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

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

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерний моніторинг та геометричне моделювання процесів та систем»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Адміністрування програмного продукту «Система підготовки науково – педагогічних кадрів» відділу аспірантури і докторантури університету»

Виконала:
студентка IV курсу, групи ТР-72 _____
Рожко Тетяна Юріївна

Керівник:
асистент,
Касьянов Антон Сергійович _____

Рецензент:
к.т.н., доцент кафедри ТЕУТ та АЕС
Сірий О.А.
_____

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

Київ – 2021 року

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

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

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

ЗАВДАННЯ

на дипломну роботу студенту

Рожко Тетяні Юріївні
(прізвище, ім’я, по батькові)

1. Тема роботи _____
Адміністрування програмного продукту «Система підготовки науково-педагогічних кадрів» відділу аспірантури і докторантури університету»_____
керівник роботи Касьянов Антон Сергійович, асистент_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи
мова програмування JavaScript, програмна платформа Firebase, середовище розробки Visual Studio Code_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
проаналізувати проблему вступу спеціалістів науково-педагогічних кадрів вищої кваліфікації, розробити систему супроводження та аналізу вступу спеціалістів науково-педагогічних кадрів вищої кваліфікації, виконати зміни в програмній системі відповідно до вимог адміністрації відділу аспірантури_____
5. Перелік ілюстративного матеріалу
схеми архітектури програмного продукту, знімки інтерфейсу, знімки структури

проекту, схема бази даних_____
7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	9.10.2020	
2.	Вивчення та аналіз задачі	15.10.20-01.02.21	
3.	Розробка архітектури та загальної структури системи	1.02.21-15.02.21	
4.	Розробка структур окремих підсистем	15.02.21-21.02.21	
5.	Програмна реалізація системи	15.04.2021-1.05.21	
6.	Оформлення пояснювальної записки	12.03.2021 - 04.06.2021	
7.	Захист програмного продукту	12.05.2021	
8.	Передзахист	26.05.2021	
9.	Захист	15.05.2021	

Студентка _____ _____Рожко Т.Ю._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Касьянов А.С._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Рожко Тетяна Юріївна

Тема: Адміністрування програмного продукту «Система підготовки науково – педагогічних кадрів» відділу аспірантури і докторантури університету.

Спеціальність: 122 Комп'ютерні науки та інформаційні технології.

Установа: Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», 2021 р.

Бакалаврська робота: 78 с., вступ, 3 розділів, висновок, 21 джерело, 5 додатків, 31 рисунків.

Актуальність теми. Робота відділу аспірантури та докторантури університету пов'язана з накопиченням великої кількості інформації. Велика кількість інформації зберігається на паперових носіях. Таким чином, автоматизація процесу роботи відділу аспірантури та докторантури є необхідним та перспективним процесом.

Метою дослідження. Метою дослідження є розробка програмного продукту у вигляді web-додатка, який буде надавати можливість створення, відстежування заяв для вступу до третього рівня вищої освіти, а також їх обробка та аналіз.

Об'єктом дослідження є web-система відділу аспірантури і докторантури університету.

Предметом дослідження є програмна система вступу до третього рівня вищої освіти, аккаунти адміністратора і вступної комісії.

Результати роботи: розроблено систему супроводження та аналізу вступу спеціалістів науково-педагогічних кадрів вищої кваліфікації; виконані зміни в програмній системі відповідно до вимог адміністрації.

Ключові слова: АСПІРАНТУРА; РЕАКТИВНЕ ПРОГРАМУВАННЯ, ФРЕЙВОРК ANGULAR; ВЕБ-ДОДАТОК; FIREBASE.

ANNOTATION

Rozhko Tetiana Yuriivna

Topic: Administration of the software product "System of training of scientific and pedagogical staff" of the department of postgraduate and doctoral studies of the university.

Specialty: 122 Computer Science and Information Technology.

Institution: National Technical University of Ukraine "Kyiv Polytechnic Institute. Igor Sikorsky ", 2021.

The purpose of the bachelor's work is 78 pages, introduction, 3 sections, conclusion, 21 source, 5 applications, 31 drawings.

Actuality of theme. The work of the department of postgraduate and at the university is associated with the accumulation of a large amount of information. A large amount of information is stored on paper. Thus, the automation of the process of postgraduate and doctoral studies is a necessary and promising process.

The purpose of the study. The purpose of the study is to develop a software product in the form of a web-application that will provide the ability to create, track applications for admission to the third level of higher education, as well as their processing and analysis.

The object of research is the web-system of the department of postgraduate and doctoral studies of the university.

The subject of the study is the software system for admission to the third level of higher education, administrator accounts and admission commission.

Results of work: the system of support and the analysis of entrance of experts of scientific and pedagogical shots of the highest qualification is developed; changes in the program system according to requirements of administration were executed.

Key words: POSTGRADUATE STUDIES; JET PROGRAMMING, ANGULAR FRAMEWORK; WEB APPLICATION; FIREBASE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система.

HTML – мова розмітки web-сторінки.

CSS – каскадна таблиця стилів.

HTTP – протокол прикладного рівня передачі даних.

FTP – протокол передачі даних по мережі.

DOM – це об’єктне уявлення вихідного HTML-документа, спроба перетворити його структуру і вміст в об’єктну модель, з якої змогли б працювати різні програми.

SGML – метамова, на якій можна визначити мову розмітки для документів.

ВСТУП

Актуальність роботи. Робота відділу аспірантури та докторантури університету пов’язана з накопиченням великої кількості інформації. Велика кількість інформації зберігається на паперових носіях. Через це важко здійснити швидкий відбір потрібних даних під час роботи приймальної комісії. Таким чином, автоматизація процесу роботи відділу аспірантури та докторантури є необхідним та перспективним процесом.

Для розв’язку проблеми керування, супроводження, аналізу та задля полегшення вступу до третього рівня вищої освіти було створено систему, що надає можливість подавати заяви на навчання, ознайомлюватися з рейтинговою інформацію з обраного напрямку та факультету, а також розроблено адміністративну частину, завдяки якій можна переглядати кандидатів на вступ та оброблювати заяви за визначеним регламентом.

Програмна система має дві підсистеми – перша для користувача, друга для адміністраторів відділу аспірантури. Перша підсистема дає змогу вступникам в електронному вигляді створювати та надсилати заяву для вступу до третього рівня вищої освіти – аспірантури, відстежувати процес та рейтинг. Після зарахування, аспірант має змогу переглядати дані: оцінки, статистику, відмітки присутності, рейтинговий список тощо. Друга підсистема дозволяє адміністратору здійснювати моніторинг, контроль та аналіз процесу вступу. Наприклад, редагування та подання вхідних заяв вступників, внесення в систему залікових балів.

Розроблена програмна система покращує процес вступу, спрощує супроводження вступників, дає можливість автоматизувати ручну систему та зменшити обсяг паперових носіїв.

Практичне значення розробки – надання інтуїтивно-зрозумілого інтерфейсу для взаємодії вступника та представників адміністрації відділу аспірантури.

1 ПОСТАНОВКА ЗАДАЧІ

Метою роботи є розробка програмного продукту у вигляді web-додатка, який буде надавати можливість створення, відстеження заяв для вступу до третього рівня вищої освіти, а також їх обробка та аналіз.

Об'єктом дослідження є web-система відділу аспірантури і докторантури університету.

Предметом дослідження є програмна система для вступу до третього рівня вищої освіти, адміністрування системи, функціональність для вступної комісії відділу аспірантури.

Відповідно до мети, предмета дослідження, було визначено основні **завдання дослідження**:

- проектування структури web-додатку;
- визначення основних потреб та вимог до автоматизованого програмного продукту відділу аспірантури;
- аналіз процесу вступу на третій рівень вищої освіти
- проектування архітектури програмної системи;
- розробка програмного продукту;
- обробка зібраних матеріалів;
- аналіз можливості удосконалення існуючих процесів вступної кампанії
- тестування розробленої програмної системи (Додаток Г);
- написання звіту про зроблену роботу.

Для вирішення завдань дослідження, використано такі **методи дослідження**: теоретичні: аналіз наукової літератури, емпіричні: збір даних про основні вимоги для вступу до третього рівня вищої освіти; експериментальні: створення та тестування імітаційної моделі.

1.1 Основні задачі

Метою практики є розробка програмного продукту у вигляді веб-додатка, який буде надавати можливість створення, відстежування заяв для вступу до третього рівня вищої освіти, а також їх обробка та аналіз.

1.2 Загальні вимоги

Призначенням розробленого веб-додатка є надання інтуїтивно зрозумілого інтерфейсу для взаємодії вступника та відділу адміністрації.

Система повинена відповідати всім рекомендаціям дизайну та функціоналу. Додаток не має перевизначати або некоректно використовувати стандартні шаблони інтерфейсу користувача.

1.3 Головні задачі

Веб-додаток розрахований на 5 категорій користувачів: незареєстровані користувачі, аспіранти, адміністратор відділу аспірантури, адміністратор приймальної комісії, викладач.

Відповідно до категорій розподіляється відповідний функціонал. Незареєстровані користувачі можуть лише подати заяву на вступ до третього рівня вищої освіти. Аспіранти мають доступ до розкладу занять, відстежування балів за іспити з предметів, перегляд статистики. Викладачі також можуть вписувати бали аспірантів за іспити. Адміністратор приймальної комісії має можливість переглядати статистику, підтверджувати/відхиляти реєстрацію користувача, контролювати обробку поданих заяв. Адміністратор відділу аспірантури має змогу оновлювати або видаляти інформацію про факультети, освітні програми, кафедри, групи. Також має можливість редагувати освітні програми для спеціальностей та додавати нових викладачів.

2 ОПИС МЕТОДІВ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Перед початком програмної реалізації Web-системи третього рівня вищої освіти, необхідно провести дослідження теоретичних установ, які є її структурними компонентами. По-перше, потрібно роздивитися основні визначення та терміни, з якими оперує система, що створюється. По-друге, провести аналіз двох основних категорій системи – серверної та клієнтської частини.

Важливим пунктом для підготовки до розробки будь-якої програмної системи є вдалий вибір засобів для програмної реалізації. Для розробки Web-орієнтованої системи відділу аспірантури та докторантури, можна використовувати середовище Visual Studio Code, приклад робочої області якого наведено на рисунку 2.1. Це середовище надає необхідний інструментарій для роботи як із серверною, так і з клієнтською частиною програмної системи у розробці.

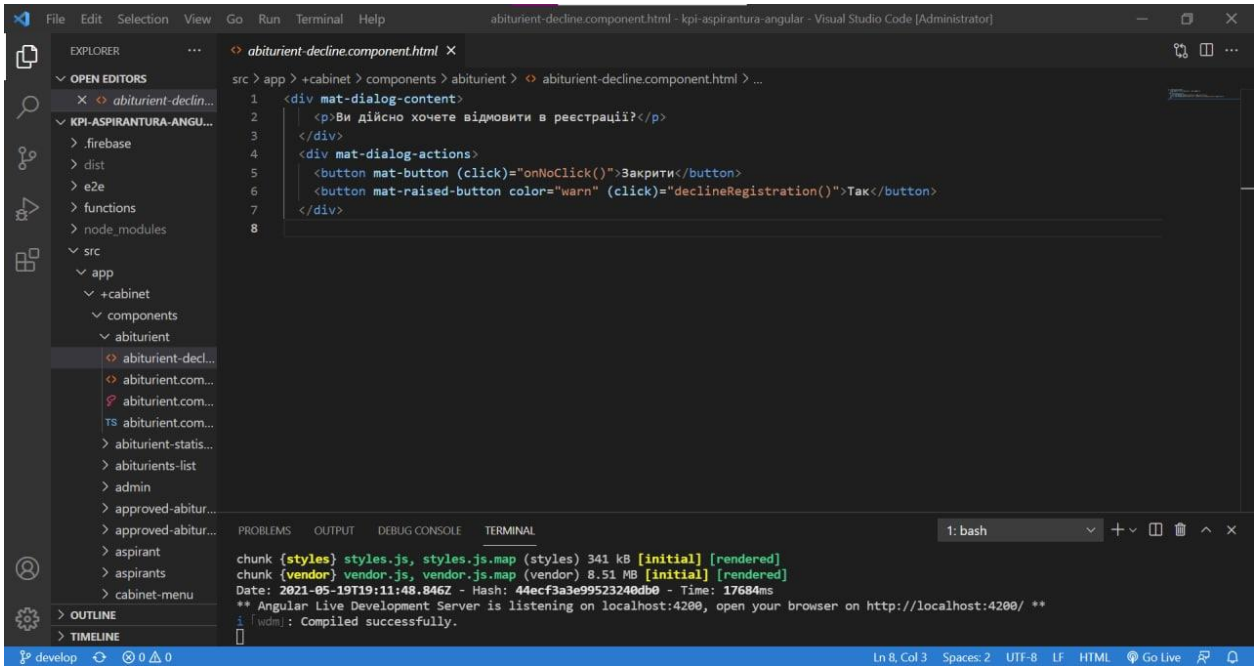


Рисунок 2.1 Робоча область Visual Studio Code

2.1 Основні визначення, терміни та поняття

Щоб розмістити інформацію в мережі, її необхідно представити у вигляді файлів, які програмами користувачів будуть визначені як сторінки документу – цей документ може бути призначений для читання, перегляду графіки чи відео, заповнення інтерактивних форм і інших цілей. Ці файли створюються за допомогою мови опису і розмітки гіпертексту HTML і називаються HTML-файлами.

Web-сторінки – це документ, який може отримати користувач в результаті читання HTML-файлу, використовуючи Web-оглядач. Web-сторінка може містити текст, графіку, посилання на інші документи і різні файли: текстові, графічні, аудіо та відео.

Web-оглядач – спеціальна програма, призначена для читання HTML-файлів і їх відображення. Сьогодні існує безліч таких програм, але найбільшу популярність отримали MS Edge, Opera, Chrome, Firefox. Web-оглядач може бути використаним для переходів з однієї Web-сторінку на іншу по гіперпосиланням, для завантаження файлів з Internet на комп’ютер, для відтворення впроваджених на Web-сторінку файлів мультимедіа, а також для запуску додатків.

Web-сайт – декілька Web-сторінок, об’єднаних в загальну структуру. Створення web-сайту – подія, яка підвищує імідж навчального закладу, спрощує його роботу. Це можливість продемонструвати свої досягнення, розміщувати актуальну інформацію для зацікавлений людей. У випадку розробленого програмної системи це викладачі, аспіранти та відділ адміністрації аспірантури університету. Використовуючи платформи різних університетів, абітурієнти та інші зможуть порівнювати їх і робити вибір на користь кращих.

Web-сервер – підключений до мережі Internet комп’ютер, який підтримує протокол HTTP. Він використовується для доставки даних в Internet і надає можливість користувачам програм завантажувати і відтворювати текст,

малюнки, звук, відео і інші дані. Протокол також є розповсюдженим способом завантаження файлів на віддалений сервер.

Гіперпосилання – це текстовий або графічний елемент Web-сторінки, який є покажчиком переходу на іншу Web-сторінку. Розрізняють внутрішні та зовнішні гіперпосилання. Внутрішні – забезпечують перехід на іншу частину даної сторінки, файл або іншу сторінку даного Web-сайту. Зовнішні – вказують на файл або сторінку іншого Web-сайту.

2.2 Обрані методи реалізації

Задля досягнення крос-платформенності системи, що розроблюється, попередньо було прийнято рішення про створення Web-додатку із реалізацією програмного продукту. Тому користувач має можливість використовувати його на будь-якій платформі, якщо на ній є змога використовувати web-браузери (Windows, Linux, macOS, iOS, Android тощо). Необхідно роздивитися клієнтську та серверну частини системи, визначивши і проаналізувавши компоненти кожної із них.

2.2.1 Клієнтська частина програмного продукту

Частина програмної системи, що запускається і працює на комп’ютері користувача є клієнтською. У випадку розробленого web-додатку, доступ до нього можна отримати через web-браузер. При реалізації програмної системи, були використані наступні web-технології розробки програмного забезпечення:

- HTML;
- CSS;
- SASS/SCSS;
- JavaScript;
- TypeScript;

- Angular;
- RxJS.

Мова тегів HTML – мова розмітки тексту, яка використовується для структурування web-сторінки та її вмісту. Наприклад, вміст може бути структурований у межах набору абзаців, списку маркованих точок або з використанням зображень та таблиць даних.

Розмітка HTML – перша і найголовніша мова, для початку вивчення web-розробки. Можна легко зрозуміти чужий код і, якщо потрібно, внести в нього зміни. Більше того, він не видає жодної помилки, як інші мови програмування, якщо розробник не закриє теги або допустить деякі помилки в коді.

HTML не є мовою програмування, тобто не дає можливості створювати динамічні функції. Замість цього він дозволяє організовувати і форматувати документи, аналогічно Microsoft Word [1, с. 19].

HTML є універсальною мовою розмітки документів SGML, що є стандартом для обміну документами між різними платформами. Точніше, весь синтаксис HTML повністю описується за допомогою SGML DTD. З цієї причини майже всі програми, сумісні з SGML, можуть бути використані при підготовці HTML-документів [2, с. 32].

HTML підтримують всі сучасні браузери. Web-додаток буде сумісним із всіма браузерами, крім того в програмі можна передбачити оптимізацію для різних браузерів.

Розробник використовує прості структури коду під час роботи з HTML – атрибути та теги, щоб розмістити сторінку web-додатку.

Остання версія цієї мови розмітки, HTML5 надає багато мультимедійних можливостей. Наприклад:

- використання відео- та аудіо- записів;
- малювання за допомогою структури Canvas;
- Drag’n’Drop;
- обробка помилок;

- доступ та використання геолокації;
- кросплатформна підтримка;
- нові елементи форм.

Спеціальна мова стилю сторінок CSS – мова для задання зовнішнього виду документу (web-сторінки), написаного з використанням мови розмітки – частіше за все HYML або XHTML. Також може бути задіяною до будь-яких XML-документів, наприклад, до SVG або XUL.

CSS може бути використаним будь-якою мовою на основі XML. Код, написаний на CSS, можна описати один раз, а потім повторно використовувати на декількох HTML-сторінках. Щоб внести глобальні зміни слід просто змінити стиль, і всі елементи будуть оновлені автоматично[3, с. 39].

CSS використовується розробниками web-сторінок для задання шрифтів, стилів, розташування окремих блоків, кольорів та інших аспектів. CSS обробляє дизайн сторінки і є її структурною частиною [4, с. 21]. Таке розподілення розмітки і стилів може надати розробнику велику гнучкість, простоту керування, збільшити доступність документа, а також зменшити складність і повторювання в структурній складовій.

Існує чотири способи застосування стилів:

1. Перевизначення стилю в елементі розмітки;
2. Розміщення опису стилю в заголовку документу з використанням тегу style;
3. Розміщення посилання на зовнішньому описі через тег link;
4. Імпорт стилю в документ.

Формат CSS – це текстовий файл, в якому міститься перелік правил CSS і коментарі до них, обов’язкове розширення файлу .css.

Серед недоліків CSS можна зазначити наступні пункти:

- бібліотеки можуть містити зайву інформацію – надлишковий код, який не буде використовуватися в проекті;
- дизайн залежить від CSS-бібліотеки;
- через необхідність додавання великої кількості класів на один

елемент стилю порушується принцип, для якого був створений CSS: поділ описів структури і зовнішнього вигляду[5, с. 36].

Скриптова метамова **SASS** – мова скриптів препроцесора, яка інтерпретується або компілюється в каскадні таблиці стилів (CSS); модуль, що входить в Haml. SASS – це метамова на основі CSS, призначена для збільшення рівня абстракції CSS-коду і спрощення файлів каскадних таблиць стилів.

CSS складається з серії селекторів, які групують правила, які до них потім застосовуються. SASS розширює CSS, надаючи кілька механізмів, доступних зокрема у об’єктно-орієнтовних мовах програмування, але CSS не має до них доступу. Інтерпретатор SASS переводить SassScript у CSS [6, с. 41].

Мова SASS складається з двох синтаксисів:

SASS – характеризується відсутністю фігурних дужок, у ньому за допомогою відступів реалізовані вкладені елементи;

SCSS – використовує фігурні дужки, як і сам CSS.

Мова програмування **JavaScript** – динамічна мультипарадигмальна мова програмування, яка застосовується до HTML документу, і може забезпечити динамічну взаємодію із користувачем на web-сторінках. JavaScript використовується зазвичай як вбудована мова для програмного доступу до об’єктів додатку. Скрипти запускаються в браузері користувача, а не на сервері і зазвичай звертаються до бібліотек третьої сторони для забезпечення розширених функцій без необхідності кожен раз писати код для розробників.

Перелік основних архітектурних особливостей:

- слабка типізація;
- прототипне програмування;
- функції як об’єкти першого класу;
- автоматичний збір мусору;
- динамічна типізація;
- автоматичне керування пам’яттю;

- анонімні функції.

Мова JavaScript дуже популярна і використовується для:

- створення SPA – React, Angular, Vue;
- створення стаціонарних застосунків – Electron, NW.js;
- серверне програмування – Node.js;
- створення мобільних застосунків – React Native, Cordova.

У мови програмування JavaScript наявна стандартна бібліотека: відсутній інтерфейс програмування додатків по роботі з файловою системою, керування потоками вводу та виводу, базових типів для бінарних даних. Також містить стандартні інтерфейси до web-серверів та баз даних; систему керування пакетами, яка відстежує залежність та автоматично встановлює її [7, с. 32].

Всі операції, які можна виконувати в програмі на JavaScript, описують дії над відомими і визначеними об'єктами. Власне, об'єктна-орієнтованість JavaScript на цьому закінчується. Є тільки об'єкти з набором властивостей і набір функцій над об'єктами. Останні називаються методами. Крім методів існують і інші функції, які більше схожі на функції з традиційних мов програмування і дозволяють працювати зі стандартними математичними типами або керувати процесом виконання програми.

В JavaScript є події - аналог програмних переривань. Використовуючи події, автор гіпертекстової сторінки і програми може організувати перегляд динамічних об'єктів, наприклад, функціонал бігучого рядка , або управління багатовіконних інтерфейсів [8, с. 28].

JavaScript можна структурно поділити на три об’єднання, які чітко відрізняються один від одного:

- ядро;
- об’єктна модель браузера;
- об’єктна модель документа.

Якщо розглядати JavaScript не у браузерних середовищах, то можуть не підтримуватися об’єктна модель браузера і об’єктна модель документа [9, с.

55].

Серед недоліків JavaScript, можна зазначити:

- низький рівень безпеки через повсюдний і вільний доступ до вихідного коду популярних скриптів;
- JavaScript іноді інтерпретується по-різному різними браузерами, що призводить неузгодженості з точки зору функціональності та інтерфейсу. Значна частина JavaScript залежить від маніпуляції елементами DOM браузерів. І різні браузери надають різні типи доступу до об'єктів, зокрема Internet Explorer;
- безліч дрібних помилок на кожному етапі роботи. Значна частина з них легко виправляється, але їх наявність дозволяє вважати цю мову менш професійною, ніж інші;
- велика кількість конкурентів. JavaScript – застаріла мова сценаріїв, працююча на машинах. Існують інші технології, які виконують ті ж функції, але легшим і кращим чином;
- завантаження файлу: файл JavaScript завантажується на клієнтській машині, щоб кожен користувач міг прочитати код і повторно використати його;
- повсюдне поширення. Своєрідним недоліком можна назвати той факт, що частина активно використовуваних програм (особливо додатків) перестануть існувати при відсутності мови, оскільки цілком базуються на ньому.

Фреймворк **Angular** – це платформа для розробки web-додатків з відкритим вихідним кодом, який розробники використовують для створення односторінкових Web-додатків. Його також використовують для створення анімованих меню для web-сторінок HTML.

Angular впроваджує архітектуру Model-View-Controller, яка використовується при розробці web-додатків.

Даний тип архітектури складається з:

- Model – структура даних, яка керує інформацією та отримує вхідні дані від контролера;
- View – подання інформації;
- Controller – реагує на вхід і взаємодіє з моделлю.

Особливістю роботи з Angular є те, що він може використовувати в якості мови програмування TypeScript, Dart, JavaScript. Але основною мовою програмування є TypeScript.

Компоненти користуються сервісами, які забезпечують специфічні функції, не пов'язані безпосередньо з представленнями. Постачальники послуг можуть бути введені в компоненти як залежності, що робить код модульним, багаторазовим і ефективним – схема роботи зазначена на рисунку 2.2.

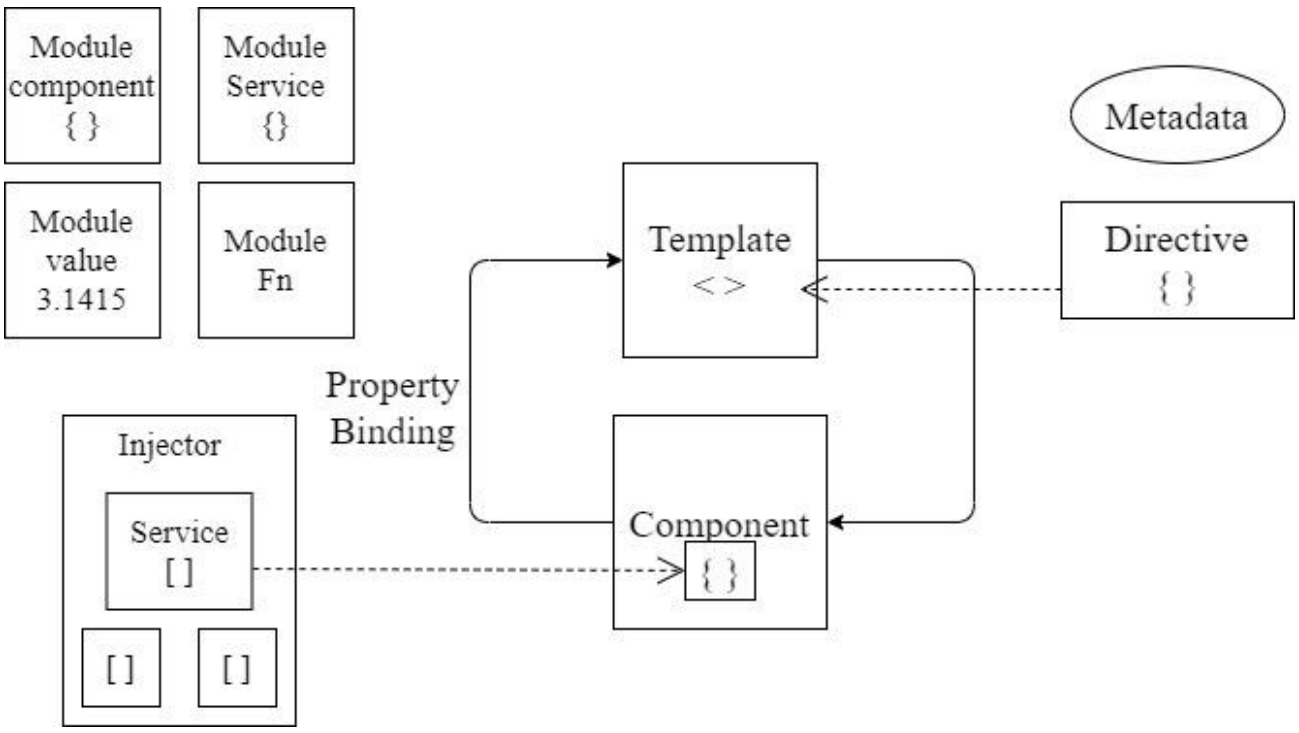


Рисунок 2.2 Схема архітектури Angular додатка

Під час роботи з Angular використовується інтерфейс командного рядка Angular CLI. Він дає можливість використовувати такі команди:

- ng new – створення нового проекту;
- ng generate – генерація нового компоненту або сервісу;

- ng serve – компіляція проекту та завантаження на локальний сервер;
- ng test – компіляція додатку та тестування написаного коду;
- ng build – компіляція проекту в готовий додаток.

При зміні даних моделі в одному місці Angular надає можливість динамічно змінювати дані в іншому.

Основними характеристиками даної платформи є:

- єдина структура об’єкту;
- кросбраузерна підтримка HTTP, WebSockets, Service Workers;
- реактивне програмування з RxJS;
- підтримка Google, Microsoft;
- кросбраузерний Shadow DOM;
- можливість писати типізований код.

Angular надає не тільки інструменти, а й шаблони дизайну для побудови проєкту. Коли Angular додаток створено належним чином, то класи та методи не будуть сплутані і їх не важко модифікувати чи перевірити. Код зручно структурований. Надаються директиви, щоб надати елементам HTML динамічну поведінку. Також є можливість налаштувати маршрутизацію. Angular надає можливість ретельного тестування. Платформа підтримує як модульне, так і наскрізне тестування [10, с. 67].

Недоліки Angular:

- можуть виникнути помилки під час міграції між версіями;
- складна мова програмування в основі;
- Angular використовує DOM, в результаті чого знижується його швидкість та ефективність;
- В Angular досі залишилось фізична роздільність між JavaScript.

Бібліотека RxJS – це бібліотека JavaScript для реактивного програмування, що стосується асинхронних викликів даних, зворотних викликів; дозволяє писати складну логіку декларативно – схематично це

зображено на рисунку 1.3.

Бібліотека RxJS часто використовується на фреймворці Angular. Вона пришвидшує операції вводу-виводу. Також в RxJS оптимізовано використання шаблону Observer (спостережувач) для функціонального програмування. Observer – це шаблон, в якому суб'єкт зберігає список своїх залежностей-спостережувачів і автоматично повідомляє про будь-які зміни їх станів.

Головна ціль реактивного програмування – це реалізація асинхронного додатку на основі подій за допомогою спостережуваних послідовностей. Ця концепція схожа на додаток, з яким більшість знайома – Microsoft Excel.

Мова програмування **TypeScript** – об'єктно-орієнтована мова програмування, яка позиціонує себе як інструмент для розробки web-додатків. Ця мова містить всі елементи JavaScript, але крім цього розширює її можливості – схематично це зображено на рисунку 2.3. TypeScript призначена для широкомасштабної розробки web-додатків, які можуть виконуватися в будь-якому браузері, на будь-якому хості та в будь-якій операційній системі[11, с. 17].

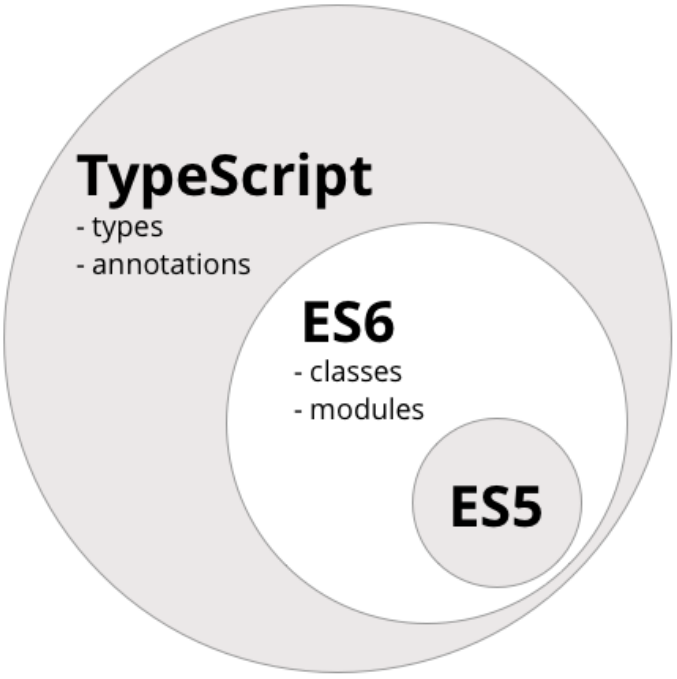


Рисунок 2.3 Розширення JavaScript за допомогою TypeScript

Переваги використання TypeScript:

- транслятор TypeScript надає функцію перевірки помилок. Він буде компілювати код і генерувати похибки компіляції, якщо виявить певні синтаксичні помилки. Це допомагає виділити помилки перед запуском;
- підтримка використання повноцінних класів;
- сильна статична типізація. Тип змінної, оголошений без типу, може бути визначений за її значенням;
- мова підтримує об'єктно-орієнтоване програмування(класи, інтерфейси, успадкування тощо);
- підтримка інших бібліотек JavaScript;
- використовуючи модулі, TypeScript підтримує інкапсуляцію класів, інтерфейсів, функцій та змінних. TypeScript відрізняє зовнішні і внутрішні модулі;
- рефакторинг та повторне використання коду.

Програму, написану на TypeScript, можна виконувати на будь-якому сучасному браузері або використовувати разом з серверною платформою Node.js.

Недоліки TypeScript:

- більш висока вартість написання нового коду. Вимагаючи більше коду, це можу замінити швидкість розробки нових функцій, що може бути не ідеальним компромісом для розробки програмного продукту;
- якщо ви хочете використовувати функції, що знаходяться в нових версіях TS, потрібно рефакторувати код за мережею виходу нових версій;
- складні типи можуть бути складними для розуміння;
- більш докладніший код. Розробник використовує набагато більше символів для написання того ж обсягу коду на відміну від мови програмування JavaScript.

Реактивне програмування - парадигма програмування з асинхронними потоками даних.

Реактивний підхід збільшує рівень абстракції коду і надає можливість концентруватися на взаємозв'язку подій, які визначають бізнес-логіку, щоб постійно підтримувати код з великою кількістю деталей реалізації. Код в реактивному програмуванні буде коротший. Для обробки великої кількості інформації підходить використання реактивного програмування. [12, с. 32].

Характеристика реактивної системи – стійкість, еластичність, орієнтовність на обмін повідомленнями, відзивність.

Модель реакції на події передбачає можливість простого поширення цих або трансформованих подій далі по системі. Яскравим прикладом реалізації реактивного підходу може служити таблиця Excel. У ній існує ланцюжок обчислень, розділена на кілька осередків: при зміні значення однієї з осередків в ланцюжку значення в залежних осередках перераховуються автоматично.

Ідея реактивного програмування покликана спростити створення складних систем. У складній великій системі, як наш програмний продукт, виникає значна кількість різноманітних подій, кожна вимагає певного механізму реакції і обробки. При використанні реактивного програмування події об'єднуються в потоки, а компоненти системи обробляють потік подій. Таким чином, яка не була б складною система і скільки в ній не було б різноманітних подій, вся система будується за принципом генерації потоків і реакції на них. Підхід в моделюванні складної системи дозволяє обробляти кожну подію асинхронно і ізольовано від інших. Дизайн реактивної системи передбачає, що будь-яка функціональність в системі реалізується за допомогою подій і обробників. Завдяки цьому досягається зменшення пов'язаності між компонентами системи, збільшення гнучкості, спрощення масштабування і підвищення стійкості [13, с. 49].

Можна зробити висновок, що у реактивному програмуванні обробка ділиться на велику кількість невеликих завдань, виконання кожної з яких закінчується якоюсь подією. На подію реагує обробник, який виконує своє завдання і знову генерує задачу. Генерація події і реакція на неї відбувається асинхронно.

Термін Observable використовується у реактивному програмуванні. Це ліниві push-колекції декількох значень. Для того, щоб створити свій потік, достатньо викликати конструктор даного класу і передати йому в якості аргументу функцію підписки. Спостерігач підписується на Observable і потім реагує на будь-який предмет або послідовність предметів, що продукує Observable – схематично цей процес зображено на рисунку 2.4.

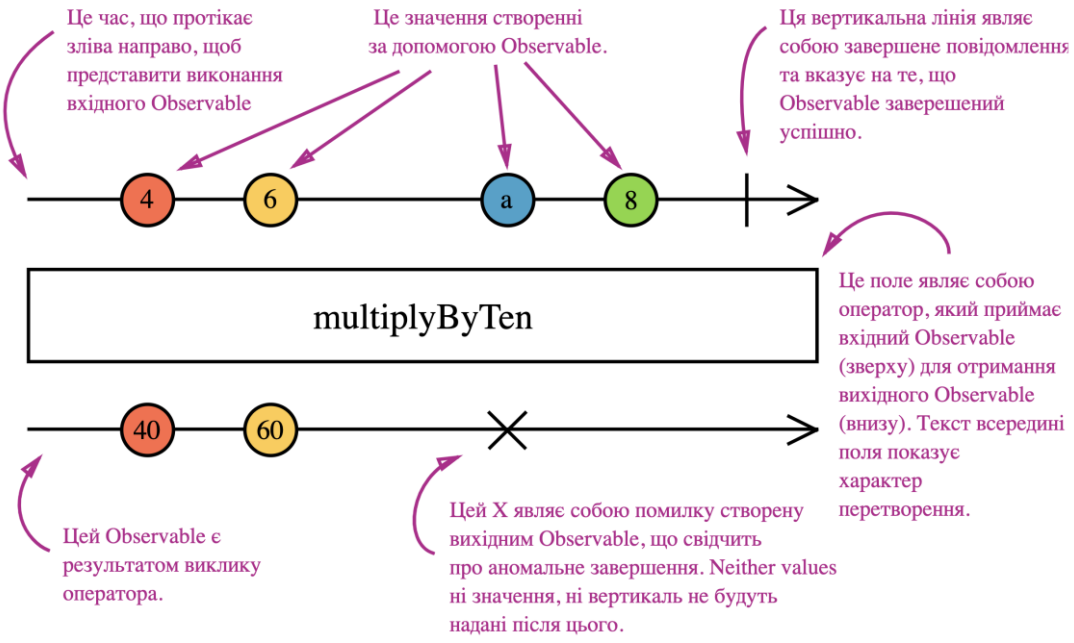


Рисунок 2.4 Представлення Observable у виді marble діаграми

Через виклик конструктора класу Observable, створюється новий потік. В якості аргументу в конструктор передається функція підписки. Функція підписки – це звичайна функція, яка в якості параметру приймає спостерігача (observer). Сам спостерігач являє собою об’єкт, у якого є три методи:

- next – видає нове значення в потік;
- error – видає в потік помилку, після чого він завершується;
- complete – завершує потік.

Таким чином створений потік, що приймає два значення і завершується.

Коли в проекті існує маса завдань, які не залежать одне від одного, то перевага такого підходу очевидна – є можливість розпочати їх одночасно, а не чекати, коли кожне закінчиться, щоб розпочати наступне.

Можна зробити висновок, що пакет завдань займає стільки часу, скільки найдовше завдання в пакеті.

2.2.2 Серверна частина програмного продукту

Серверне програмне забезпечення виконує серверні функції по запиту користувача, надає йому доступ до конкретних послуг або ресурсам. На серверній частині зазвичай описано більшу частину бізнес-логіки.

Найбільша користь програмування серверної частини в тому, що воно дозволяє формувати контент web-додатку під конкретного користувача. Це також може спростити використання сайтів за рахунок збереження інформації.

Від серверної частини залежить швидкість обробки запитів користувача, стійкість роботи web-додатку, і функціональні можливості web-додатку в цілому.

З огляду на швидкий розвиток програмного забезпечення для web-додатків, простежується тенденція розвитку нових алгоритмів для розробки, але на першому місці завжди залишається серверна частина, розробці якої приділяється більше всього уваги.

У створеній системі використовуються **документо-орієнтовані бази даних**, що призначені для збереження і керування документо-орієнтованої або напівструктуризованої інформації. На відміну під реляційних баз даних з їх поняттям відношень, ці системи побудовані навколо абстрактного поняття документа. Такий документ містить довільне число властивостей. Документи в документо-орієнтованій базі схожі на записи в реляційній базі даних, але до них не такі суворі вимоги. Вони не повинні містити однакову структуру і кількість атрибутів [14, с. 7].

Документо-орієнтований підхід зручно використовувати у наступних випадках:

- коли основними даними є документи;
- якщо важлива швидкість роботи програми і важко сортувати дані

по таблицях і сутностям;

- якщо необхідні зв’язки один-до-одного чи багато-до-багатьох і не потрібно їх видаляти;
- якщо необхідні швидкі онлайн-транзакції.

Переваги використання документо-орієнтованого підходу:

- в порівнянні з реляційними базами даних швидша продуктивність при індексуванні великих обсягів даних і великої кількості запитів на читання;
- легше масштабуються в порівнянні з SQL рішеннями;
- не потрібно виконувати ніяких операцій оновлення для додавання нових полів;
- немає проблем зі зберіганням неструктурованих даних;
- зручний інтерфейс для спілкування з базою даних;
- збереження всієї інформації про об’єкт в єдиному місці: менше операцій виду «join».

Недоліки використання документо-орієнтованого підходу:

- відсутність логіки і контролю цілісності в більшості реалізацій: необхідно її реалізувати в логіці програми. Спеціалізована логіка може виявитися ефективніше загальних алгоритмів реляційних баз даних;
- для обробки даних необхідне використання додаткової мови програмування.

Основним компонентом серверної частини програмної системи є база даних Firebase.

Платформа **Firebase** – це одне з BaaS-рішень, хмарна база даних, яка дозволяє легко зберігати інформацію і отримувати її. Вона має зручні інструменти та методи взаємодії з нею.

Firebase керує даними в базі даних у режимі реального часу. Отже, він миттєво обмінюється даними до бази даних та з неї.

Firebase дозволяє синхронізувати дані в режимі реального часу на всіх пристроях – iOS, Android та Web – без оновлення екрана і має просту панель

управління. На рисунку 2.5 зображено схему роботи Firebase в порівнянні з іншими базами даних.

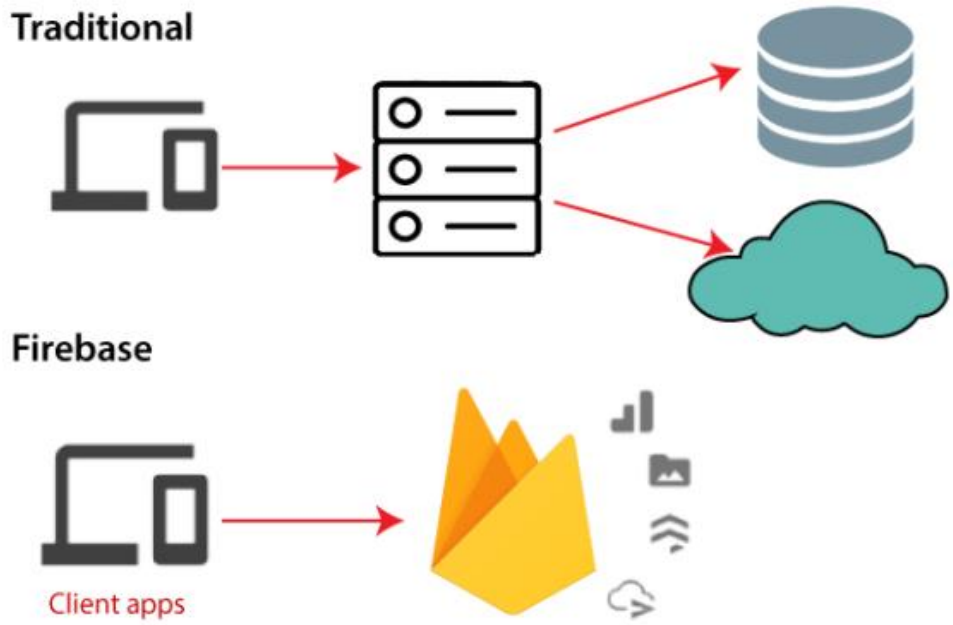


Рисунок 2.5 Схема роботи Firebase в порівнянні з іншими базами даних

Переваги використання Firebase:

- база даних в режимі реального часу;
- відсутність серверів;
- висока швидкість роботи програми;
- мінімальні налаштування;
- має величезний потенціал розміру сховища;
- легкий доступ до даних, файлів, автентифікації тощо;
- має сховище JSON, що означає відсутність бар'єру між данами та об'єктами;
- програма може працювати з усіма операційними системами.

Firebase повністю корегована база даних, яка не потребує особливих зусиль і задовольняє будь-які потреби, без вікон технічного обслуговування та додаткових простоїв. Потужний механізм запитів Firestore дозволяє запускати складні транзакції ACID щодо даних документів. Це надає велику гнучкість у

структурі даних.

В якості бази даних в системі із Firebase використовується Cloud Firestore. Це NoSQL-база даних, яка дає швидкий доступ до документів і колекцій, що зберігаються в БД, синхронізація даних у режимі реального часу, автоматичну багаторегіональну реплікацію даних із сильною узгодженістю тощо [17]

Security Firestore легко інтегрується з платформою автентифікації Firebase та платформою ідентифікації, щоб забезпечити засоби управління доступом на основі ідентифікації та забезпечити перевірку даних за допомогою мови конфігурації, що схематично зображено на рисунку 2.6.

Серед недоліків Firebase, можна виділити наступні:

- формат збереження даних повністю відрізняється від формату SQL(Firebase використовує JSON), тому не легко виконати міграцію даних;
- не має відкритого вихідного коду;
- залежність від постачальника;
- Firebase не існує в багатьох країнах;
- працює тільки в Google Cloud;
- виділені сервери і корпоративна технічна підтримка відсутня;
- це не дешева платформа зі складно прогнозованою ціною;
- повільні запити;
- доступні бази даних тільки NoSQL;
- розробник не розміщує дані, це робить Firebase;
- якщо додаток не запускає єдину централізовану базу даних, оновлену великою кількістю користувачів, то це є зайвим.

Отже, використання платформи Firebase під час розробки і її окремих функцій надають можливість зосередитися виключно на те, як програмна система буде виглядати і наскільки зручним буде його використання.

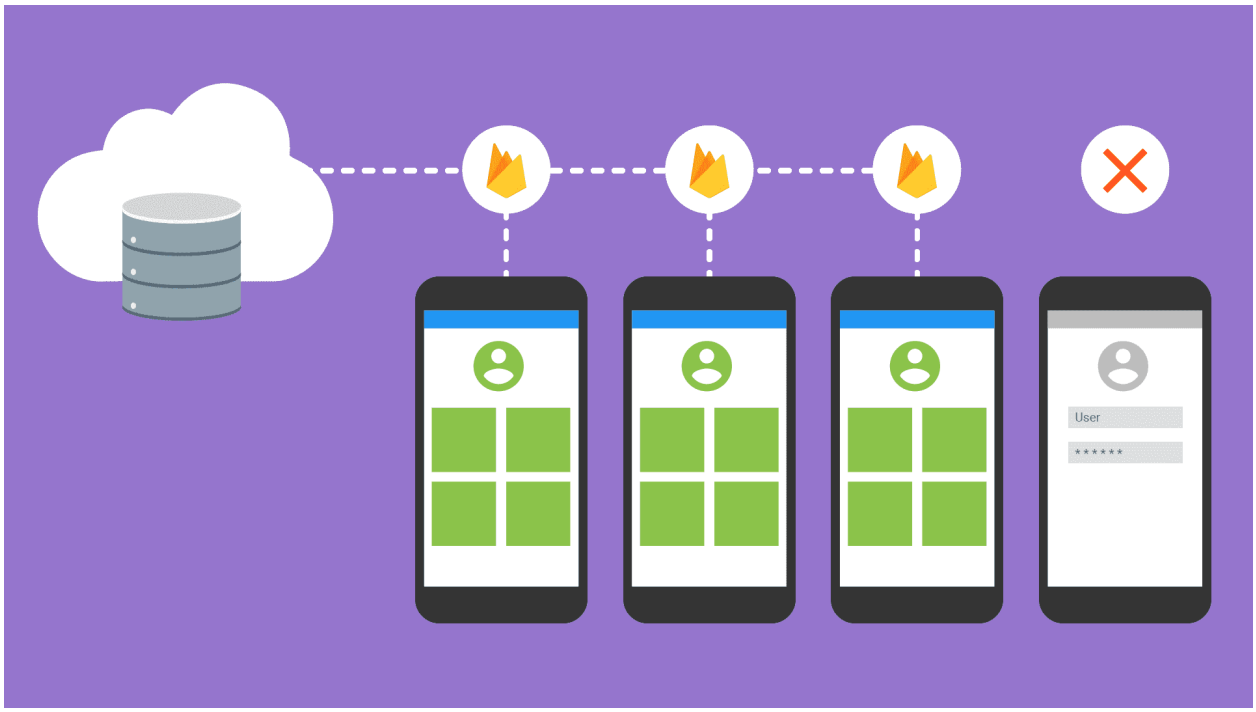


Рисунок 2.6 Схема роботи Firebase зі сторони безпеки доступу до даних через автентифікацію користувача

2.3 Варіанти впровадження web-системи

Однією з задач дослідження є впровадження створеної web-системи третього рівня вищої освіти для подальшого використання відділом аспірантури і докторантури. Для впровадження системи, необхідно придбати або замовити хостінг, обрати серверну ОС та налаштувати її роботу із створеним web-додатком. Оскільки замовлення послуг хостера виходить за межі теоретичного дослідження, будуть розглянуті лише можливі варіанти вибору серверної ОС.

Серед найпопулярніших серверних операційних систем, можна виділити CentOS, Ubuntu Server та Windows Server. У кожного з варіантів є свої переваги та особливості роботи.

Комерційна лінійка продуктів Windows Server, із останньою версією цього продукту – Windows Server 2019, надає своїм клієнтам 2 типи платної підписки – напіврічну підписку або підписку довгострокового обслуговування. Серед явних переваг цієї ОС можна виділити активну

підтримку користувачів та постійне оновлення налаштувань безпеки. Ця ОС легко інтегрується із іншими продуктами компанії Microsoft, такими як сервера баз даних MSSQL та застосунками, створеними на ASP.NET, проте не надає явних переваг при інтеграцією із Firebase. Із недоліків вибору цієї ОС можна зазначити додаткові витрати на ліцензію [18]. Із інших недоліків можна виділити постійну роботу цієї ОС в інтерактивному режимі користувача, що займає додаткові системні ресурси на виділеному web-сервері. Для інтеграції системи із Firebase, встановлюється спеціальний SDK з офіційного сайту Firebase, а також Node Package Manager і його залежності [19].

ОС Ubuntu Server – серверна операційна система, основана на дистрибутиві Debian Linux. Є абсолютно безкоштовною та підтримується активною спільнотою користувачів. Додаткове програмне забезпечення встановлюється за допомогою менеджеру пакетів, який може використовуватися у режимі командної строки [20, с 10]. Менеджер пакетів за замовчуванням використовує вбудований репозиторій, проте користувачі системи можуть додати власні за своїм бажанням. Оскільки робота може вестися в режимі CLI, це зберігає ресурси виділеного серверу. Для інтеграції із різними інструментами Firebase, можна використовувати компіляцію з офіційного репозиторію на GitHub. Для встановлення самого Firebase достатньо запустити на виконання автоматичний curl скрипт. [19]

Іншою альтернативою може виступити серверна ОС CentOS – це дистрибутив Linux на основі комерційного дистрибутиву Red Hat Enterprise Linux компанії Red Hat. Навідміну від RHEL має некомерційну ліцензію. Система має ті самі особливості роботи, що і Ubuntu Server. CentOS за своїм дизайном був спрямований на оптимізацію використання ресурсів, тому навантажує сервер навіть менше за Ubuntu. Із недоліків цієї системи можна виділити досить повільну розробку нових версій та патчів, а також обмежену підтримку користувачів, здебільшого за рахунок інших користувачів системі у чатах та форумі офіційного Web-сайту [21].

3 СУПРОВОДЖЕННЯ ДОДАТКУ «СИСТЕМА ПІДГОТОВКИ ТА АТЕСТАЦІЇ НАУКОВО-ПЕДАГОГІЧНИХ КАДРІВ ВИЩОЇ КВАЛІФІКАЦІЇ»

Розроблений програмний продукт представляє собою web-додаток і серверну частину. Архітектуру програмного продукту зображено на рисунку 3.1.

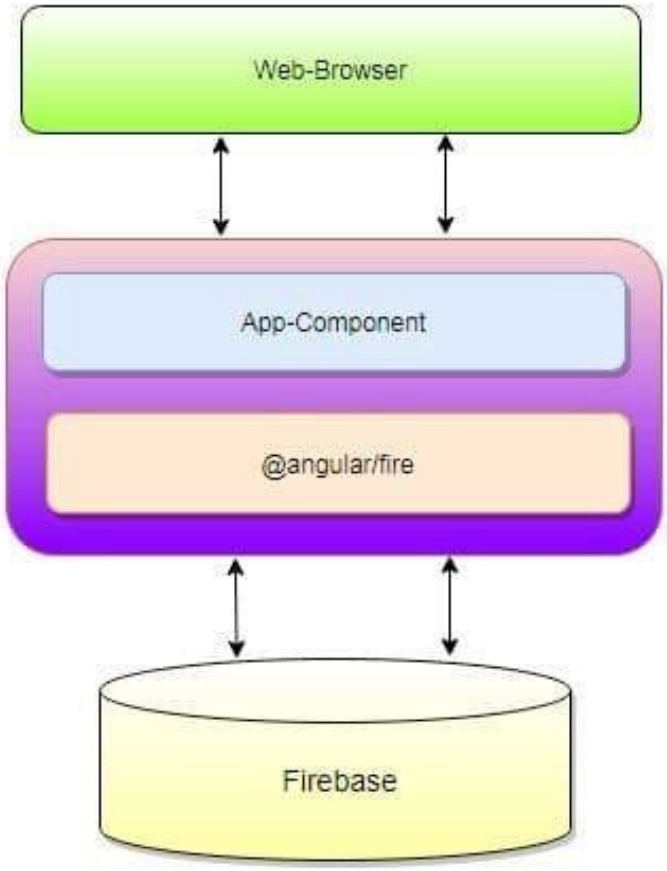


Рисунок 3.1 Архітектура програмної системи

3.1 Концептуальний опис взаємодії користувача з модулем «Вступ» та аккаунтами Адміністратора

Список основного функціоналу необхідного для розробки відповідно до потреб відділу аспірантури:

1. Відправлення заяв для вступу до третього рівня вищої освіти.

2. Перегляд балів за іспити.

3. Можливість редагування вхідних даних користувача.

4. Функція формування груп.

5. Функція виставлення балів.

6. Функція виставлення додаткових балів, які можуть змінити результат вступної кампанії.

7. Фільтрування кандидатів за визначеними критеріями.

8. Перегляд статистики.

Створити головну сторінку для взаємодії з користувачем, з якої є можливість перейти на сторінку для реєстрації або сторінку для перегляду рейтингового списку.

Вступнику треба заповнити та обрати такі дані, щоб подати реєстраційну форму:

- прізвище;
- ім’я;
- по-батькові;
- громадянство;
- стать;
- дата народження;
- спеціальність – випадаючий список;
- освітня програма – випадаючий список;
- кафедра – випадаючий список;
- факультет – заповнюється на основі обраних даних про спеціальність, освітню програму та кафедру;
- форма навчання – випадаючий список;
- форма оплати – випадаючий список;
- іноземна мова – випадаючий список;
- ВНЗ, що закінчив вступник;
- рік закінчення ВНЗ;

- освітній ступінь – випадаючий список;
- серія, номер диплому;
- наявність відзнаки;
- адрес електронної пошти;
- номер телефону;
- потреба в гуртожитку;
- середній бал диплому;
- кількість наукових публікацій;
- прізвище наукового керівника;
- особливі відзнаки (перемоги на олімпіадах, патенти тощо);
- додатковий коментар.

Дані повинні зберігатися в базі даних (рисунок 3.2) після їх подання вступником. Передбачити можливість адміністратору обробити вхідні дані вступника після цього.

Передбачити наступні розділи для інтерфейсу адміністратора:

- моніторинг надісланих кандидатами заяв;
- пошук обраної заяви за критеріями;
- перегляд деталізованої інформації про кожну заяву;
- підтвердження реєстрації;
- відхилення реєстрації;
- завантаження таблиць з даними про аспірантів у форматі .xlsx;
- створення нових кафедр, факультетів, освітніх програм, іноземних мов;
- створення нових груп;
- видалення з бази абітурієнта/аспіранта;
- моніторинг статистики приймальної комісії;
- перегляд переліку зарахованих вступників;
- встановлення балів за вступні іспити;
- завантаження таблиць у форматі .xlsm з інформацією про

вступників.

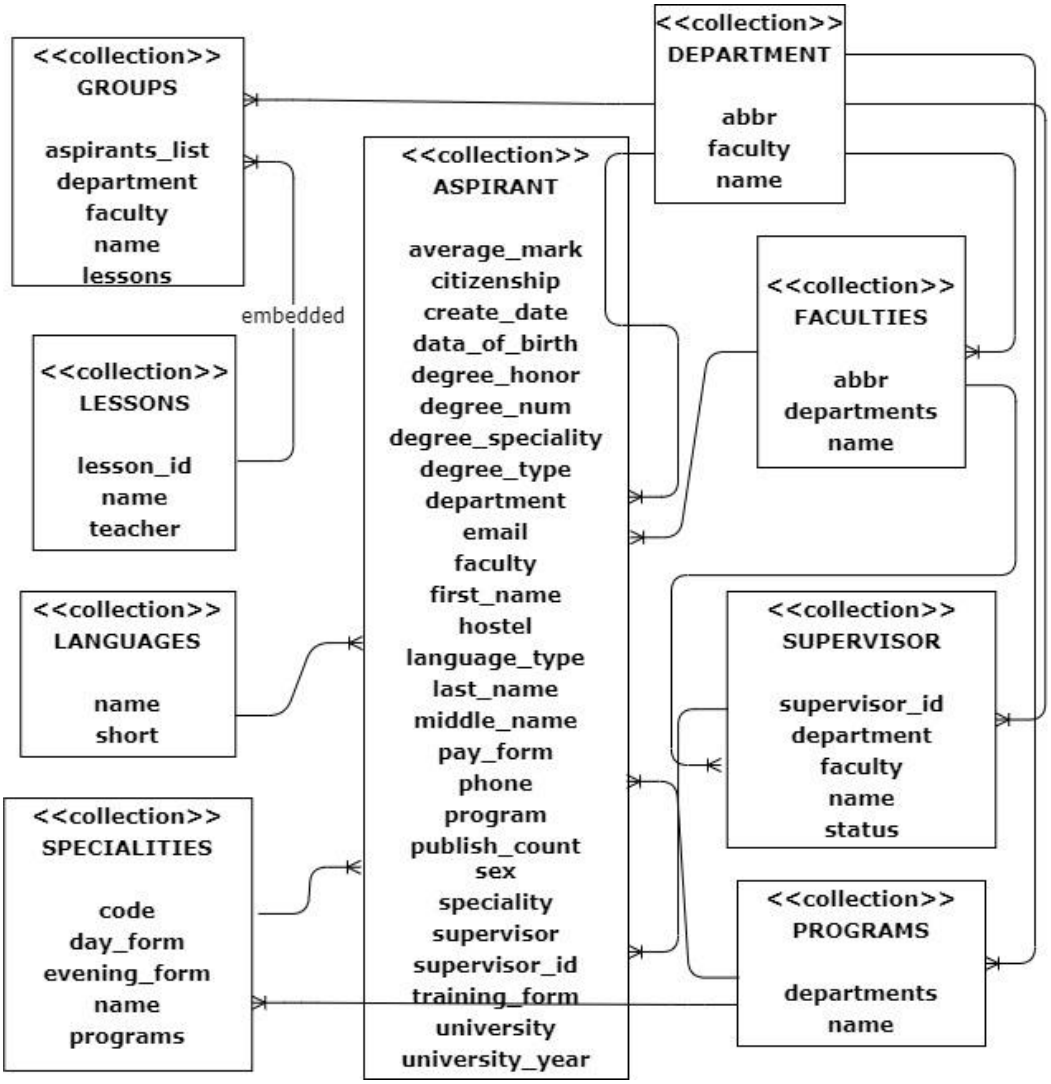


Рисунок 3.2 Схема бази даних програмної системи

На рисунку 3.3 Зображено діаграму прецедентів системи.

На діаграмі послідовності програмного продукту для ролі вступника (рисунок 3.4) продемонстровано його взаємодію з фрагментами програмного продукту. Вступник надається можливість подати реєстраційну форму для вступу. Вхідні дані автоматично зберігаються в базі даних.

На діаграмі послідовності програмного продукту (рисунок 3.5) схематично описана взаємодія адміністратора з фрагментами системи, що супроводжують вступ до аспірантури. Адміністратор повинен ввести дані пари логін-пароль, щоб авторизуватися в програмній системі. Після успішної

авторизації адміністратор має можливість ознайомитися з переліком заяв вступників, підтверджувати або відхиляти їх реєстрацію, побачити перелік підтверджених заяв, заповнити бали за іспити та змінити потрібну інформацію в даних користувача. Відділ адміністрації може прийняти абітурієнта до аспірантури, визначити йому номер групи в якій буде проходити навчання.

Рисунок 3.3 Діаграма прецедентів системи

Продемонстровано взаємодію викладача з фрагментами програмного продукту на діаграмі послідовності (рисунок 3.6). Викладач має змогу переглянути особисті дані, виставити оцінки та заповнити відвідування аспіранта.

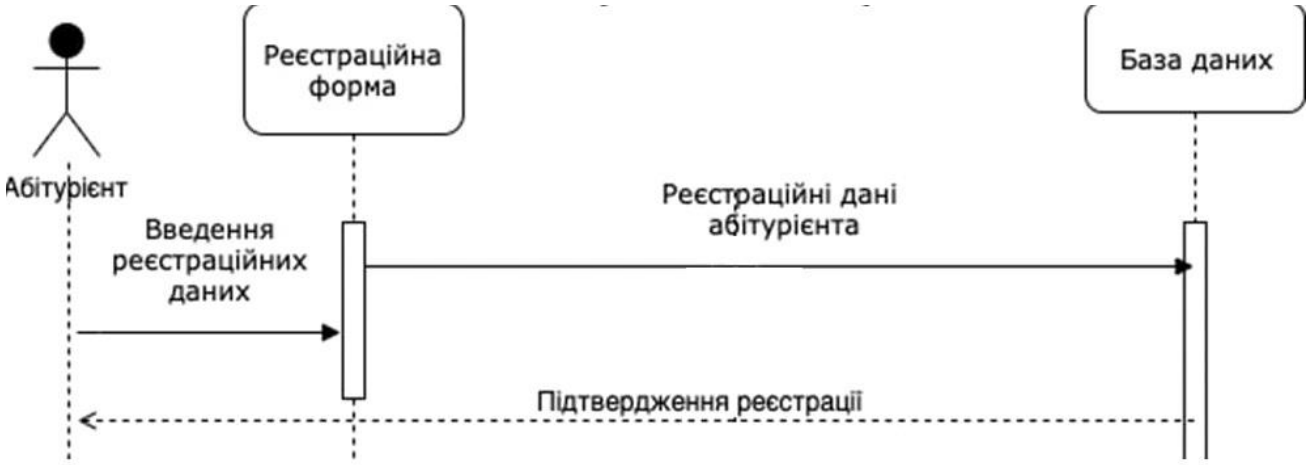


Рисунок 3.4 Діаграма послідовності системи для ролі Абітурієнт

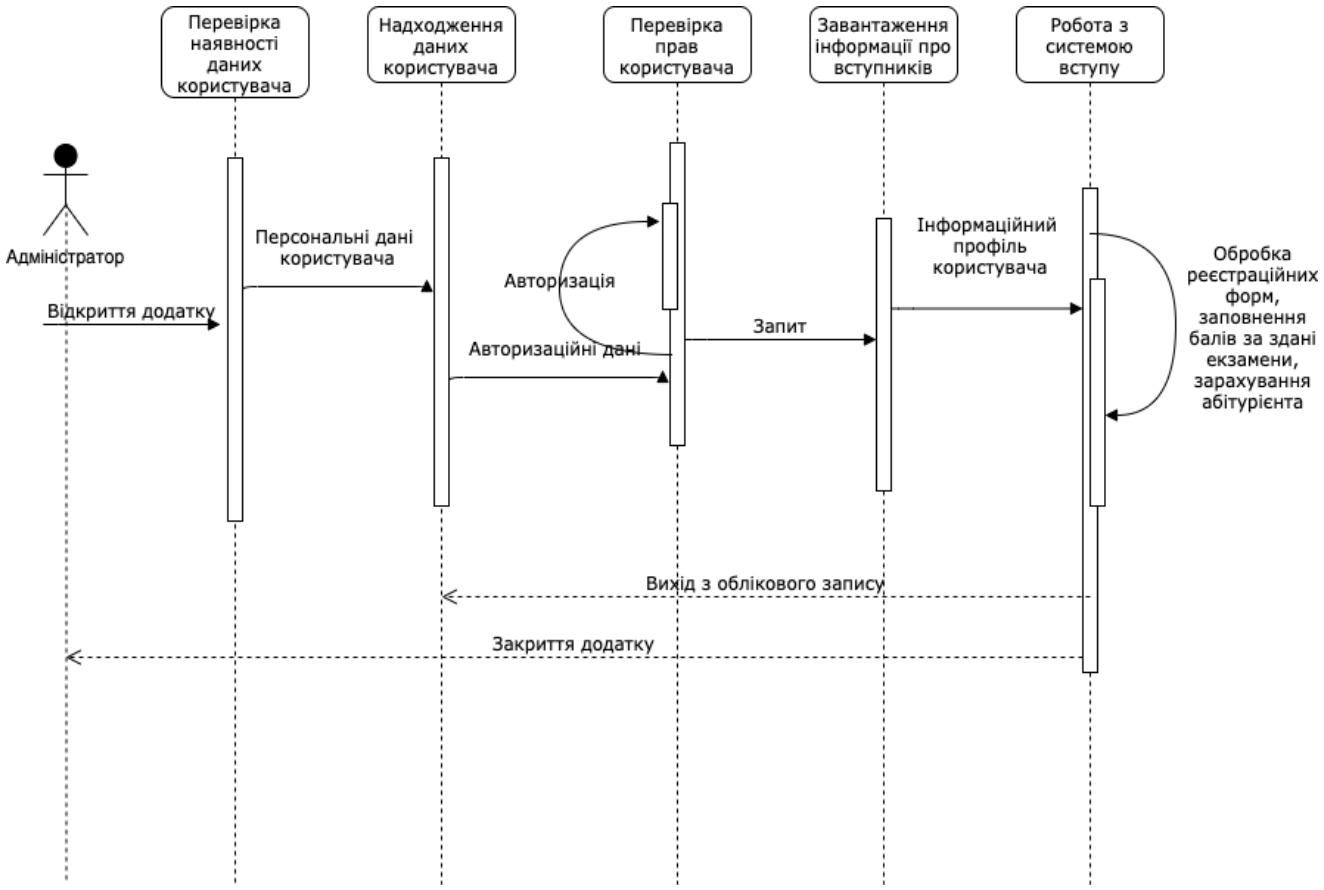


Рисунок 3.5 Діаграма послідовності системи для ролі адміністратора

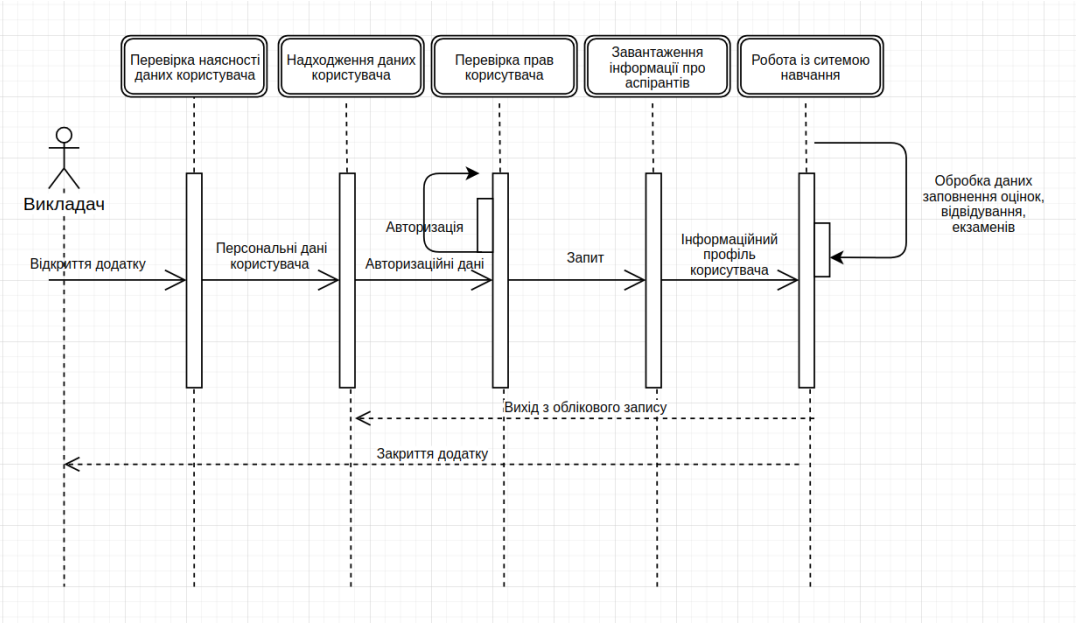


Рисунок. 3.6 Діаграма послідовності для ролі викладача

Web-додаток потрібно змінити таким чином, щоб були виконані наступні пункти:

1. Після заповнення та вибору всіх даних для подання заяви до вступу до третього рівня вищої освіти абітурієнту надати можливість отримати електронну копію поданої реєстраційної картки для перегляду та перевірки інформації (рисунок 3.7).

Рисунок 3.7 Електронна копія поданої реєстраційної картки для перегляду та перевірки інформації

2. Передбачити блокування подачі двох або більше заяв однією особою на одну спеціальність. Але є можливість подачі декількох заяв на різні спеціальності.

3. Після громадянства додати поле для вибору громадянства – Громадянин України або Посвідка на постійне проживання (рисунок 3.8).

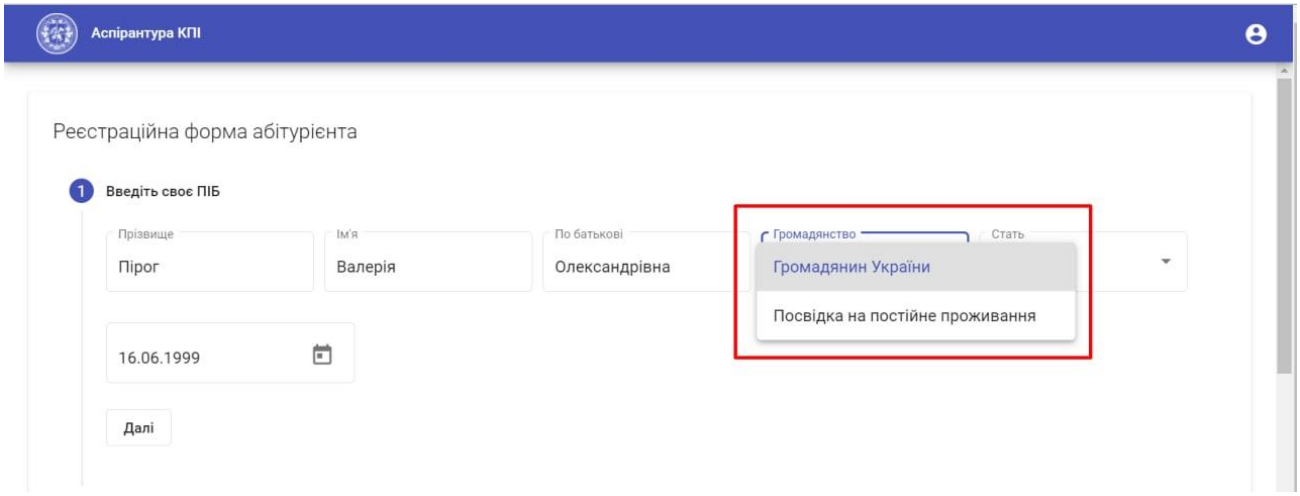


Рисунок 3.8 Поле для вибору громадянства

4. Максимальний вибір дати народження 20 років тому від дати реєстрації (рисунок 3.9).

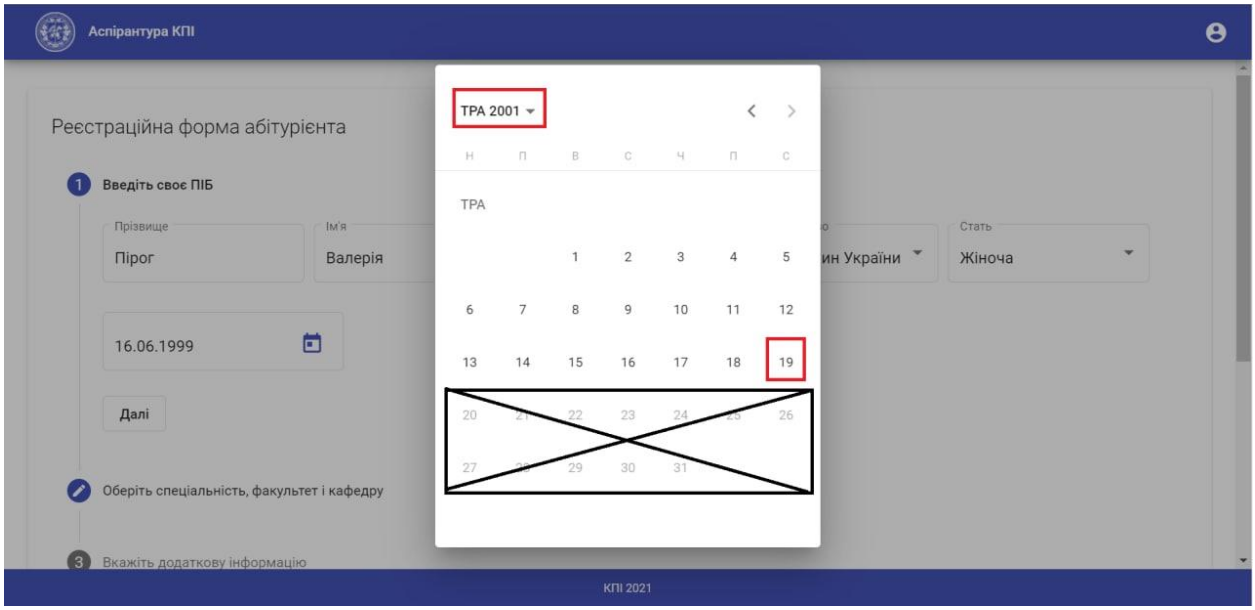


Рисунок 3.9 Вибір дати народження

5. Додати поле «Освітня програма» на другому етапі реєстрації (рисунок 3.10). Для її вибору відображається випадаючий список. Для однієї спеціальності може бути одна або більше освітніх програм, а за однією програмою може здійснюватися підготовка на різних кафедрах. До того ж, факультет має заповнюватися автоматично на основі вибору кафедри, освітньої програми та спеціальності.

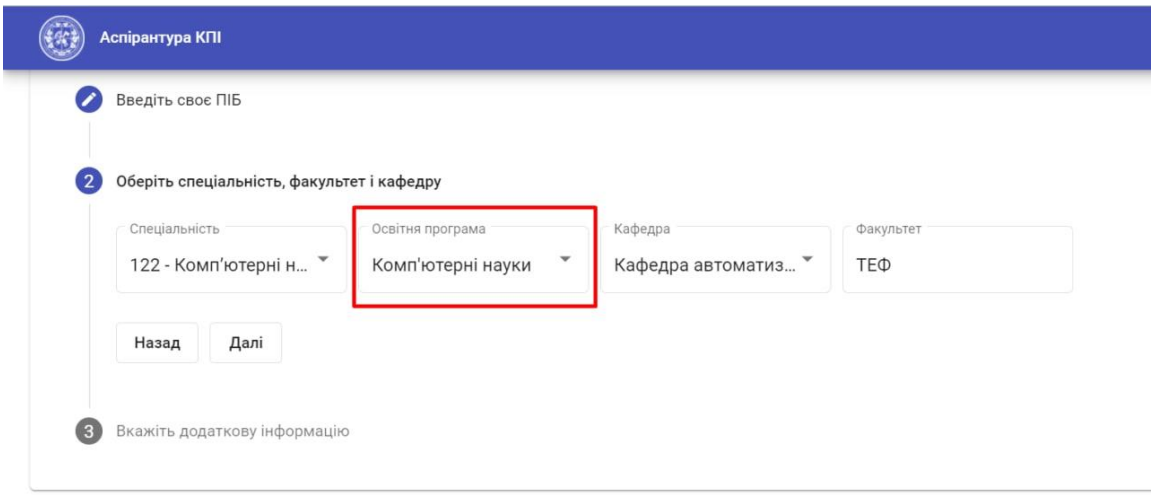


Рисунок 3.10 Поле для вибору освітньої програми

6. Загальний перелік освітніх програм та їх відповідність спеціальностям повинен визначатися адміністратором (рисунок 3.11).

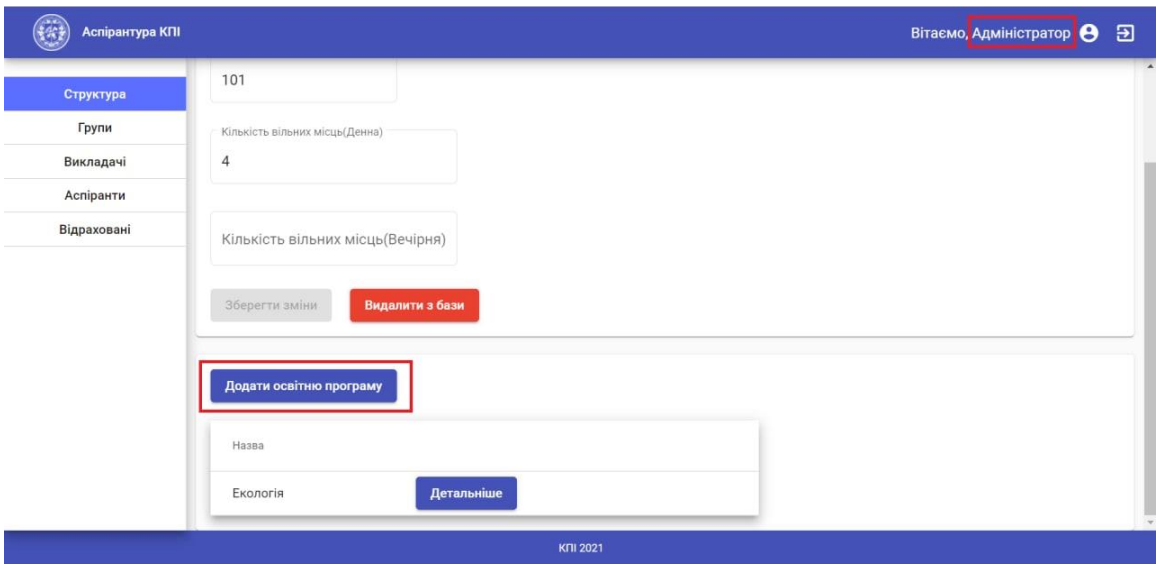


Рисунок 3.11 Вікно для додавання освітньої програми

7. Надати можливість адміністратору додавати, виключати зі списку, вносити зміни з переліки та назви факультетів, кафедр, спеціальностей та їх зв’язки (рисунок 3.12).

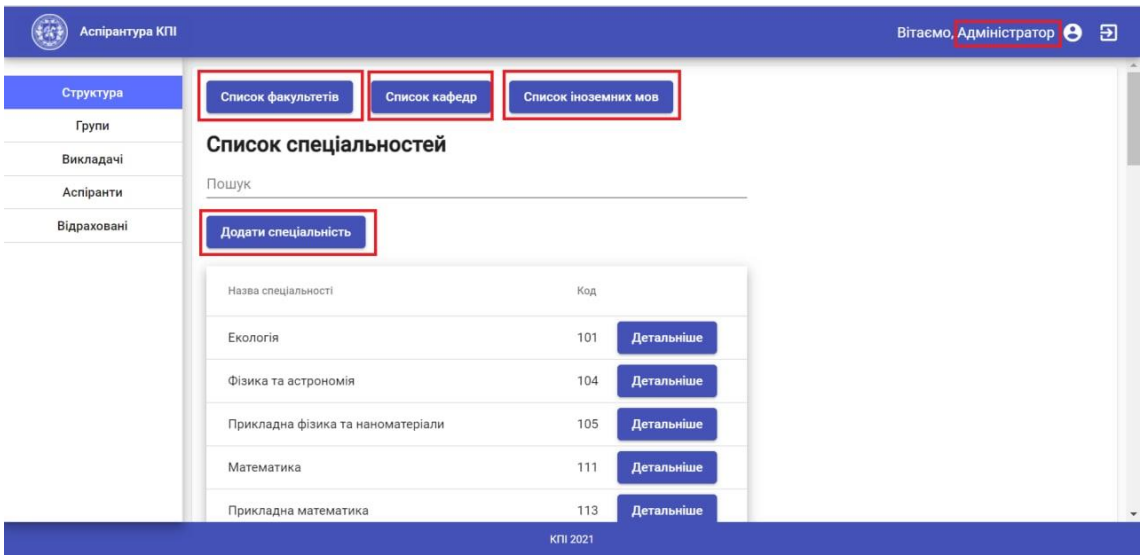


Рисунок 3.12 Вікно для перегляду/додавання нових факультетів, кафедр, іноземних мов, спеціальностей

8. Додати текстові поля «серія, номер диплому», «Спеціальність за дипломом про вищу освіту». Замінити абревіатуру «ВНЗ» на «ЗВО» (рисунок 3.13).

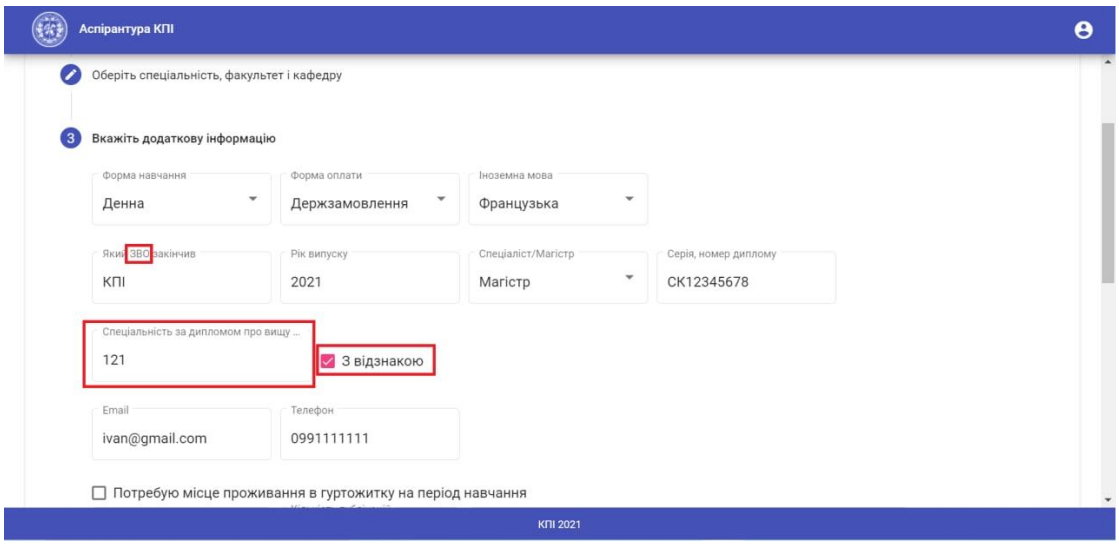


Рисунок 3.13 Поля «серія, номер диплому», «Спеціальність за дипломом про вищу освіту»

9. Видалити можливість вибору заочної форми за державним замовленням (рисунок 3.14).

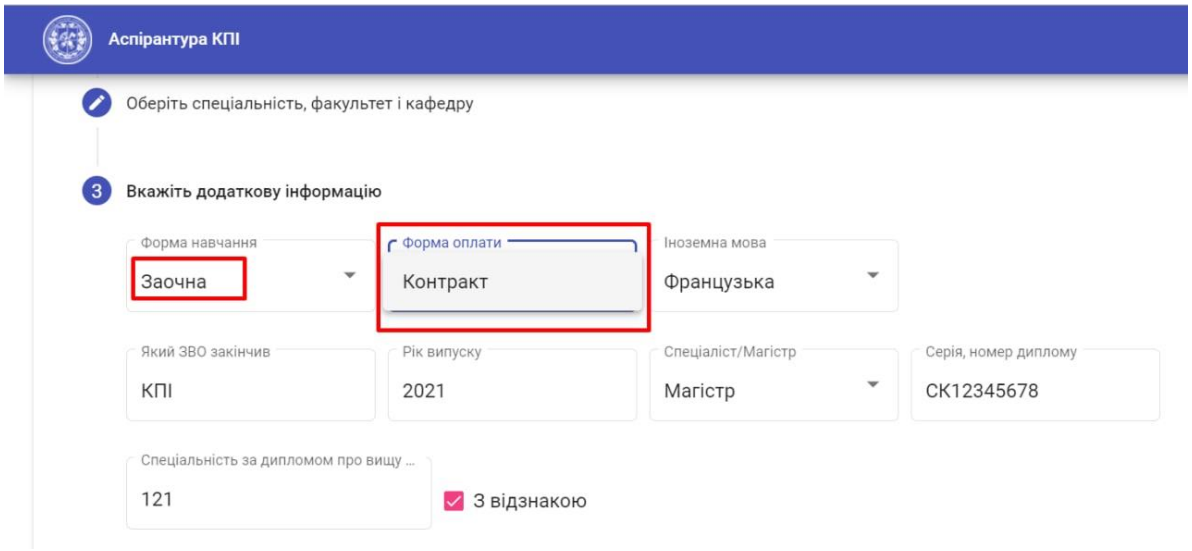


Рисунок 3.14 Приклад вибору форми оплати з заочною формою навчання

10. Коли адміністратор натискає на кнопку «Відмовити в реєстрації» передбачити запит-підтвердження на видалення картки або відмовитися від видалення (рисунок 3.15). Код розробленого фрагменту знаходиться в Додатку Б.

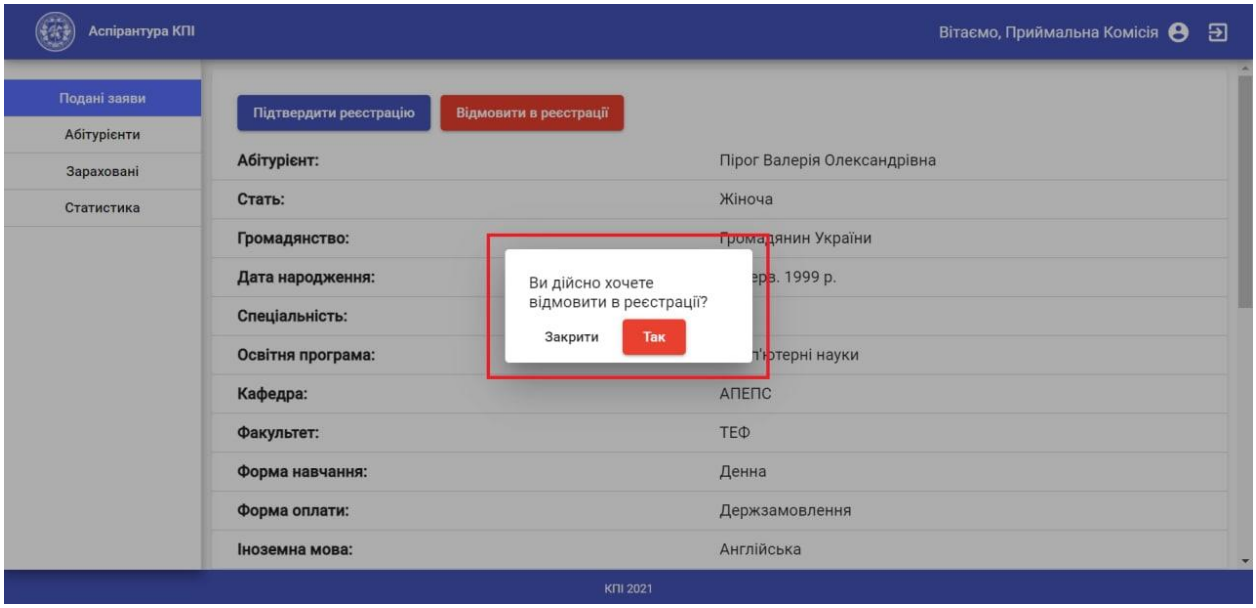
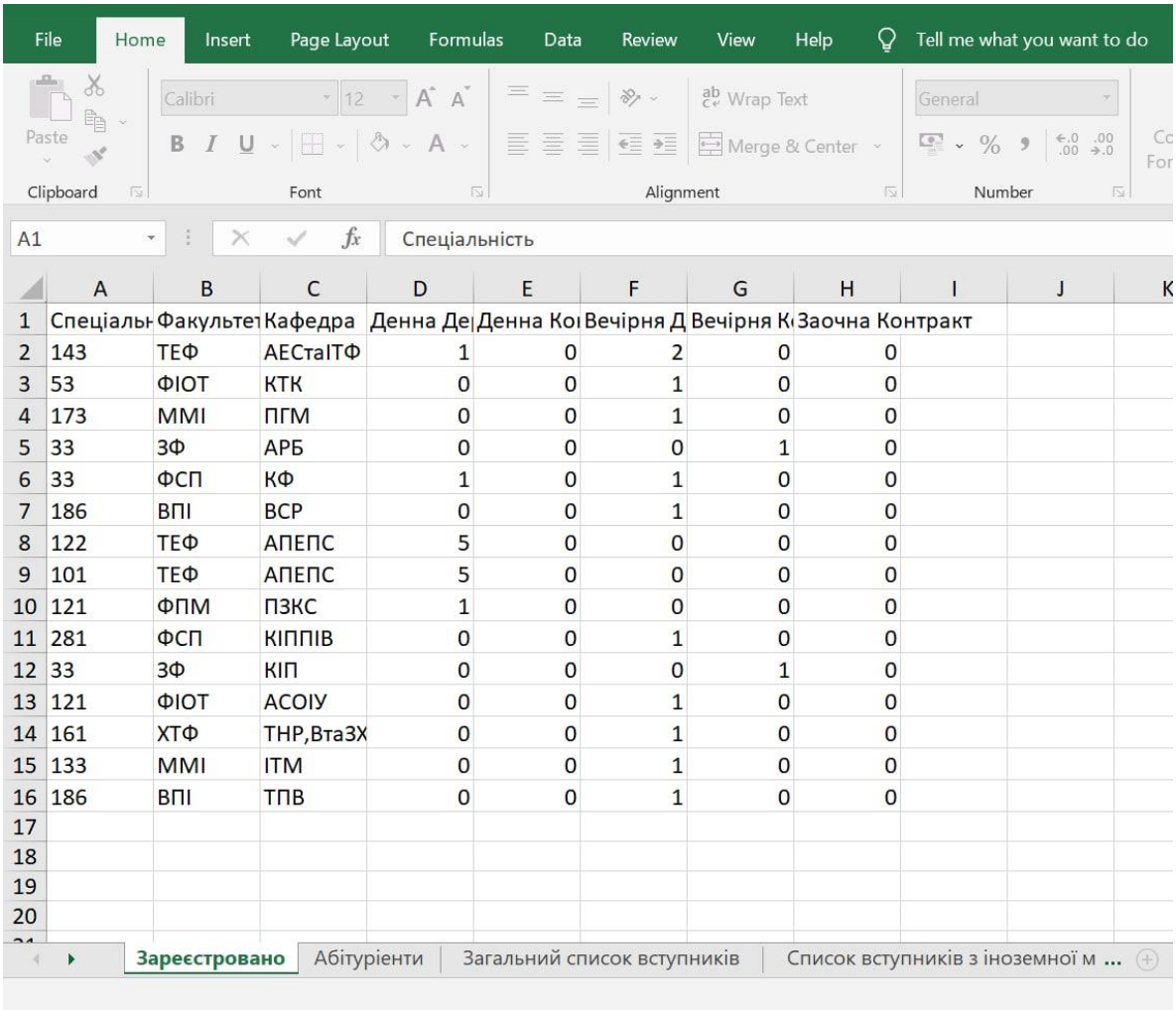


Рисунок 3.15 Запит-підтвердження на видалення реєстраційної картки

11. З поданих заяв формувати різні статистичні форми-таблиці у форматі .xmlx на сторінці адміністратора (рисунок 3.16):

- повні дані реєстраційних карток;
- список абітурієнтів за іноземними мовами;
- список вступників, які потребують гуртожиток;
- зведені дані всіх абітурієнтів.



	A	B	C	D	E	F	G	H	I	J	K
1	Спеціальн	Факультет	Кафедра	Денна Де	Денна Ко	Вечірня Д	Вечірня К	Заочна	Контракт		
2	143	ТЕФ	АЕСтаІТФ	1	0	2	0	0			
3	53	ФІОТ	КТК	0	0	1	0	0			
4	173	ММІ	ПГМ	0	0	1	0	0			
5	33	ЗФ	АРБ	0	0	0	1	0			
6	33	ФСП	КФ	1	0	1	0	0			
7	186	ВПІ	ВСР	0	0	1	0	0			
8	122	ТЕФ	АПЕПС	5	0	0	0	0			
9	101	ТЕФ	АПЕПС	5	0	0	0	0			
10	121	ФПМ	ПЗКС	1	0	0	0	0			
11	281	ФСП	КІППІВ	0	0	1	0	0			
12	33	ЗФ	КІП	0	0	0	1	0			
13	121	ФІОТ	АСОІУ	0	0	1	0	0			
14	161	ХТФ	ТНР,ВтаЗХ	0	0	1	0	0			
15	133	ММІ	ІТМ	0	0	1	0	0			
16	186	ВПІ	ТПВ	0	0	1	0	0			
17											
18											
19											
20											

Рисунок 3.16 Приклад завантаженої таблиці з даними про аспірантів

12. В анкеті вступника додати стовпчик дати народження. Пошук абітурієнта здійснювати за нею теж (рисунок 3.17).

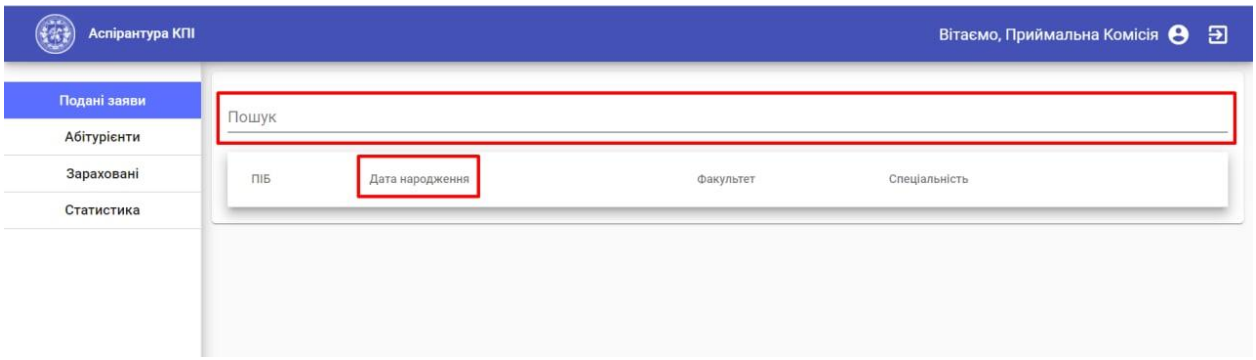


Рисунок 3.17 Поле пошуку поданих заяв та стовпчик з датою народження

13. З головної сторінки web-додатку видалити кнопку для перегляду рейтингу.

14. Група не повинна обиратися аспірантом. Її повинен створити, якщо немає, і обрати в системі адміністратор для кожного аспіранта окремо (рисунок 3.18).

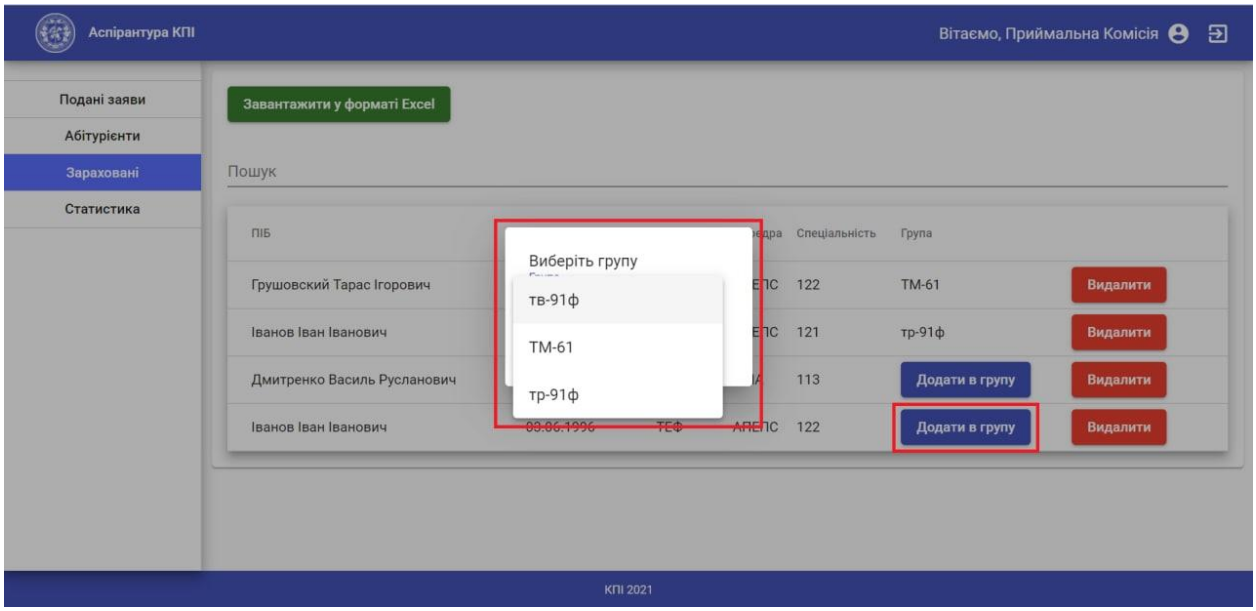


Рисунок 3.18 Сектор для визначення групи аспіранта в акаунті адміністратора

Проведено тестування всіх вищезазначених вимог по зміні web-додатку. Ознайомитися з процесом та результатами можна в Додатку Г.

3.2 Опис програмної реалізації web-додатку

Web-додаток розроблений за допомогою мови TypeScript та фреймворку Angular 8, що дає можливість створити швидко-зростаючий додаток. Він має модульну структуру (рисунок 3.19) і включає в себе з декілька модулів:

а) модуль особистого кабінету адміністратора;

б) модуль реєстрації вступної заяви абітурієнта;

в) модуль ядра додатку;

г) спільний модуль.

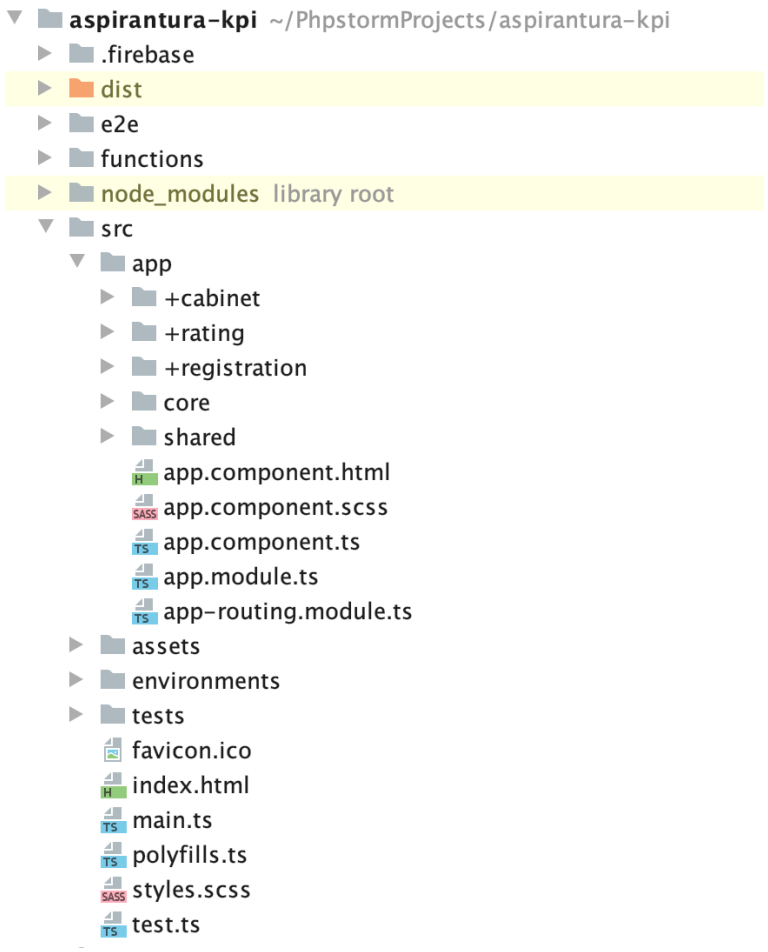


Рисунок 3.19 Структура проекту web-додатку

Сервіс модулю особистого кабінету адміністратора відділу аспірантури використовує зовнішній сервіс Firebase та HttpClient для зв’язку з зовнішніми джерелами даних за допомогою шаблону проектування впровадження

залежності в конструктор (рисунок 3.20).

```
@Injectable()
export class CabinetService {
  constructor(
    private _afs: AngularFirestore,
    private _authService: AuthService,
    private _httpClient: HttpClient,
  ) {}
```

Рисунок 3.20 Впровадження залежності в сервіс особистого кабінету адміністратора

Розроблені методи, що завантажують інформацію з баз даних Firebase в режимі реального часу, за допомогою бібліотеки реактивного програмування RxJS. Тобто будь-які зміни в базі даних відразу будуть відображені в клієнтському застосунку. Щоб це зробити застосовується шаблон проектування «Спостерігач». Утворюється потік даних типу Observable, на нього треюа підписатись в компоненті (рисунок 3.21).

```
public getUser(): Observable<any> {
  return from(
    this._afs.collection<any>( path: 'users-info')
        .doc(this._authService.getUserId())
        .get()
        .pipe(map( project: (user: any) => {
          return {
            ...user.data(),
            id: user.id,
          };
        })));
}
```

Рисунок 3.21 Підписка на дані користувача

Модуль реєстрації вступної заяви абітурієнта включає в себе

компонент та сервіс (рисунок 3.22). Сервіс RegistrationService має метод createStudent(), що створює реєстраційну заяву в базі даних.

Компонент подачі реєстраційної заяви на вступ до аспірантури створений за допомогою елементів Angular Material SDK. Це дозволяє користуватися компонентами в єдиному стилі, вони цілком відповідають користувацьким вимогам:

- matInput – рядок вводу строкових даних;
- matSelect – випадаючий елемент форми;
- matCheckbox – компонент, який приймає булеві значення;
- matLabel – помилка, яка відображається у формі;
- matLabel – підпис до рядку форми.

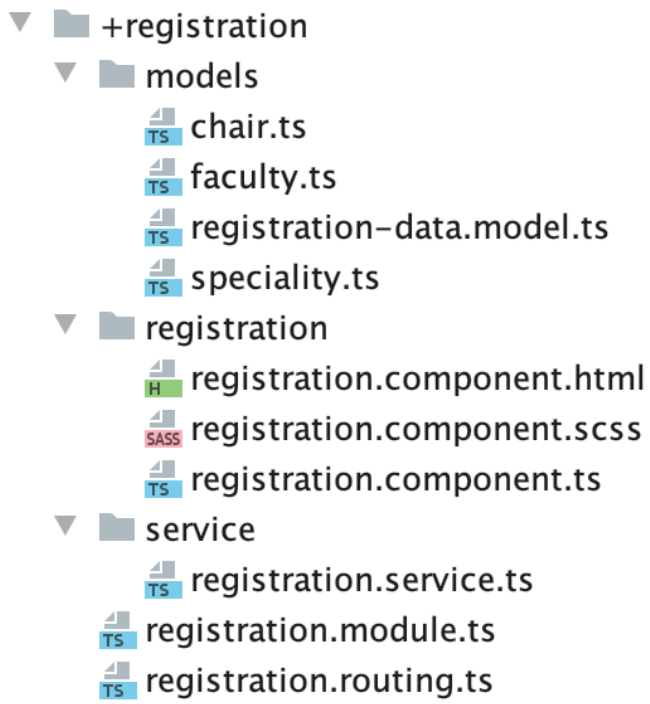


Рисунок 3.22 Структура модулю реєстрації вступної заяви абітурієнта

Модуль ядра додатку об’єднує в собі сервіс автентифікації та компоненти, які є загальними для всієї програмної системи (рисунок 3.23).

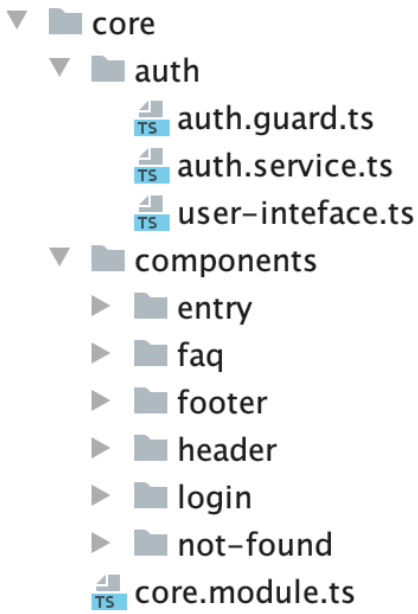


Рисунок 3.23 Структура модулю ядра додатку

Спільний модуль імпортується в інші модулі і містить декларації компонентів Angular Material SDK (рисунок 3.24).

```
@NgModule({
  exports: [
    MatButtonModule, MatIconModule, MatFormFieldModule,
    MatInputModule, MatStepperModule, MatCardModule,
    MatSelectModule, MatCheckboxModule, MatSnackBarModule,
    MatDatepickerModule, MatNativeDateModule, MatSidenavModule,
    MatTableModule, MatDividerModule, MatListModule,
  ]
})
export class MaterialModule {}
```

Рисунок 3.24 Декларації компонентів Angular Material SDK

3.3 Опис програмного використання платформи Firebase

Для розробки була використана хмарна база даних Firebase. Firebase працює з шаблоном реактивного програмування. Це надає можливість працювати та відображати дані в режимі реального часу. Для розробки системи були використані функції автентифікації, хмарних функцій і хмарного

Firestore.

Для збереження даних Firestore використовує документи та колекції. Документ – це запис, в якому записані будь-які поля. Вони об’єднуються в колекції. Також деякі документи можуть мати вкладені колекції. Якщо порівнювати з SQL-базою, то документ це поле в таблиці, а колекція і є цією таблицею. Різна кількість полів може бути в колекції (рисунок 3.25).

За параметрами запиту, у Cloud Firestore є можливість робити запити, щоб отримати документ, набір документів або всі документи у визначеній колекції. Дані можна отримати за допомогою фільтру або декількох фільтрів. В Cloud Firestore всі колекції проіндексовані за замовчуванням. З цього випливає, що продуктивність запиту пропорційна результату, а не розміру колекції.

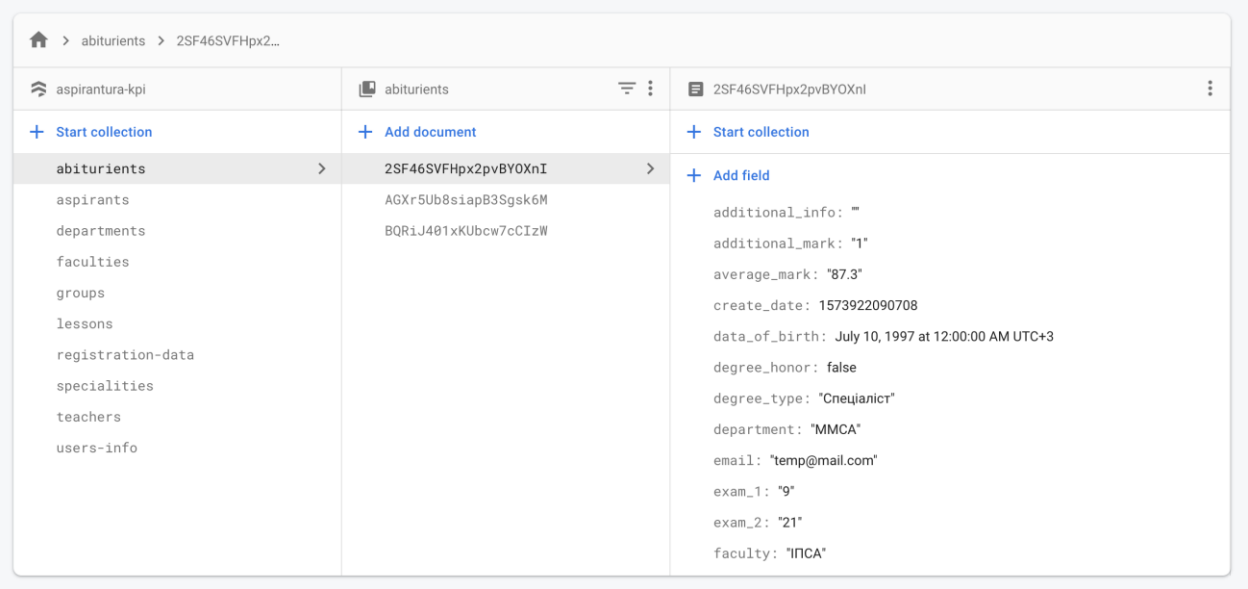


Рисунок 3.25 Структура бази даних системи

Треба отримати посилання на базу даних Firestore, щоб отримати можливість записувати дані в неї.

Після цього треба створити нову колекцію або мати посилання на вже існуючу, викликати її методом collection().

Для того щоб додавати документи в колекцію треба використовувати

метод add(). Завдяки ньому у кожного документа є унікальний згенерований ідентифікатор, який створюється автоматично. Якщо необхідно встановити власний ідентифікатор, то треба викликати спочатку метод document(), який приймає унікальний рядок ідентифікатора. Потім є можливість заповнити документ за допомогою метода set(). Якщо буде вказаний вже існуючий ідентифікатор, то система перезапише існуючий документ з цим ідентифікатором.

Найпотужніша функція Firestore – підтримка підколекцій. Є можливість додавати вкладені структури, використовуючи мінімальні ресурси.

Створення підколекції схоже на створення колекції. Для цього треба визвати метод collection() на об'єкті DocumentReference і передати йому рядок, який буде застосовуватися як ім'я підколекції.

Якщо відомо ідентифікатор документа, то його легко прочитати. Треба викликати методи collection() та document(), щоб отримати посилання на документ. Якщо ідентифікатор не відомий, то треба запустити запит для всієї колекції. API Firestore надає інтуїтивно названі методи фільтрації, такі як whereEqualTo (), whereLessThan () і whereGreaterThan (). Оскільки методи фільтрації можуть повертати кілька документів в якості їх результатів, знадобиться цикл всередині функції обробки результатів.

Якщо потрібно видалити документ з відомим ідентифікатором, все, що потрібно зробити, це мати посилання на нього, а потім використати метод delete().

Таким чином, ми ознайомилися з супроводженням програмного продукту «Система підготовки та атестації науково-педагогічних кадрів відділу аспірантури». Був створений модульний web-додаток, який містить в собі компоненти та сервіси для роботи з базою даних Firestore.

Програмне забезпечення дозволяє супроводжувати систему вступу спеціалістів науково-педагогічних кадрів вищої кваліфікації до вищого навчального закладу.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи, було проведено супроводження, доповнення та впровадження програмного продукту «Система підготовки науково-педагогічних кадрів» відділу аспірантури і докторантури університету. Ця система надає можливість створення і відстеження заяв для вступу до третього рівня вищої освіти, а також їх обробки та аналізу. Програмний продукт реалізований у вигляді web-застосунку. В процесі роботи, були покращені вміння та навички розробки програмних web-систем, зокрема за допомогою мови програмування TypeScript та фреймворку Angular, та з хмарною платформою Firebase, яка включає в себе бази даних реального часу.

В результаті виконання роботи, вдалося досягти всіх поставлених цілей дослідження, а також забезпечити майбутню підтримку системи – завдяки розробці системи на мові програмування TypeScript та фреймворку Angular, система має можливість масштабування, підключення додаткових модулів та розробці нового функціоналу.

В програмній системі передбачена можливість її використання на мобільних пристроях, що забезпечує високу доступність програмного засобу, незалежно від операційної системи, встановленої на пристрої користувача та його програмному забезпеченні.

Таким чином, розроблений web-додаток дає можливість брати участь у вступній компанії всім вступникам та з боку відділу адміністрації аспірантури керувати даними. Доступність системи зумовлена її кросплатформеністю. Створена система забезпечує прозорий процес вступної компанії до третього рівня вищої освіти навчального закладу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комолова Н. HTML, XHTML и CSS / Комолова Нина. — Москва, 2012. — 285 с.

2. Глушаков, С.В. Программирование Web-страниц. JavaScript. VBScript / С.В. Глушаков. - М: Фолио, 2002.-390 с.

3. Пасічник В.В. Веб-технології та веб-дизайн / Пасічник Володимир Володимирович. — Львів: Магнолія 2006. — 336 с.

4. Шенгили-Робертс К. CSS: каскадные таблицы стилей / Шенгили-Робертс Кейт. —Вильямс, 2005. — 717 с.

5. Хэмптон С. CSS3 для профессионалов / Сэм Хэмптон., 2016. — 183 с.

6. Catlin H, L. Michael Pragmatic Guide to SASS / Hampton Catlin, Michael Lintom. — Pragmatic Bookshelf, 2011. — 150 с.

7. Стефанов С. JavaScript. Шаблоны проектирования / Стефанов Стоян. — Санкт-Петербург: Лори, 2011. — 272 с.

8. Вайк, А. Java Script. Энциклопедия пользователя: Пер. с англ./Аллен Вайк. - К.: ООО "ТИД" ДС", 2001. - 461с.

9. Флэнаган Д. JavaScript. Подробное руководство / Дэвид Флэнаган — Символ-Плюс 2013. — 1080 с.

10. Фримен А. Angular для профессионалов/ Адам Фримен., 2018. — 800 с.

11. Файн Я., Моисеев А. Angular TypeScript. Сайтостроение для профессионалов / Яков Файн, Антон Моисеев. — Санкт-Петербург: Питер, 2018. — 464 с.

12. Sergi M. Reactive Programming with RxJS 5 / Sergi Mansilla. — Pragmatic Bookshelf, 2018. — 146 с.

13. Бен Кристенсен — «Reactive Programming with RxJava: Creating Asynchronous», 2014 р. — 374 с.

14. Drake, Mark A Comparison of NoSQL Database Management Systems and Models / Drake, Mark., DigitalOcean, 2019. — 32 с.

15. Uttam Agarwal. Hands-On Full Stack Development with Angular 5 and Firebase: Build real-time, serverless, and progressive web applications with Angular and Firebase / Uttam Agarwal, 2018 – 256 с.

16. Міністерство освіти і науки України. Умови прийому на навчання до закладів вищої освіти України в 2019 році / Міністерство освіти і науки України — Київ, 2019 – 54 с.

17. Cloud Firestore: Store and Sync app data at global scale [Електронний ресурс]. — Режим доступу до ресурсу: https://firebase.google.com/products/firestore

18. Windows Server release information [Електронний ресурс]. — Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows-server/get-started/windows-server-release-info

19. Firebase CLI reference [Електронний ресурс]. — Режим доступу до ресурсу: https://firebase.google.com/docs/cli#mac-linux-auto-script

20. Ubuntu Server Guide [Електронний ресурс]. — Режим доступу до ресурсу: https://assets.ubuntu.com/v1/f954307f-ubuntu-server-guide.pdf

21. CentOS [Електронний ресурс]. — Режим доступу до ресурсу: https://www.centos.org/about/

ДОДАТОК А

Модуль «Дослідження» Програми «Система підготовки аспірантів і докторантів університету»

СПЕЦИФІКАЦІЯ

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б

Аркушів 2

Київ 2021

Позначення	Найменування	Примітки
Документація		
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б 81-1	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТМ-71178_21Б 12-1	cabinet.service.ts	Основний сервіс кабінету
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТМ-71178_21Б 12-2	file-upload.service.ts	Сервіс для роботи з файлами
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б 12-1	abiturient.component.ts	Компонент абітурієнта
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б 12-2	registration.component.ts	Компонент для реєстрації вступника
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б 12-3	registration.service.ts	Сервіс для реєстрації
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТМ-71178_21Б 13-1	Текст програми	
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТМ-71178_21Б 14-1	Опис програми	
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТМ-71178_21Б 15-1	Діаграма прецедентів	
УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТМ-71178_21Б 16-1	Тестування системи	

ДОДАТОК Б

«Система підготовки науково – педагогічних кадрів» відділу аспірантури і докторантури університету

КОД ФРАГМЕНТУ ПРОГРАМНОЇ СИСТЕМИ

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б

Аркушів 5

2021

```
import {Component, Inject, OnInit} from '@angular/core';
import {CabinetService} from '../../services/cabinet.service';
import {ActivatedRoute, Router} from '@angular/router';
import {pluck, switchMap, take} from 'rxjs/operators';
import {MatSnackBar} from '@angular/material/snack-bar';
import { MatDialog, MatDialogRef, MAT_DIALOG_DATA } from '@angular/material';

@Component({
  selector: 'ak-abiturient',
  templateUrl: './abiturient.component.html',
  styleUrls: ['./abiturient.component.scss']
})
export class AbiturientComponent implements OnInit {
  public abiturient: any;
  public isLoadingFinished = false;

  private _abiturientId: string;

  constructor(
    private _cabinetService: CabinetService,
    private _route: ActivatedRoute,
    private _router: Router,
    private _snackBar: MatSnackBar,
    private _dialog: MatDialog
  ) {
  }

  public ngOnInit(): void {
    this._route.params.pipe(pluck('id')).subscribe((id: string) => {
```

```
            this._abiturientId = id;
            this._cabinetService.getRegistrationCard(id).subscribe((abiturient:    any)
=> {
              this.abiturient = abiturient.data();

this._cabinetService.getLanguageByShort(this.abiturient.language_type).subscribe
((language)=>{
                this.abiturient.language_type = language[0].name;
                this.isLoadingFinished = true;
              })

            });
          });
        }

        public approveRegistration(): void {
          this._cabinetService.approveRegistration(this._abiturientId,
this.abiturient)
            .pipe(
              switchMap(() => {
                return          this._snackBar.open('Абітурієнта          зареєстровано',
'OK').onAction();
              }),
              take(1),
            )
            .subscribe(() => {
              this._router.navigateByUrl('cabinet/registration-data');
            });
        }
```

```
  public openDialog() {
    const dialogRef = this._dialog.open(AbiturientDeclineComponent, {
      width: '250px',
      data: this._abiturientId
    });

    dialogRef.afterClosed().subscribe(result => {
      console.log('The dialog was closed');
    });
  }
}

@Component({
  selector: 'ak-abiturient-decline',
  templateUrl: 'abiturient-decline.component.html',
})
export class AbiturientDeclineComponent {

  constructor(
    public dialogRef: MatDialogRef<AbiturientDeclineComponent>,
    @Inject(MAT_DIALOG_DATA) public data: any,
    private _cabinetService: CabinetService,
    private _route: ActivatedRoute,
    private _router: Router,
    private _snackBar: MatSnackBar,
  ) {

}

  public onNoClick(): void {
```

```
        this.dialogRef.close();
      }

      public declineRegistration(): void {
        this._cabinetService.declineRegistration(this.data)
        .pipe(
          switchMap(() => {
            this.dialogRef.close()
            return this._snackBar.open('Реєстраційна картка видалена',
'OK').onAction();
          }),
          take(1),
        )
        .subscribe(() => {

          this._router.navigateByUrl('cabinet/registration-data');
        });

      }
    }
```

ДОДАТОК В

Адміністрування програмного продукту «Система підготовки науково – педагогічних кадрів» відділу аспірантури і докторантури університету

ОПИС ПРОГРАМНОГО КОДУ

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б

Аркушів 9

Київ 2021

АНОТАЦІЯ

Додаток містить опис програмної системи «Система підготовки аспірантів і докторантів університету», який призначений для полегшення системи вступу на третій рівень вищої освіти, а також для автоматизації процесу навчання. Система ділиться на декілька підсистем, які призначені для таких користувачів як: аспірант, викладач, адміністратор відділу аспірантури та адміністратор приймальної комісії.

Програмний продукт виконано у вигляді веб-додатку. Для розробки клієнтської частини було обрано такі мови програмування, мови розмітки та інструменти, як HTML, CSS, препроцессор SCSS, Angular, TypeScript та RxJs. Для розробки серверної частини додатку використовувалась платформа Firebase.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів програмної системи «Система підготовки аспірантів і докторантів університету».

Користувач має можливість використовувати систему на будь-якій платформі, якщо на ній є змога використовувати web-браузери (Windows, Linux, macOS, iOS, Android тощо).

Модуль дослідження був створений за допомогою наступних технологій:

- **Клієнтська частина**
    1. HTML;
    2. CSS(SCSS)
    3. Angular
    4. TypeScript
    5. RxJs
- **Серверна частина**
    1. Cloud Firestore
    2. Cloud Functions for Firebase
    3. Cloud Storage for Firebase

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Перелік функціоналу, який забезпечує програмна система «Система підготовки аспірантів і докторантів університету»:

- **Абітурієнт**
    1. Подання вступної заяви на третій рівень вищої освіти.
- **Аспірант**
    1. Перегляд особистої інформації.
    2. Перегляд навчального плану.
    3. Перегляд балів протягом семестру.
- **Адміністратор відділу аспірантури**
    1. Створення груп, кафедр, факультетів, освітніх програм, спеціальностей.
    2. Додавання нових викладачів в систему.
    3. Встановлення кількості вільних місць на спеціальності.
    4. Додавання освітньої програми до спеціальності.
- **Адміністратор приймальної комісії**
    1. Підтвердження/відхилення реєстрації вступника.
    2. Перегляд вступних заяв.
    3. Завантаження таблиць з даними про аспірантів/вступників.
    4. Перегляд списку аспірантів.
    5. Редагування вступного балу абітурієнта.
    6. Зарахування/відрахування абітурієнта.
    7. Перегляд інформаційної сторінки абітурієнта.
    8. Перегляд зарахованих аспірантів.
- **Викладач**
    1. Оцінювання аспіранта.
    2. Перегляд особистих даних.
    3. Заповнення відвідування студента.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний додаток розроблений за схемою дворівневої архітектури «Клієнт-Сервер». Доступ до бази даних здійснюється за допомогою платформи Firebase.

Для запуску системи необхідно перейти за посиланням. Після запуску з’являється головна сторінка. Далі необхідно натиснути кнопку “Мій кабінет”, після цього, з’явиться форма авторизації, де потрібно ввести дані для входу у власний кабінет користувача. Залежно від типу користувача буде доступний відповідний функціонал.

ВИКОРИСТОВУВАНІ ТЕХІЧНІ ЗАСОБИ

Програмний продукт виконано у вигляді веб-додатку. Для розробки клієнтської частини було обрано такі мови програмування, мови розмітки та інструменти, як HTML, CSS, препроцессор SCSS, Angular, TypeScript та RxJs. Для розробки серверної частини додатку використовувалась платформа Firebase.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску розробленого програмного додатку необхідно перейти за посиланням https://pgc.kpi.ua/ та авторизуватись. Додаткових інсталяцій не потрібно.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є електронна пошта та пароль для авторизації у систему.

Вхідні дані для роботи програмного додатку зчитуються з клавіатури користувача.

Вихідні дані, які отримуються в результаті роботи з програмною системою «Система підготовки науково – педагогічних кадрів» відділу аспірантури і докторантури університету є таблиці-файли зі зведеними даними про вступників та аспірантів.

ДОДАТОК Г

«Система підготовки науково – педагогічних кадрів»
відділу аспірантури і докторантури університету

ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б

Аркушів 8

2021

ТЕСТОВИЙ ВИПАДОК 1

Назва тесту: отримання електронної поданої реєстраційної картки перед підтвердженням.

Опис: перевірка можливості отримання електронної копії поданої реєстраційної картки для перегляду та перевірки інформації після заповнення та вибору всіх даних для подання заяви до вступу до третього рівня вищої освіти надана.

Передумови: наявність інтернет-з’єднання, відкрита сторінка реєстрації, проходження трьох етапів реєстрації.

Залежності: -

Післяумови: електронна копія заяви відображена у вигляді списку. Дані співпадають з заповненими.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 2

Назва тесту: вибір дати народження.

Опис: перевірка вибору дати народження для вступника. Повинна відображатися дата 20 років тому від дати реєстрації.

Передумови: наявність інтернет-з’єднання, відкрита сторінка реєстрації на першому етапі.

Залежності: -

Післяумови: відображена максимальна дата вибору дати народження 20 років тому від дати реєстрації.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 3

Назва тесту: вибір освітньої програми під час реєстрації.

Опис: перевірка вибір освітньої програми на другому етапі реєстрації. Для цього відображається випадаючий список. Для однієї спеціальності може бути одна або більше освітніх програм, а за однією програмою може здійснюватися підготовка на різних кафедрах. Факультет заповнюється автоматично на основі вибору кафедри, освітньої програми та спеціальності.

Передумови: наявність інтернет-з’єднання, відкрита сторінка реєстрації на другому етапі.

Залежності: -

Післяумови: після вибору спеціальності відображено випадаючий список для вибору освітньої програми.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 4

Назва тесту: ввести літери замість цифр в якості номеру телефону вступника.

Опис: перевірка, чи може користувач ввести неправильну інформацію в полі «Номер телефону».

Передумови: відкрита сторінка реєстрації вступника на другому етапі.

Залежності: -

Післяумови: вступник може ввести тільки цифри.

Тест не пройдено.
Помилку виправлено у зборці від 12.04.2021

ТЕСТОВИЙ ВИПАДОК 5

Назва тесту: ввести літери замість цифр в якості номеру телефону вступника.

Опис: перевірка, чи може користувач ввести неправильну інформацію в полі «Номер телефону».

Передумови: відкрита сторінка реєстрації вступника на другому етапі.

Залежності: -

Післяумови: вступник може ввести тільки цифри.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 6

Назва тесту: додавання освітньої програми для спеціальності.

Опис: перевірка, чи може адміністратор визначати загальний перелік освітніх програм та їх відповідність спеціальностям.

Передумови: відкрита сторінка адміністратора, розділ структура.

Залежності: створена необхідна спеціальність.

Післяумови: додано одну освітню програму до обраної спеціальності.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 7

Назва тесту: створення нової спеціальності в системі.

Опис: адміністратор має можливість додавати, виключати зі списку.

Передумови: відкрита сторінка адміністратора.

Залежності: -

Післяумови: створено нову спеціальність, яка відображається в списку в

розділі «Структура» на аккаунті адміністратора.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 8

Назва тесту: вибір форми оплати на заочній формі навчання.

Опис: перевірка, чи може абітурієнт обрати форму оплати за державним замовленням на заочній формі навчання. У випадаючому списку має бути тільки форма оплати «Контракт» за таких параметрів.

Передумови: відкрита сторінка реєстрації вступника на другому етапі.

Залежності: обрано заочну форму навчання під час реєстрації.

Післяумови: відображено дві форми оплати – контракт та за державним замовленням.

Тест не пройдено.
Помилку виправлено у зборці від 16.04.2021

ТЕСТОВИЙ ВИПАДОК 9

Назва тесту: вибір форми оплати на заочній формі навчання.

Опис: перевірка, чи може абітурієнт обрати форму оплати за державним замовленням на заочній формі навчання. У випадаючому списку має бути

тільки форма оплати «Контракт» за таких параметрів.

Передумови: відкрита сторінка реєстрації вступника на другому етапі.

Залежності: обрано заочну форму навчання під час реєстрації.

Післяумови: відображено одна форми оплати – контракт.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 10

Назва тесту: відображення запиту-підтвердження на видалення картки.

Опис: перевірити наявність запиту-підтвердження на видалення картки у розділі «Подані заяви» на сторінці прийомної комісії.

Передумови: відкрита сторінка приймальної комісії, обраний розділ «Подані заяви».

Залежності: відкрита анкета абітурієнта, натиснута на кнопка «Відмовити в реєстрації».

Післяумови: відображено запит-підтвердження на видалення заяви абітурієнта з бази вступників.

Тест пройдено.

ТЕСТОВИЙ ВИПАДОК 11

Назва тесту: завантаження таблиці з даними про аспірантів.

Опис: перевірка, чи може адміністратор завантажити таблицю в форматі .xmlx з даними про аспірантів.

Передумови: відкрита сторінка прийомної комісії в розділі статистика.

Залежності: натиснута кнопка «Завантажити у форматі Excel».

Післяумови: завантажено файл-таблицю з даними про аспірантів.

Тест пройдено.

ДОДАТОК Д

Адміністрування програмного продукту «Система підготовки науково – педагогічних кадрів» відділу аспірантури і докторантури університету

Довідка про впровадження результатів роботи

УКР.КПІім.ІгоряСікорського_ТЕФ_АПЕПС_ТР-72270_21Б

Аркушів 2

Київ 2021

Затверджую
Зав. аспірантури та докторантури
КПІ ім. Ігоря Сікорського
Олена ДМИТРІЄВА

АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи бакалавра Рожко Т.Ю. на тему «Адміністрування програмного продукту «Система підготовки та атестації Науково-педагогічних кадрів вищої кваліфікації» відділу аспірантури і докторантури університету», яка виконана в Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» 17 травня 2021 р.

Нами, представниками кафедри автоматизації проектування енергетичних процесів і систем КПІ ім. Ігоря Сікорського та відділом аспірантури та докторантури КПІ ім. Ігоря Сікорського, даний акт складено про те, що для використання в розробках спеціалізованого програмного забезпечення відділу аспірантури та докторантури прийняті результати дипломної роботи бакалавра Рожко Т.Ю., а саме програмне забезпечення – Web-додаток, який доступний за посиланням https://pgc.kpi.ua/, а також документацію програмного супроводу.

Представник кафедри АПЕПС КПІ ім. Ігоря Сікорського
Керівник дипломної роботи
Касьянов А.С.

Зав. аспірантури та докторантури
Дмитрієва О.І.