевдонімом default і будь-

яку кількість додаткових баз даних. При підключенні до інших типів баз даних, таких як MySQL, Oracle або PostgreSQL, необхідні додаткові параметри.

– змінна «CACHES» – словник, що містить налаштування для всіх механізмів кешування Django, використовуваних в проекті. Містить псевдоніми кешей і словник з настройками для кожного. CACHES повинна визначати кеш default і будь-яку кількість додаткових. При визначенні будь-якого кеша крім кешування в пам'яті, потрібно вказати додаткові параметри.

– змінна «SESSION_ENGINE» – вказує, де Django зберігає сесійні дані.

– змінна «SESSION_CACHE_ALIAS» – при використанні кешуючого бекенд для сесії, вказує який кеш використовувати.

– змінна «AUTH_PASSWORD_VALIDATORS» – список валідаторів, які перевіряють надійність пароля користувача. За замовчуванням перевірка не використовується і приймаються всі паролі.

– змінна «LANGUAGE_CODE» – код використовуваного в проекті мови. Повинен відповідати формату скорочень назв мов. Наприклад, U.S. English позначається як "en-us".

– змінна «TIME_ZONE» – часовий пояс, який буде використовуватися в проекті, або None. Зазначений часовий пояс не зобов'язаний збігатися з часовим поясом сервера. Наприклад, один сервер може обслуговувати кілька Django-проектів, кожен може використовувати свій часовий пояс.

– змінна «USE_I18N» – вказує, чи використовується механізм перекладу Django. Дозволяє легко відключити його для підвищення продуктивності. При False, Django виконає невелику оптимізацію щоб не завантажувати механізм перекладу.

– змінна «USE_L10N» – вказує, чи використовувати локалізований формат дати. При True, наприклад, Django буде відображати числа і дати в форматі поточної локалі.

– змінна «USE_TZ» – вказує, чи використовується часовий пояс. При True, Django буде використовувати об'єкти дати і часу з зазначеним часовим

поясом. Інакше Django буде використовувати об'єкти дати і часу без обліку часового поясу.

– змінна «STATIC_URL» – URL, який вказує на каталог зі статичними файлами STATIC_ROOT.

– змінна «STATIC_ROOT» – абсолютний шлях до каталогу, в який collectstatic збере всі статичні файли.

– змінна «LOGIN_REDIRECT_URL» – URL куди перенаправляється користувач після авторизації в представленні contrib.auth.login, якщо не переданий параметр next.

– змінна «CACHE_TTL» – час кешування становить 15 хвилин (60*15).

Більш детальне пояснення налаштувань цих файлів можна знайти у документаціях для Django та Docker.

ВИСНОВКИ

У процесі написання кваліфікаційної роботи був, розроблений інформаційний освітній ресурс для вивчення IT технологій. Дана система складається з 2 частин – це сервер та web-додаток. Веб-сайт є безкоштовним освітнім ресурсом для навчання різним мовам програмування і технологій, що використовуються при розробці.

У першому розділі даної роботи було визначено та проаналізовано предметну область. Далі були обрані технології. Клієнт був написаний на мові програмування JavaScript. Сервер був написаний на мові програмування Python. Також були обрані архітектурні шаблони.

У другому розділі були сформульовані мета та головні задачі інформаційної системи. Розроблено макети користувацького інтерфейсу для онлайнового навчального ресурсу, а також діаграму станів, що описує переходи між ними. Розроблено діаграми логічного представлення ІС та представлення розгортання, що відображають взаємодію компонентів усієї системи.

У третьому розділі була представлена структура кожної з частин проекту, а також побудовані до них діаграми класів. Загалом було здійснено реалізацію системи.

Таким чином, мета та усі поставлені до даної роботи завдання були успішно виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Python – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 18.05.2022)
2. Django Project – Офіційний сайт. URL: https://www.djangoproject.com/ (дата звернення: 22.05.2022).
3. Docker – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Docker (дата звернення: 22.05.2022).
4. Хейз Д. Освой самостоятельно HTML и XHTML. С–П.: Вильямс, 2003. 224 с.
5. Що таке CSS. URL: http://phpist.com.ua/css/5-whatcss/ (дата звернення: 18.05.2022).
6. Мова JavaScript та її можливості. URL: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti (дата звернення: 08.06.2021) (дата звернення: 18.05.2022).
7. How to use PostgreSQL with Django. URL: https://www.enterprisedb.com/postgres-tutorials/how-use-postgresql-django (дата звернення: 18.05.2022).
8. Django Elasticsearch DSL. URL: https://django-elasticsearch-dsl.readthedocs.io/en/latest/ (дата звернення: 19.05.2022).
9. Django-Redis Documentation. URL: https://readthedocs.org/projects/django-redis-chs/downloads/pdf/latest/ (дата звернення: 19.05.2022).
10. Redis в Python – Повна документація на прикладах. URL: https://python-scripts.com/redis (дата звернення: 19.05.2022).
11. Web Scraping in Python using Scrapy. URL: https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/ (дата звернення: 19.05.2022).
12. Documentation – PostgreSQL. URL: https://www.postgresql.org/docs/ (дата звернення: 19.05.2022).

13. Elasticsearch – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Elasticsearch (дата звернення: 18.05.2022)
14. Redis – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Redis (дата звернення: 18.05.2022)
15. MVC – модель-представление-контроллер : стаття. URL: https://bit.ly/3542MMC (дата звернення: 19.05.2022).
16. Многослойная архитектура: стаття. URL: https://bit.ly/2TkNRuI (дата звернення: 19.05.2022). Паттерн MVVM: стаття. URL: https://bit.ly/3g9bwYc (дата звернення: 19.05.2022).
17. Coffeescript – Офіційний сайт. URL: https://bit.ly/2Suv684 (дата звернення: 20.05.2022).
18. TypeScript – Офіційний сайт. URL: https://bit.ly/3bOzCET (дата звернення: 20.05.2022).

ДОДАТОК А
КОД ПРОГРАМИ

Файл urls.py

```
from django.contrib import admin
from django.urls import path,include
urlpatterns = [
path('admin/', admin.site.urls),
#path to djoser end points
path('auth/', include('djoser.urls')),
path('auth/', include('djoser.urls.jwt')),
#path to our account's app endpoints
path("api/accounts/",include("accounts.urls"))
]
```

Файл models.py

```
from django.db import models
from django.contrib.auth.models import User
class AuthUserModel(models.Model):
us-
er=models.OneToOneField(User,on_delete=models.CASCADE,related_name=
"profile")
description=models.TextField(blank=True,null=True)
location=models.CharField(max_length=30,blank=True)
date_joined=models.DateTimeField(auto_now_add=True)
updated_on=models.DateTimeField(auto_now=True)
is_organizer=models.BooleanField(default=False)
def __str__(self):
return self.user.username
```

Файл signals.py

```
from django.contrib.auth.models import User 90
```

```
from django.db.models.signals import post_save
from django.dispatch import receiver
from .models import userProfile
@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
if created:
userProfile.objects.create(user=instance)
from django.apps import AppConfig
class AccountsConfig(AppConfig):
name = 'accounts'
def ready(self):
import accounts.signals
```

Файл serializers.py

```
from rest_framework import serializers
from .models import userProfile
class userProfileSerializer(serializers.ModelSerializer):
user=serializers.StringRelatedField(read_only=True)
class Meta:
model=userProfile
fields=’__all__’
```

Файл views.py

```
from rest_framework.generics import
(ListCreateAPIView,RetrieveUpdateDestroyAPIView,)
from rest_framework.permissions import IsAuthenticated
from .models import userProfile
from .permissions import IsOwnerProfileOrReadOnly
from .serializers import userProfileSerializer 91
```

```
class UserProfileListCreateView(ListCreateAPIView):
queryset=userProfile.objects.all()
serializer_class=userProfileSerializer
permission_classes=[IsAuthenticated]
def perform_create(self, serializer):
user=self.request.user
serializer.save(user=user)
class userProfileDetailView(RetrieveUpdateDestroyAPIView):
queryset=userProfile.objects.all()
serializer_class=userProfileSerializer
permission_classes=[IsOwnerProfileOrReadOnly,IsAuthenticated]
from django.urls import include, path
from rest_framework.routers import DefaultRouter
from .views import UserProfileListCreateView, userProfileDetailView
urlpatterns = [
#gets all user profiles and create a new profile
path("all-profiles",UserProfileListCreateView.as_view(),name="all-
profiles"),
# retrieves profile details of the currently logged in user
path("profile/<int:pk>",userProfileDetailView.as_view(),name="profi
le"),
]
```

Файл license.py

```
from rest_framework.permissions import BasePermission,SAFE_METHODS
class IsOwnerProfileOrReadOnly(BasePermission):
def has_object_permission(self, request, view, obj):
if request.method in SAFE_METHODS:
return True
return obj.user==request.user 92
```

Файл test.py

```
class userProfileTestCase(APITestCase):
profile_list_url=reverse('all-profiles')
def setUp(self):
# создайте нового пользователя, отправив запрос к конечной точке
djoser
self.user=self.client.post('/auth/users/',data={'username':'mario',
'password':'i-keep-jumping'})
# получить веб-токен JSON для вновь созданного пользователя
re-
sponse=self.client.post('/auth/jwt/create/',data={'username':'mario
','password':'i-keep-jumping'})
self.token=response.data['access']
self.api_authentication()
def api_authentication(self):
self.client.credentials(HTTP_AUTHORIZATION='Bearer '+self.token)
# получить список всех профилей пользователей во время аутентифика-
ции пользователя запроса
def test_userprofile_list_authenticated(self):
response=self.client.get(self.profile_list_url)
self.assertEqual(response.status_code,status.HTTP_200_OK)
# получить список всех профилей пользователей, пока запрос пользо-
вателя не прошел проверку подлинности
def test_userprofile_list_unauthenticated(self):
self.client.force_authenticate(user=None)
response=self.client.get(self.profile_list_url)
self.assertEqual(response.status_code,status.HTTP_401_UNAUTHORIZED)
# проверьте, чтобы получить данные профиля аутентифицированного
пользователя
def test_userprofile_detail_retrieve(self):
response=self.client.get(reverse('profile',kwargs={'pk':1}))
# print(response.data)
self.assertEqual(response.status_code,status.HTTP_200_OK)
# заполнить профиль пользователя, который был автоматически создан
с использованием
```

```
сигналов
def test_userprofile_profile(self):
profile_data={'description':'I am a very famous game
character','location':'nintendo world','is_creator':'true',}
re-
sponse=self.client.put(reverse('profile',kwargs={'pk':1}),data=prof
ile_data)
print(response.data)
self.assertEqual(response.status_code,status.HTTP_200_OK)
```

ДОДАТОК Б
МАКЕТИ ГРАФІЧНОГО ІНТЕРФЕЙСУ

1) Головна сторінка ресурсу:

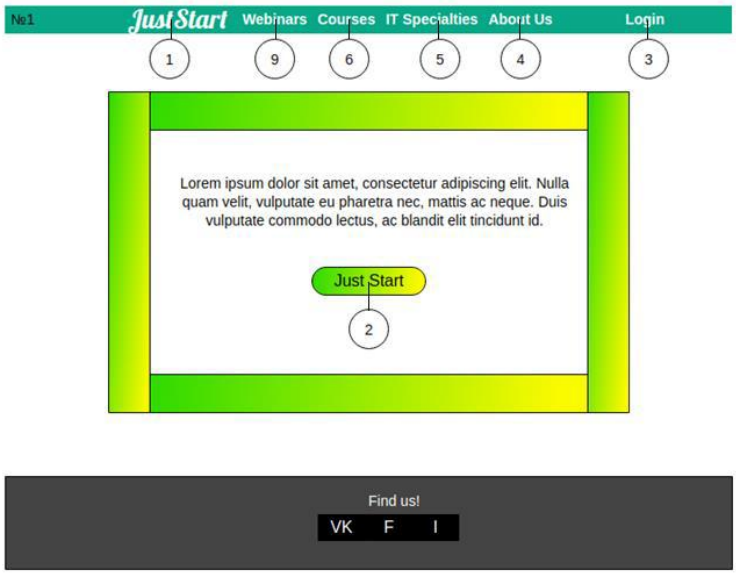

Рисунок Б.1 – Головна сторінка

2) Сторінка реєстрації:

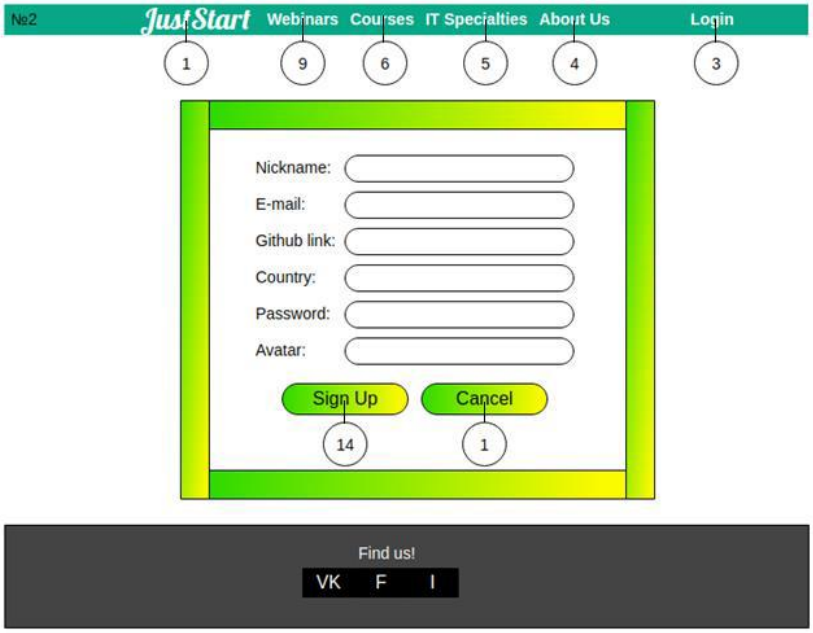

Рисунок Б.2 – Сторінка реєстрації

3) Сторінка логіну:

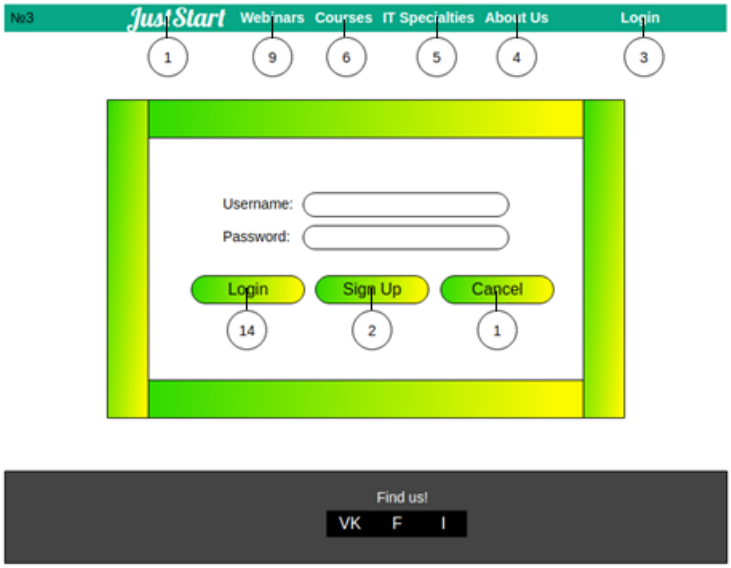

Рисунок Б.3 – Сторінка логіну

4) Сторінка інформації про ресурс:

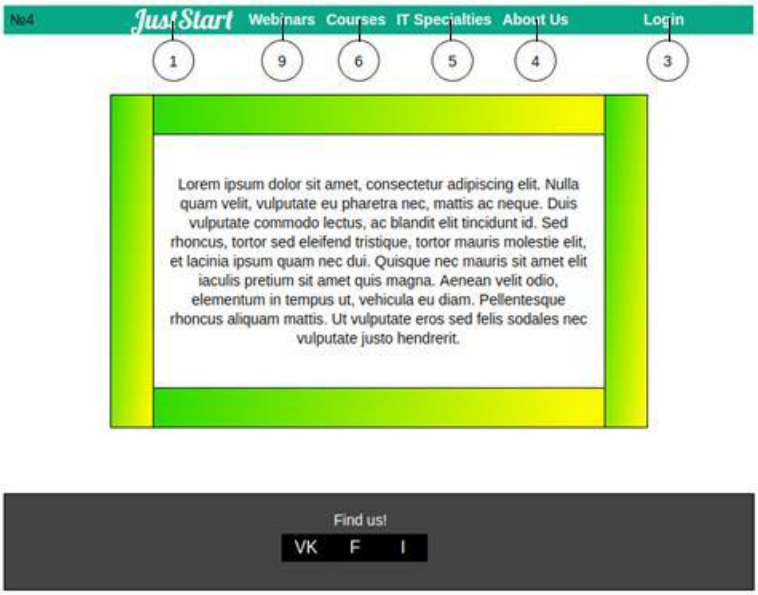

Рисунок Б.4 – Сторінка інформації про ресурс

5) Сторінка зі списком вакансій:

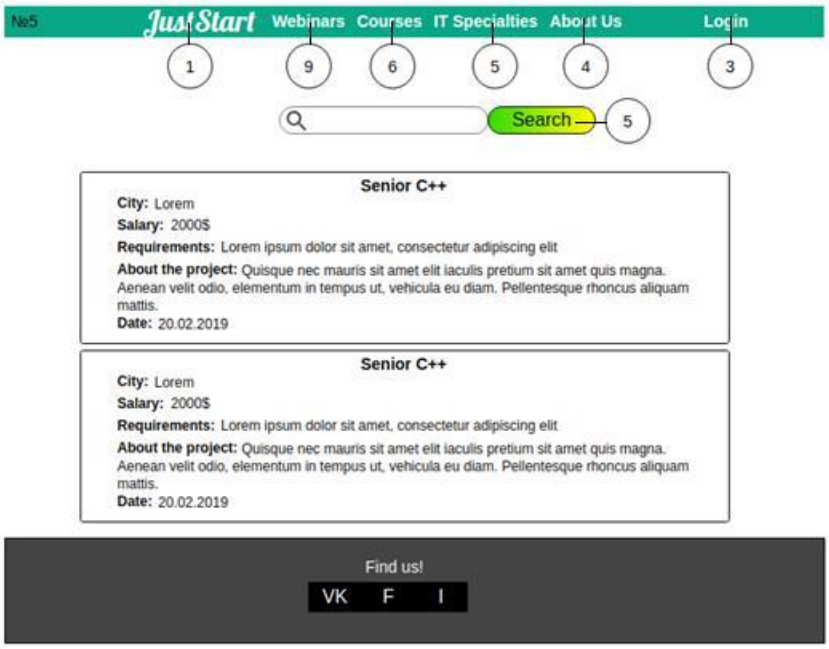

Рисунок Б.5 – Сторінка зі списком вакансій

6) Сторінка списку курсів:

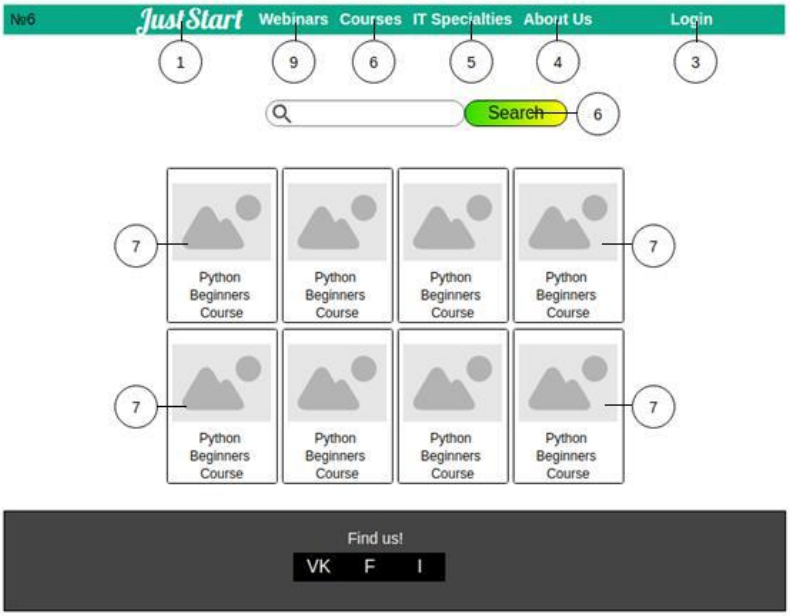

Рисунок Б.6 – Список курсів

7) Сторінка вступу курсу:

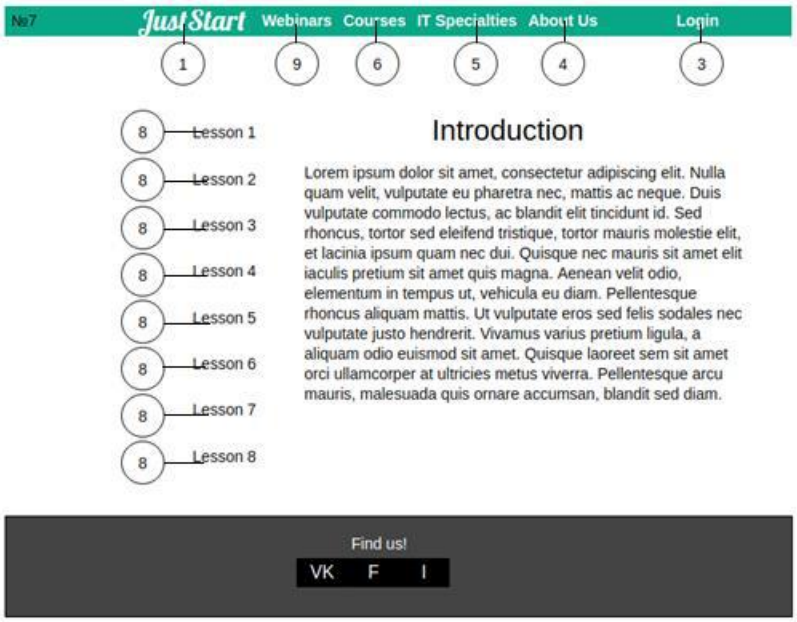

Рисунок Б.7 – Початкова сторінка курсу

8) Сторінка уроку:

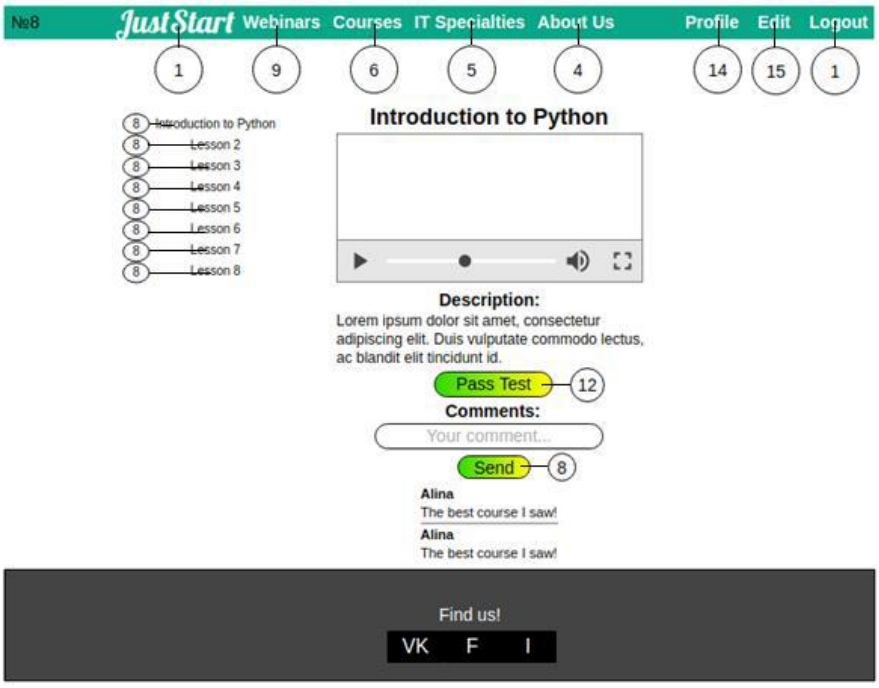

Рисунок Б.8 – Сторінка уроку

9) Сторінка зі списком вебінарів:

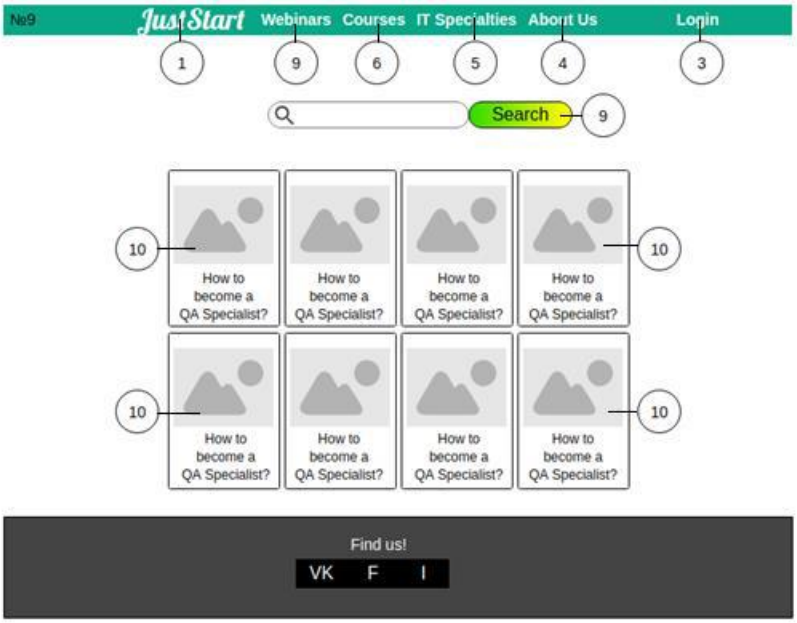

Рисунок Б.9 – Сторінка зі списком вебінарів

10) Сторінка зі вступом вебінару:

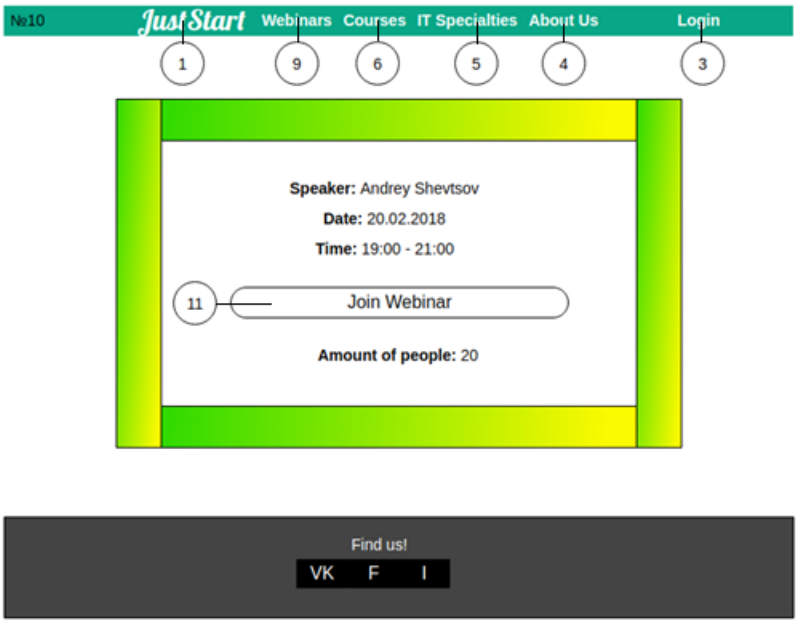

Рисунок Б.10 – Сторінка зі вступом вебінару

11) Сторінка вебінару:

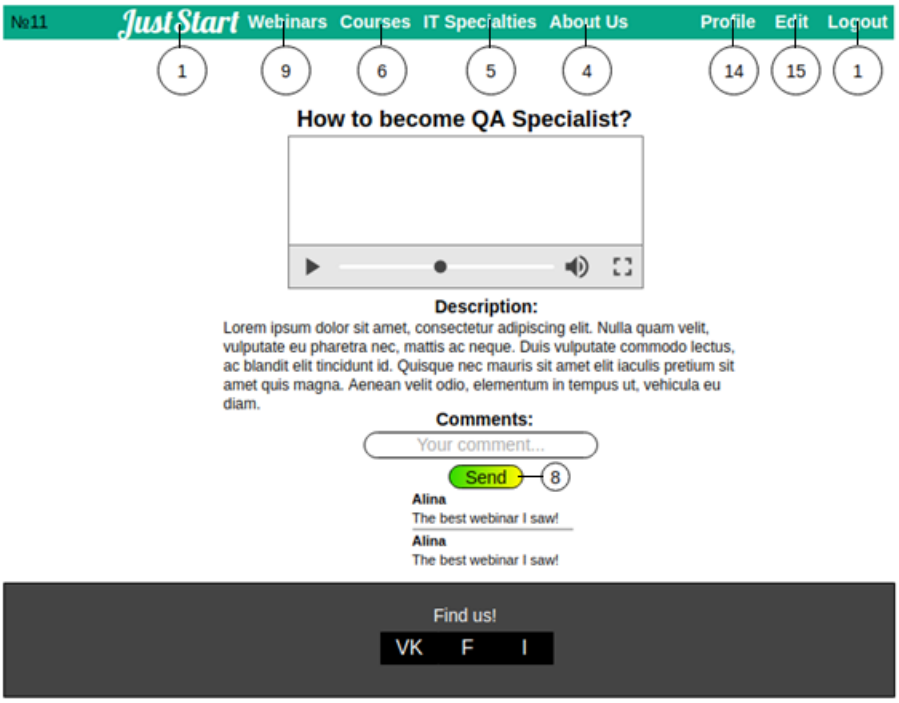

Рисунок Б.11 – Сторінка вебінару

12) Сторінка з тестами:

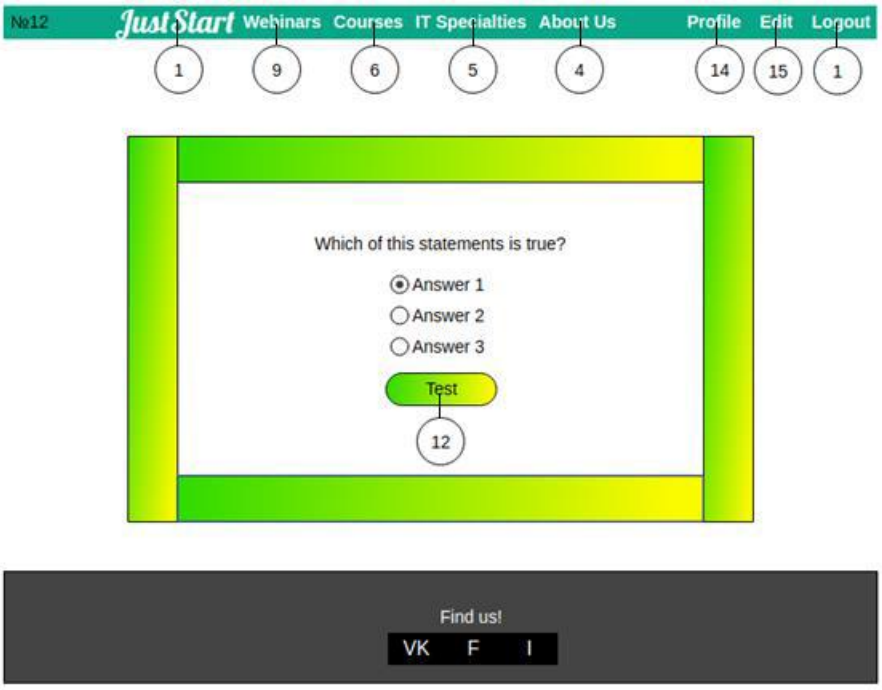

Рисунок Б.12 – Сторінка тестування

13) Сторінка з результатом тестування:

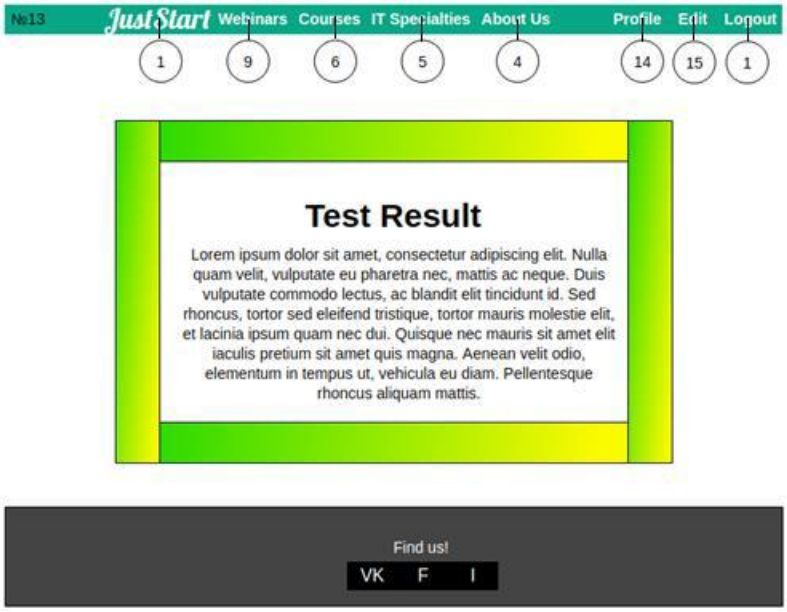

Рисунок Б.13 – Сторінка з результатом тестування

14) Сторінка профілю:

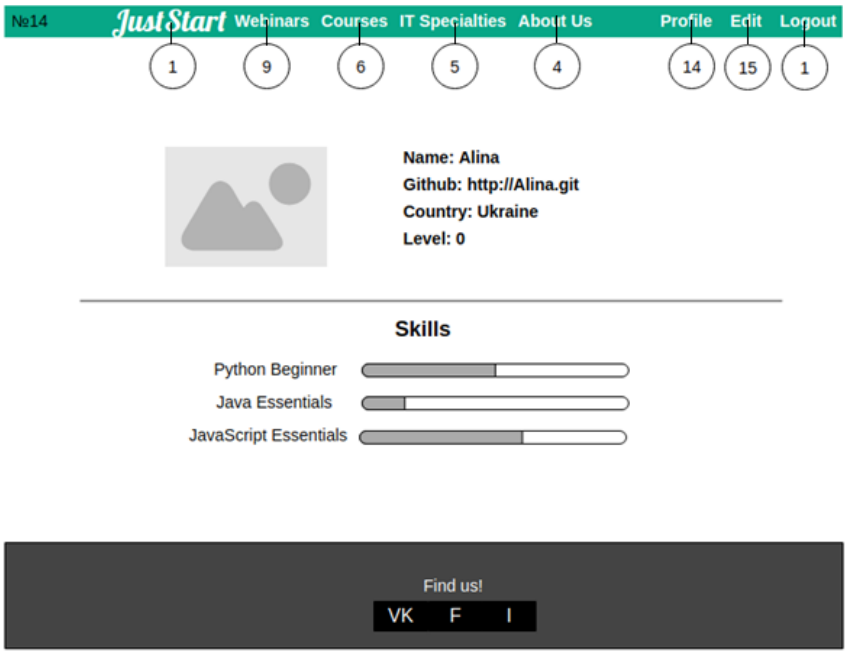

Рисунок Б.14 – Профіль користувача

15) Сторінка редагування профілю:

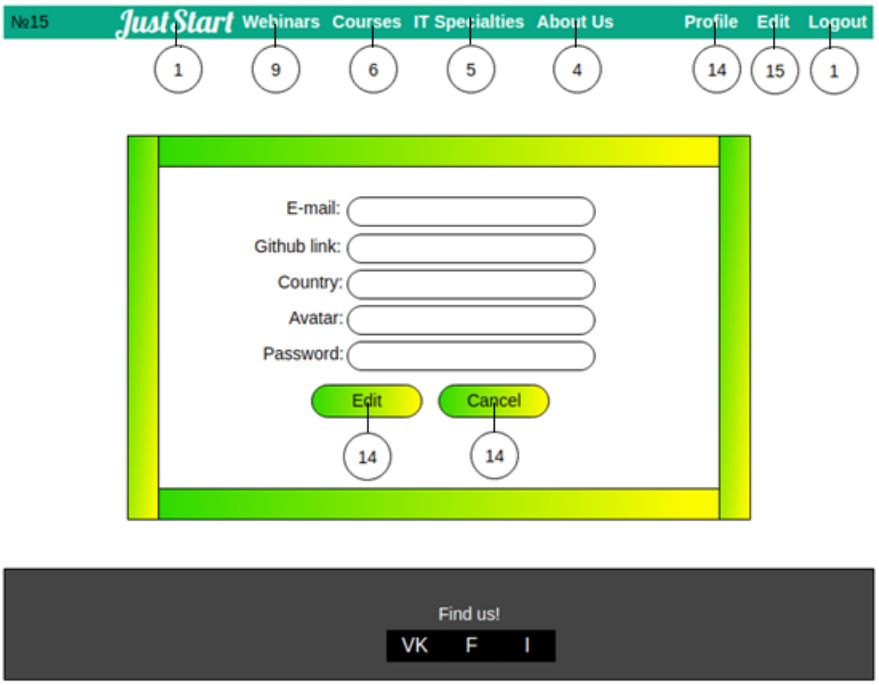

Рисунок Б.15 – Сторінка редагування профілю