

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБСАЙТУ ДЛЯ ЗАЛУЧЕННЯ АБІТУРІЄНТІВ
WEBSITE DEVELOPMENT TO ATTRACT UNIVERSITY APPLICANTS

Виконав: студент(ка) 4 курсу, групи КНТ-118
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Шевердінкін І.А.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Шевєрдінкін Ілля Андрійович

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) РОЗРОБКА ВЕБСАЙТУ ДЛЯ ЗАЛУЧЕННЯ АБІТУРІЄНТІВ. WEBSITE DEVELOPMENT TO ATTRACT UNIVERSITY APPLICANTS

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27”квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 7 червня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Розробка програми. 4. Керівництво програміста. 5. Керівництво оператора. Додаток А. Текст програми. Додаток Б. Слайди презентації.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Каплієнко Т. І., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання « 04 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-5 тижні	Розділ 3
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)

(підпис)

Шевердінкін І. А.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Каплієнко Т. І.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра:
114 с., 8 табл., 56 рис., 2 дод., 17 джерел.

ВЕБСАЙТ, JAVASCRIPT, EXPRESS, REACT, REDUX, MATERIAL UI,
АБІТУРІЄНТИ

Об'єкт дослідження – проведення інформаційної кампанії в інтернеті для залучення абітурієнтів для вступу до закладу вищої освіти.

Предмет дослідження – розробка вебсайтів для залучення абітурієнтів.

Мета роботи – створення вебсайту для залучення абітурієнтів, наповненого актуальною та корисною інформацією у зручній формі.

Матеріали, методи та технічні засоби: функціональне програмування, фреймворк React, мови програмування JS.

Результати. Створено вебсайт з адаптивним інтерфейсом, що дозволяє користуватися сайтом зручно як з персонального комп'ютера, так і з телефону або планшету.

Висновки. Розроблено вебсайт, для залучення абітурієнтів, який містить необхідну для вступу та вибору спеціальності інформацію.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 114 pp., 8 tables, 56 figures, 2 appendices, 17 sources.

WEBSITE, JAVASCRIPT, EXPRESS, REACT, REDUX, MATERIAL UI,
APPLICANTS

The object of the research is to conduct an information campaign on the Internet to attract applicants to enter a higher education institution.

The subject of research is the development of websites to attract applicants.

The purpose of the work is to create a website to attract applicants, filled with relevant and useful information in a convenient form.

Materials, methods and technical means: functional programming, React framework, JS programming languages.

Results. A website with an adaptive interface has been created, which allows you to use the site conveniently both from a personal computer and from a phone or tablet.

Conclusions. Developed a website to attract applicants, which contains the necessary information for admission and choice of specialty.

ЗМІСТ

ABSTRACT	5
ЗМІСТ	6
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП.....	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Огляд предметної області	10
1.2 Опис структури та складників вебсайтів	10
1.3 Наявні рішення.....	11
1.3.1 Сайт Національного університету «Запорізька політехніка»	11
1.3.2 Сайт кафедри програмних засобів	12
1.3.3 Сайт приймальної комісії	13
1.4 Висновки.....	14
2 ОПИС ПРОГРАМИ.....	16
2.1 Вибір та обґрунтування мови програмування	16
2.1.1 Порівняння мов програмування.....	16
2.1.2 Порівняння фреймворків/бібліотек	17
2.1.3 Бібліотека React	19
2.2 Обґрунтування вибору середовища розробки	19
2.3 Опис складників програми	20
2.4 Використані бібліотеки.....	21
2.5 Обґрунтування вибору системи керування базами даних	24
2.6 Висновки розділу	26
3 РОЗРОБКА ПРОГРАМИ	27
3.1 Схема БД.....	27
3.2 Розробка інтерфейсу.....	30
3.3 Структура проекту.....	33
3.4 Використані алгоритми.....	35

3.5 Використання бібліотек та фреймворків.....	38
3.5.1 Бібліотека React	38
3.5.2 Фреймворк Express	40
3.5.3 Бібліотека Joi.....	41
3.5.4 Бібліотека Mongoose	42
3.5.5 Бібліотека Redux.....	43
3.5.6 Бібліотека MUI.....	45
4 КЕРІВНИЦТВО ПРОГРАМІСТА.....	50
4.1 Опис застосування.....	50
4.2 Інструкція по експлуатації програми	50
4.3 Взаємодія з БД	52
4.4 Хостинг	53
4.5 Помилки.....	54
5 КЕРІВНИЦТВО ОПЕРАТОРА	56
5.1 Призначення та умови виконання програми	56
5.2 Виконання програми	56
5.3 Повідомлення оператору	56
5.4 Користувачі	57
5.5 Панель адміністратора	60
ВИСНОВКИ	62
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ	63
ДОДАТОК А	64
ТЕКСТ ПРОГРАМИ.....	64
ДОДАТОК Б.....	107
СЛАЙДИ ПРЕЗЕНТАЦІЇ.....	107

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД	– База даних;
ООП	– Об’єктно орієнтоване програмування;
ПК	– Персональний комп’ютер;
Ред’юсер	– Файл, у якому зберігається стан додатку та методи для його зміни;
СКБД	– Система керування базами даних;
Ендпоінт	– Шлях, за яким можна звернутися до back-end серверу для отримання або передачі інформації;
API	– Application Programming Interface;
Cookie-файл	– Інформація, отримана від вебсайту, яка зберігається у клієнта, тобто браузера, а потім відправлена на той самий сайт, якщо його буде повторно відвідано;
IDE	– Integrated development environment;

ВСТУП

Мабуть вже кожний казав про те, як сильно інтернет став важливою та не від'ємною частиною нашого життя. Вебсайти є значною частиною взаємодії з мережею інтернет. За допомогою вебсайтів люди знаходять та поширюють інформацію, купують та продають речі, їжу, техніку тощо.

Університети всього світу вже мають або створюють свої вебсайти, сторінки у соціальних мережах аби поширювати інформацію про себе, про внутрішні заходи для студентів та можливих абітурієнтів. Національний університет “Запорізька політехніка” не виняток, у нього вже є сайт, на якому знаходиться велика кількість інформації з різною метою та для різних глядачів

Це звісно добре, коли є багато інформації, проте серед усього масиву даних буває важко, а іноді майже не можливо знайти інформацію, що потрібна саме тобі.

З цією проблемою зіштовхуються абітурієнти, коли вперше заходять на сайт університету, не важко загубитися, коли перед існує стільки різних розділів з інформацією, а користувач навіть не знає, що саме йому потрібно. Таку проблему може вирішити окремий вебсайт, на якому буде знаходитися необхідна інформація.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Вебсайт – це зазвичай досить зрозуміла структура з явно визначеними рамками, у яких він виконує ту задачу, для якої його було створено. Інколи це інформаційна довідка, як, наприклад, Вікіпедія, а інколи стримінговий сервіс як Netflix. У цій роботі розглядається вебсайт, основною задачею та ціллю якого є донести необхідну інформацію до особливого користувача, а саме абітурієнта. Зайшовши на сайт він має отримати інформацію, що допоможе йому поступити та обрати спеціальність, окрім того зацікавить його у студентському житті в університеті.

1.2 Опис структури та складників вебсайтів

Зазвичай вебсайт складається з 3 частин – це верхня частина, її називають “header” англійською, основна частина з інформацією, та нижня частина – “footer” англійською. У верхній частині найчастіше знаходиться навігація, завдяки якій можна переміщуватися сторінками сайту. Нижня частина зазвичай містить у собі коротку інформацію про власника, його пошту, телефон, адресу, інколи перелік посилань на сторінки, які вважають найпопулярнішими або найважливішими для інформування користувача. Остання за переліком, проте не за важливістю - основна частина, в ній міститься інформація з різних сторінок, буде то якесь зображення, відео чи просто текст.

Окрім зовнішньої частини, сайт має внутрішню, яка бере на себе задачі тимчасового зберігання інформації, наприклад запис деяких даних до cookie файлів або до зовнішньої бази даних(БД). Таку частину називають back-end або API. Завдяки цій частині сайт отримує можливість зберігати зміни, здійснені користувачем під час роботи з сайтом.

1.3 Наявні рішення

1.3.1 Сайт Національного університету «Запорізька політехніка»

Звісно Національний університет «Запорізька політехніка» (НУ «Запорізька політехніка») має свій сайт [1], на якому наявна величезна кількість інформації. Сайт наразі оновлюється актуальною інформацією, що дозволяє завжди бути в курсі свіжих змін або пропозицій. Сайт існує досить давно, тому вже зазнав змін у дизайні, що робить його більш сучасним та в загальному покращує зовнішній вигляд.

У наявної великої кількості інформації є і негативна сторона, новий користувач, який вперше зайшов на сайт, скоріше за все може не знайти всю необхідну інформацію через її великий обсяг.

Побачити сайт можна на рисунку 1.1.

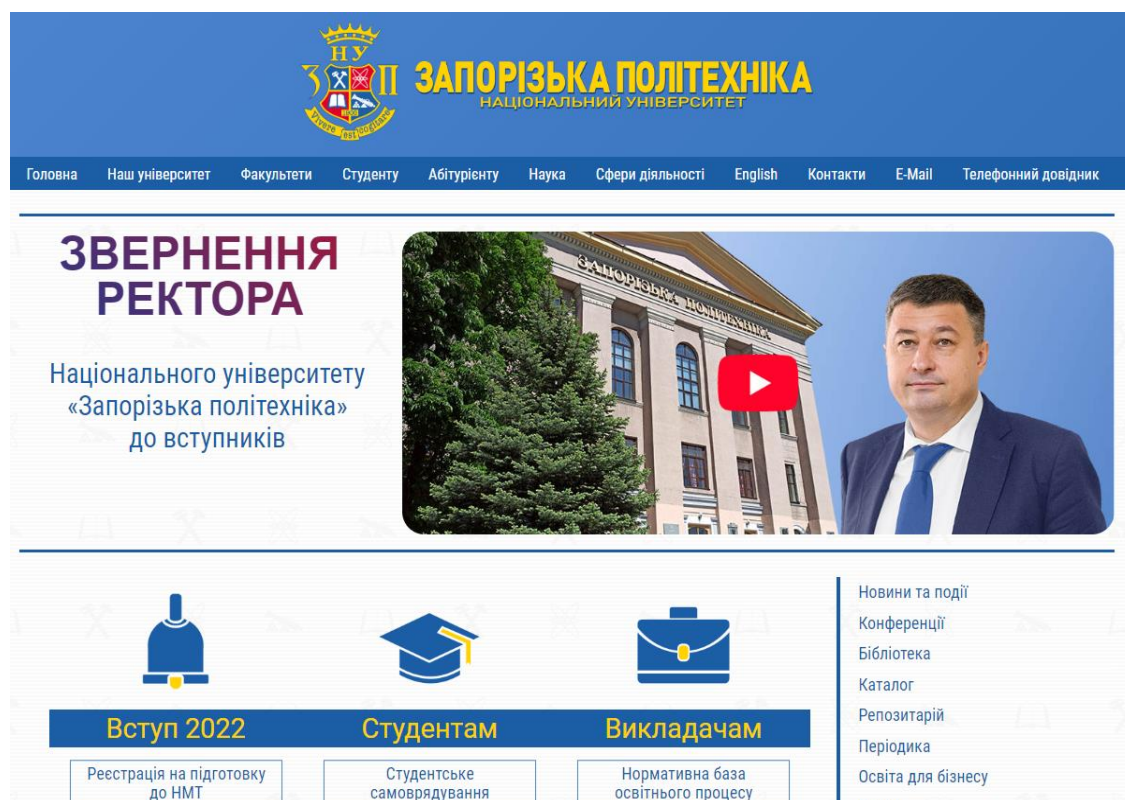


Рисунок 1.1 – Сайт НУ“ Запорізька політехніка ” [1]

1.3.2 Сайт кафедри програмних засобів

Цей сайт вже належить виключно кафедрі програмних засобів [2], що можна зрозуміти з назви. На сайті наявна інформація про кафедру, різноманітні заходи, які проводила кафедра транслювалися на сайті, поки сайт активно оновлювався. Останнє оновлення було 2017 року. Дизайн сайту досить простий, проте практичний, однак великий обсяг інформації може розгубити користувача, окрім того інформація призначена для студентів кафедри, та корисна буде здебільшого тільки їм. Сайт можна побачити на рисунку 1.2.



Рисунок 1.2 – Сайт кафедри програмних засобів [2]

1.3.3 Сайт приймальної комісії

Сайт належить приймальній комісії [3], відповідно містить інформацію пов'язану зі вступом до університету. Сайт постійно оновлюється та відображає актуальну інформацію, проте інформація наявна на сайті стосується лише умов вступу та не допоможе вступнику обрати кафедру та дисципліну для вступу.

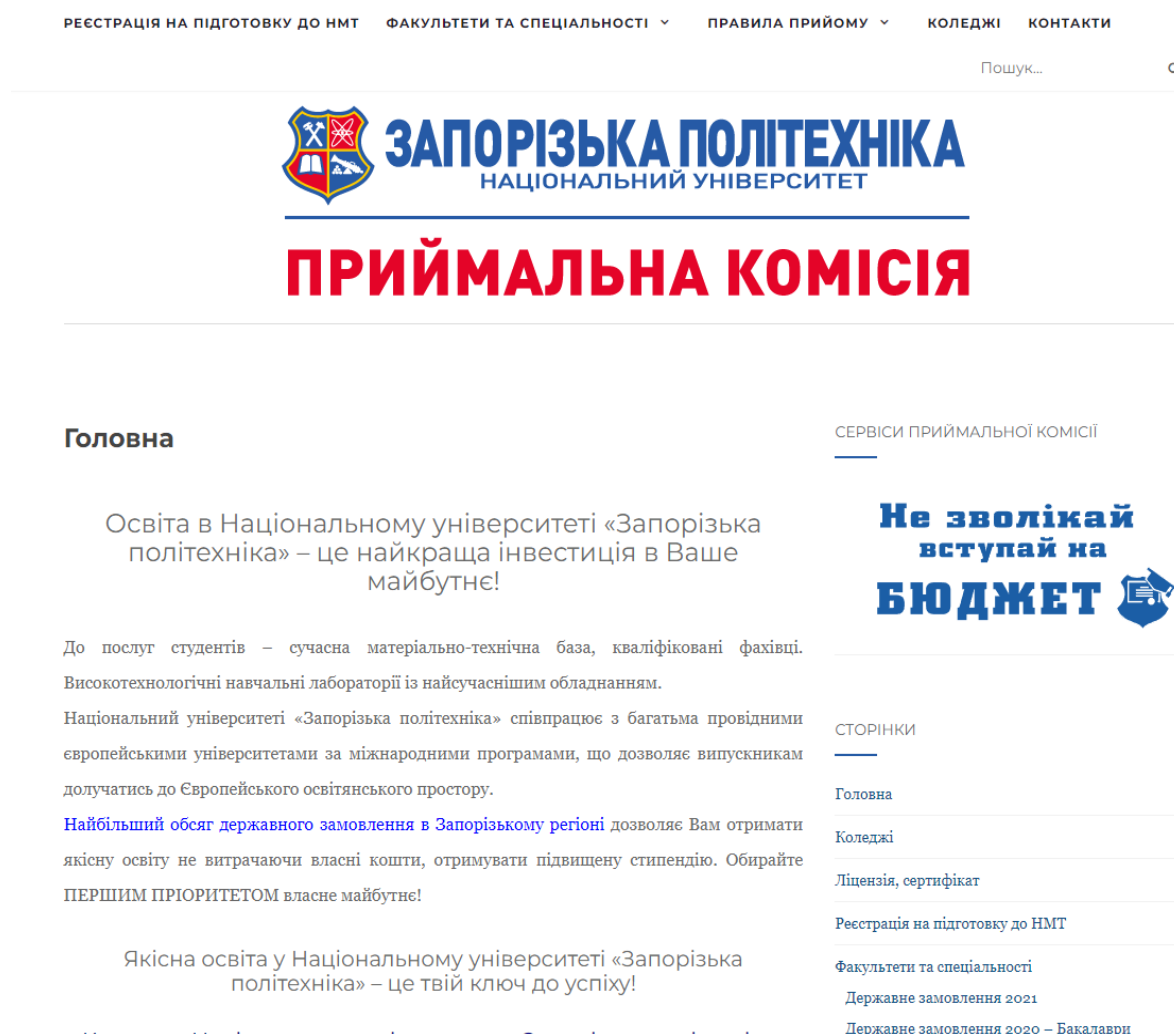


Рисунок 1.3 – Сайт приймальної комісії [3]

Порівняння сайтів наведено на таблиці 1.1.

Таблиця 1.1 – Порівняння сайтів аналогів

Сайт НУ“ Запорізька політехніка ”		Сайт кафедри програмних засобів		Сайт приймальної комісії	
Переваги	Недоліки	Переваги	Недоліки	Переваги	Недоліки
Сайт оновлюється та має актуальну інформацію	Надвелика кількість різноманітної інформації	Окремий сайт для кафедри, задовольняє потреби студента кафедри	Сайт припинив оновлення у 2017 році	Сайт оновлюється та має актуальну інформацію	Відсутність інформації про можливість подальшого працевлаштування студентів
Оновлений дизайн сприяє кращому засвоєнню та пошуку інформації	Відсутність цільового користувача сайту	Менша кількість загальної інформації та більше інформації про кафедру	Сайту не завадить оновити дизайн та технології написання	Наявність великої кількості інформації щодо вступу.	Відсутність загальної інформації про університет, кафедри та спеціальності

1.4 Висновки

У цьому розділі було проаналізовано предметну область, а саме сайти для залучення абітурієнтів, описано складники вебсайтів та більш детальний опис складників з їх можливим наповненням. Було знайдено декілька існуючих рішень, які було порівняно між собою. Проте не дивлячись на те, що було знайдено 3 приклади схожих з темою сайтів, вони все одно не задовольняють основну задачу, а саме надання та досить простий пошук необхідної інформації абітурієнту, який обрав бажаний напрям підготовки та хотів би отримати

вичерпну інформацію щодо вступу на спеціальності окремої кафедри, особливостей навчання та подальших перспектив працевлаштування.

2 ОПИС ПРОГРАМИ

Розроблюваним проєктом буде вебсайт, основною метою якого буде розповсюдження інформації, у тому числі корисної і цікавої, яка спонукатиме абітурієнта до вступу до кафедри програмних засобів та допоможе спростити процес вступу. Вебсайт матиме здебільше динамічну інформацію, якою можна буде керувати з панелі адміністратора.

2.1 Вибір та обґрунтування мови програмування

2.1.1 Порівняння мов програмування

Найпопулярнішими мовами програмування для розробки вебсайтів та серверних застосунків є JavaScript(JS), TypeScript(TS) та PHP. PHP є більш старою та стає менш популярною щороку, на відміну від JS або TS, які стають популярнішими. Особливо значну популярність отримав TS, який є нащадком JS з явною типізацією. Порівняння цих мов можна побачити на таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Назва	Переваги	Недоліки
JS	1. Відсутність явної типізації прискорює розробку. 2. Мова має велику кількість додаткових бібліотек та вбудованих функцій.	1. Відсутність явної типізації ускладнює розробку великих проєктів через неоднозначність об'єктів що може повертати функція та типів даних елементів об'єкту
TS	1.Присутність явної типізації робить розробку великих проєктів зручнішою 2. Мова має велику кількість	1. Присутність явної типізації ускладнює та робить довшою початкову стадію розробки через необхідність створення

Продовження таблиці 2.1

Назва	Переваги	Недоліки
TS	додаткових бібліотек, вбудованих функцій та інструментів	інтерфейсів.
PHP	1. Мова не конфліктує з розміткою html, що робить можливим використання PHP частини одразу під час розмітки. 2. Мова є досить простою, завдяки чому використовувати її може будь-хто	1. Мова стає менш популярною через більш зручних конкурентів 2. Сайти написані мовою PHP є менш захищеними ніж додатки написані іншими мовами

Мовою програмування для проєкту було обрано JS. Такий вибір було зроблено через те, що PHP вже морально застаріла та потребує переробки, TS є надлишково складною для серверної частини та не обов'язковою для front-end частини, тобто для написання back-end, який не є дуже великим у проєкті, не має необхідності у типізації, бо це зробить процес розробки довшим, а користі особливо не матиме знову ж таки через розмір back-end частини. У front-end JS гратиме додаткову роль, адже це більше розмітка ніж програмування, проте вбудовані у JS методи значно спростять розробку динамічної розмітки, що є однією з основних особливостей проєкту.

2.1.2 Порівняння фреймворків/бібліотек

Основою для розробки інтерфейсу програми є фреймворк. Існує декілька популярних та розвинених фреймворків та бібліотек, це React, Vue та Angular.

Для наочності різниці, їх порівняння можна подивитися на таблиці 2.2.

Таблиця 2.2 – Порівняння фреймворків/бібліотек

Назва фреймворку / бібліотеки	Переваги	Недоліки
React	1. Завдяки технології ReactDOM, має високу швидкість рендеру компонентів 2. Використовує функціональне програмування 3. Можна розроблювати як з TypeScript так і з JSX	1. Через прагнення розробників переходити на функціональне програмування, Об'єктно орієнтоване програмування стає непотрібним при розробці. 2. Суміш логіки та розмітки у JSX у великій кількості може заплутати
Vue	1. Документація описує всі вбудовані засоби фреймворку 2. Дозволяє створювати як маленькі проекти так і великі 3. Не займає багато місця	1. Має невелику кількість зовнішніх додаткових ресурсів 2. Досить важко переробити великий проект під цей фреймворк, якщо розробка не починалася на ньому
Angular	1. Має значну підтримку TypeScript 2. Має досить детальну документацію, що дозволяє легко знайти різні інструменти, що є у системі	1. Продуктивність менша за конкурентів 2. Містить велику кількість структур для використання, через що ускладнюється вивчення фреймворку

2.1.3 Бібліотека React

Бібліотекою для розробки front-end частини сайту було обрано React. Це одна з популярних бібліотек, яка має великий додатковий функціонал, який, звісно, треба спочатку встановити до проєкту. Ця бібліотека, заснована на HTML, до якої додали мову програмування JavaScript (JS). Називається це JSX, розширення React для синтаксису JS. Має вигляд звичайного скрипта написаного мовою JS, який у функціях, як відповідь повертає структуру дуже схожу на HTML.

Такий запис структури сторінок дозволяє спростити динамічність інформації, через наявність всередині елементів коду написаного мовою JS.

Окрім того React має проривну систему оновлення компонентів сторінок. До розроблення цієї системи оновлення сторінки штучно викликалося, після чого заново створювалися всі компоненти сторінки. Розроблена система дозволяла оновлювати лише ті компоненти, які змінилися та робила це самостійно у тому числі завдяки бібліотеці Redux, яка необхідна для керування станом застосунку.

2.2 Обґрунтування вибору середовища розробки

Зазвичай середовище розробки не впливає на проєкт, проте може вплинути на якість коду та швидкість його написання.

Було обрано вже перевірене середовища розробки, а саме Visual Studio Code (VSCode). Це IDE розроблене компанією Microsoft, його основною перевагою є велика кількість розширень, які можуть перетворити простий редактор коду у дещо більше, дозволивши йому керувати контейнерами від Docker, налаштовувати лінери, такі як ESLint. VSCode має розширення пов'язані з Git та ще багато іншого. Та звісно основною перевагою цієї IDE перед багатьма іншими є те, що вона безкоштовна.

Порівняння IDE можна побачити у таблиці 2.3.

Таблиця 2.3 – Порівняння середовищ розробки

Назва	Переваги	Недоліки
VSCode	1. Безкоштовний продукт з відкритим кодом 2. Велика кількість розширень 3. Досить швидкий під час роботи з декількома проєктами	1. Майже не має вбудованих функцій, які б пришвидшували написання коду 2. При встановленні великої кількості розширень стає повільнішим
WebStorm	1. Досить швидкий 2. Має велику кількість вбудованих функцій у тому числі перевірки коду, що набагато спрощує написання коду 3. Має можливість встановлення розширень, що збільшує функціонал.	1. Платний продукт, хоча не дуже дорогий 2. Багато наявних функцій користувач може и не побачити через їх вузьке призначення

2.3 Опис складників програми

Сайт складається з 2 основних частин: перша частина це зовнішній вигляд сайту, його сторінки, анімації, переходи тощо, а друга – сервера частина, у якій відбувається взаємозв'язок з БД. Обидві частини мають важливу роль у роботі сайту, проте перша частина більша за обсягом через необхідні функції сайту.

2.4 Використані бібліотеки

У частині front-end було використано багато бібліотек для керування внутрішнім станом, тобто для тимчасового зберігання актуальної інформації та даних. Такими технологіями стали:

- Redux;
- Redux-thunk;
- Universal-cookie;

Redux – це найпопулярніша бібліотека, яку використовують з React. Ця бібліотека дозволяє створювати та керувати станом застосунку. Стан застосунку це наче невелика та тимчасова БД, яку можна використовувати для відображення інформації, головною метою її використання є зменшення кількості запитів до БД, проте цей стан ні в якому разі не може замінити БД. Окрім того застосунок React-redux налаштовує зв'язок між фреймворком та бібліотекою. Завдяки такому зв'язку React може відстежувати стан застосунку та оновлювати компоненти, стан яких змінився.

Redux-thunk – це доповнення до бібліотеки Redux, яке дозволяє використовувати асинхронні функції під час запиту на зміну стану. Це полегшує та пришвидшує написання коду для зміни стану даними, які були отримані з БД, адже це є асинхронною функцією.

Universal-cookie – це бібліотека, яка допомагає працювати з cookie браузера. Cookie – це місце, куди сайт може записати інформацію, яка буде йому необхідна для подальшого використання під час роботи. Аби не було необхідності записувати таку інформацію в базу для кожного користувача окремо Cookie стали незамінними у сучасних сайтах. На прикладі розробленого сайту можна побачити, один із варіантів використання Cookie, а саме запис інформації про обрану мову. У Cookie записується мова, обрана користувачем та при наступному заході на сайт, буде обрана та мова, яку раніше обирав користувач.

Окрім технологій керування станом були задіяні такі бібліотеки:

- React-dom;
- React-dotenv;
- Axios;
- MUI;
- React-material-ui-form-validator.

React-dom – це бібліотека, завдяки якій з’являється можливість робити переходи між сторінками без дозавантаження, або перезавантаження сторінки сайту. Окрім того бібліотека, спрощує розроблення навігації сайту, оскільки має свої інструменти для навігації, можна без зайвих проблем описати існуючі ендпоінти на яких є інформація та перенаправляти на обрану сторінку у разі введення некоректного шляху.

React-dotenv – досить проста, проте дуже корисна бібліотека, завдяки якій можна отримувати значення з .env файлу. Такі файли використовують для визначення налаштувань застосунку, однак React не має вбудованих інструментів для визначення такого файлу та отримання інформації з нього, тому така бібліотека стає досить корисною, та додає сайту більшої гнучкості у налаштуванні.

Axios – бібліотека для відправки http запитів. Має розширений функціонал, дозволяє створювати шаблони за якими будуть робитися записи, управляти заголовками та тілом запитів, отримувати результат або помилку при виконанні запита. Також має велику кількість видів запитів, окрім звичайних GET, POST, бібліотека відправляє запити типів OPTIONS, PURGE, LOCK тощо.

MUI – бібліотека, яка повністю використовувалася для написання сторінок. Бібліотека має великий обсяг та різноманітність компонентів з яких дуже зручно створювати сторінки. Елементи можна модифікувати якщо потрібно, змінювати колір, або навіть написати свій компонент використовуючи один з існуючих як основу. Найчастіше при розробці сайту використовувався компонент Grid. Цей компонент дозволяє розділяти наявне на екрані місце на частини та виділяти ці частини компонентам в середині. Завдяки цьому дуже зручно робити інтерфейс, який буде на ходу змінювати свій розмір в залежності

від екрану. Звісно це не єдиний компонент, який використовувався при розробці, адже бібліотека надає великий перелік інструментів, які можна використовувати безкоштовно, у тому числі іконки, до яких всі вже давно звикли та не потребують пояснень.

React-material-ui-form-validator – це бібліотека, завдяки якій створення валідації спрощується та прискорюється в рази. Бібліотека має велику кількість вбудованих правил валідації та помилок для них, проте завжди можна створити своє правило, або змінити текст помилки. Однією з переваг цієї бібліотеки у проєкті стало те, що вона вже використовує компоненти з MUI, що дозволяє компонентам бібліотеки гарно вписатися у загальний дизайн.

Перелічені вище технології були використані для розробки front-end частини, проте у проєкті є ще серверна частина, яка була виконана з використанням мови програмування JS. Нині ця мова програмування та її спадкоємець TypeScript(TS) є лідером для розробки як back-end так і front-end. Для розробки back-end частини використовувалися наступні інструменти:

- Express;
- Joi;
- Mongoose.

Express – фреймворк на базі NodeJS, який є основою серверної частини. Його задачею є створення застосунку, визначення ендпоінтів, приймання запитів та відправка відповідей. Фреймворк має вбудовані методи обробки запитів для подальшого використання інформації, наприклад обробка тіла запиту, метод підготовлює тіло запиту до подальшого використання та робить на його основі об'єкт мови JS.

Joi – бібліотека, яка використовується для валідації об'єктів, у цьому проєкті це тіло запиту. Для цього створюються шаблони за якими проводиться перевірка. При надходженні запиту до сервера в першу чергу проводиться перевірка тіла запиту на наявність усіх необхідних полів та на типи цих полів. У разі не співпадіння з шаблоном повертається помилка, текст якої налаштовується у шаблонах.

Mongoose – бібліотека для взаємодії з БД. Оскільки обраною СКБД стала MongoDB то Mongoose є незамінною бібліотекою для взаємодії з таблицями. Бібліотека дозволяє створити схеми таблиць, за якими при підключенні таблиці будуть створені, якщо їх вже немає.

Окрім основних бібліотек на серверній частині використовувалися ще додаткові, а саме dotenv та uuid.

Dotenv дозволяє використовувати конфігураційний файл .env для налаштування змінних оточення, що є незамінною функцією сучасних серверних застосунків.

UUID – це досить проста бібліотека, яка потрібна лише для генерації унікальних ідентифікаторів.

2.5 Обґрунтування вибору системи керування базами даних

У якості системи керування базами даних(СКБД) на проєкті було обрано MongoDB. Це сучасна не реляційна БД, тобто БД, таблиці якої не потребують чіткої структури та можуть зберігати різну інформацію. Такий метод зберігання інформації є не дуже безпечним для використання у великих проєктах через неможливість керування структурою даних у таблиці. Проте у цьому проєкті серверна частина досить невелика, тому використання такої БД не створить проблем.

Головними конкурентами під час вибору СКБД були MySQL та PostgreSQL. Ці БД вже є реляційними та потребують чіткої структури таблиць. Вони також добре підходили до проєкту, проте головна перевага була на боці MongoDB. Цією головною перевагою стало те, що MongoDB зберігається на віддалених серверах, а не на локальній машині, або хостингу.

Порівняння СКБД можна побачити на таблиці 2.4.

Таблиця 2.4 – Порівняння СКБД

Назва	Переваги	Недоліки
MongoDB	БД знаходиться на віддаленому сервері, що є великою перевагою при розміщенні на хостингу Має агрегації, завдяки яким виконуються складні запити з сортуванням фільтрацією тощо	Нереляційна БД не є безпечною через можливість запису будь-якого виду даних
MySQL	Реляційна БД, що робить її більш захищеною від запису то таблиць неочікуваних даних Має базові методи SQL для побудови запитів	Для використання у серверній частині розміщених на хостингу треба спочатку розмістити десь саму базу. Не має розширених методів для перетворення даних під час виконання запитів
PostgreSQL	Реляційна БД, що робить її більш захищеною від запису то таблиць неочікуваних даних Має базові методи SQL для побудови запитів та продвинуті методи перетворення даних, що дозволяє робити більш чіткі запити для отримання інформації без необхідності її подальшої обробки	Для використання у серверній частині розміщених на хостингу треба спочатку розмістити десь саму

2.6 Висновки розділу

У другому розділі було описано та порівняно основні частини проєкту, його складники, такі як мова програмування, фреймворк, БД та бібліотеки, що використовувалися. Окрім того було описано та порівняно середовища розробки.

3 РОЗРОБКА ПРОГРАМИ

3.1 Схеми БД

Оскільки фреймворк, за допомогою якого відбувається взаємодія з БД, дозволяє створювати схеми сутностей для невеликого контролю даних, які записуються до БД, було створено схеми сутностей. Вони та їх опис наведено у таблиці 3.1.

Таблиця 3.1 – Схеми сутностей БД

Сутність \ назва у БД	Поле	Значення за замовчуван- ням	Обов'яз- кове поле	Опис поля
Цікаві матеріали Employment Materials	createdAt	Date.now()	+	Дата створення
	updatedAt	Date.now()	+	Дата оновлення
	deletedAt	-	-	Дата видалення
	_id	MongoDB object Id	+	Ідентифікатор
	info	-	+	Основна інформація
	infoEn	-		Основна інформація англійською
	link	-	+	Посилання
Faq / FaqList	createdAt	Date.now()	+	Дата створення
	updatedAt	Date.now()	+	Дата оновлення
	deletedAt	-	-	Дата видалення
	_id	MongoDB object Id	+	Ідентифікатор
	question	-	+	Текст запитання
	questionEn	-	-	Текст запитання англійською

Продовження таблиці 3.1

Сутність \ назва у БД	Поле	Значення за замовчуван- ням	Обов'яз- кове поле	Опис поля
Faq / FaqList	answear	-	+	Текст відповіді
	answearEn	-	-	Текст відповіді англійською
Партнери / Partners Employment Info	createdAt	Date.now()	+	Дата створення
	updatedAt	Date.now()	+	Дата оновлення
	deletedAt	-	-	Дата видалення
	_id	MongoDB object Id	+	Ідентифікатор
	name	-	+	Назва компанії партнера
	link	-	-	Посилання
	headerText	-	+	Текст заголовку
	mainText	-	+	Основний текст
	headerTextEn	-	-	Текст заголовку англійською
	mainTextEn	-	-	Основний текст англійською
Професії \ professionList	createdAt	Date.now()	+	Дата створення
	updatedAt	Date.now()	+	Дата оновлення
	deletedAt	-	-	Дата видалення
	_id	MongoDB object Id	+	Ідентифікатор
	name	-	+	Назва
	nameEn	-	-	Назва англійською
	requirements	-	+	Вимоги

Продовження таблиці 3.1

Сутність назва у БД \	Поле	Значення за замовчуван- ням	Обов'яз- кове поле	Опис поля
Професії professionList \	requirementsE n	-	-	Вимоги англійською
	link	-	+	Посилання
	averagePayme nt	-	+	Середня заробітна плата
	maxPayment	-	+	Максимальна заробітна плата
	vacanciesNum ber	-	+	Кількість вакансій
Мови програмуван- ня / Programming languages	createdAt	Date.now()	+	Дата створення
	updatedAt	Date.now()	+	Дата оновлення
	deletedAt	-	-	Дата видалення
	_id	MongoDB object Id	+	Ідентифікатор
	name	-	+	Назва
	writeNowPerc	-	+	Відсоток програмістів зараз
	changes	-	+	Зміна відсотку
	writeNow	-	+	Кількість програмістів, які пишуть зараз
	useSec	-	+	Кількість програмістів, які використовують мову як другу

Продовження таблиці 3.1

Сутність назва у БД \	Поле	Значення за замовчуван- ням	Обов'яз- кове поле	Опис поля
Мови програмуванн я / Programming languages	usePrim	-	+	Кількість програмістів, які використовують мову як основну
	likeIndex	-	+	Індекс вподобання

3.2 Розробка інтерфейсу

Оскільки основна частина сайту – це front-end, то на розробку та осмислення інтерфейсу було виділено значний проміжок часу.

Окрім стандартного інтерфейсу потрібну було продумати як сайт буде виглядати на мобільних пристроях, адже через менший розмір екрану такі елементи як header та footer не могли залишитися у тому самому вигляді. Основним рішенням цієї проблеми для header стало створення меню, яке часто називають “бургер” меню. Така назва скоріше за все склалася через зовнішній вигляд цієї кнопки, а саме 3 паралельні лінії, які нагадують силует бургера. Для footer все вирішилося трохи простіше, адже через не дуже великий обсяг тексту потрібно було лише розділити текст на декілька ліній.

На рисунку 3.1 можна побачити загальний дизайн для десктопів.

Для вступника	Головна	Працевлаштування	Faq
Контент сторінок			
Адреса	Контакти	Мова	

Рисунок 3.1 - Загальний дизайн для десктопів

На рисунку 3.2 можна побачити загальний дизайн для мобільних пристроїв.

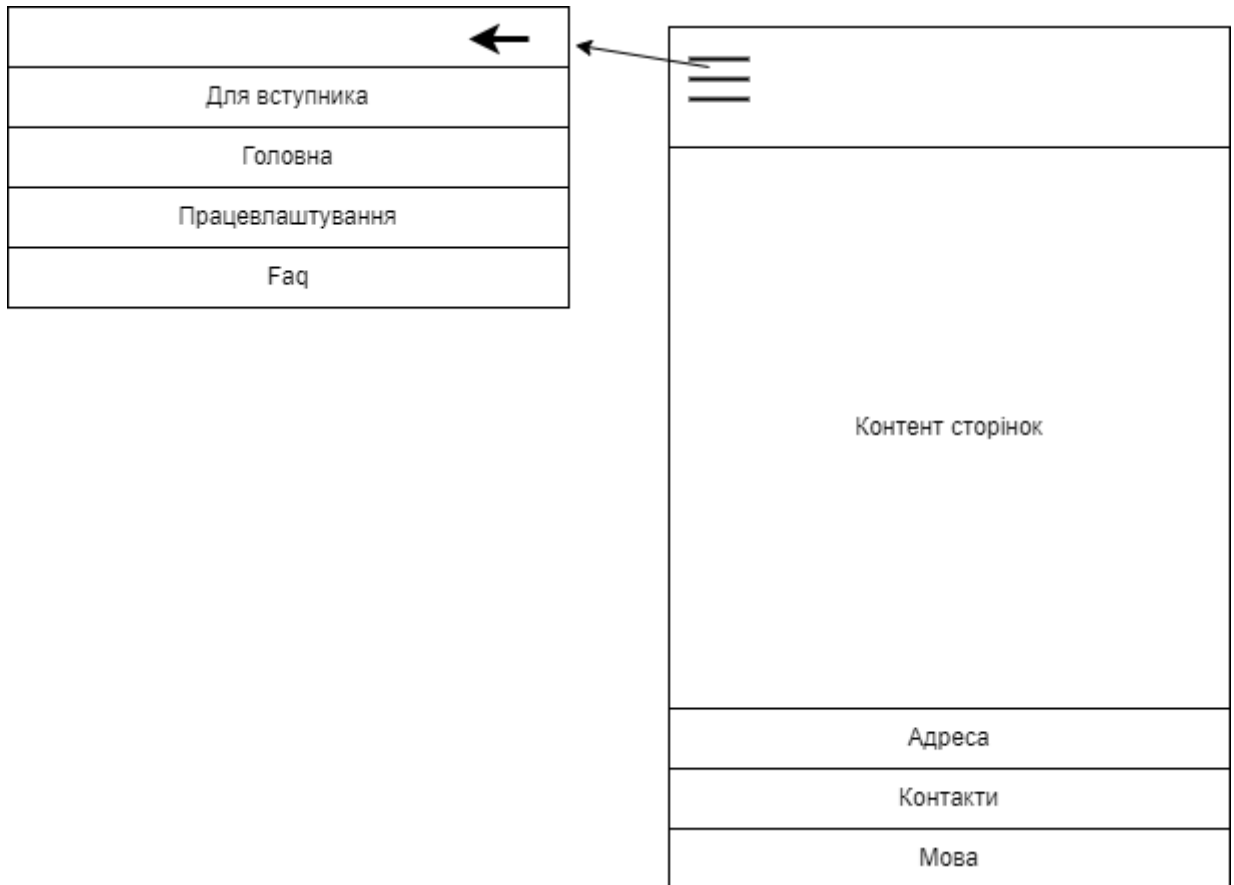


Рисунок 3.2 – Загальний дизайн для мобільних пристроїв

Основними кольорами дизайну було обрано кольори логотипу кафедри, а саме: темно-синій (#003f5a) та помаранчевий (#f18401) кольори. Темно-синього кольору було зроблено footer та неактивні елементи навігації, помаранчевим кольором було зроблено активний елемент навігації та фон основної частини.

Для відображення елемента вибору мови було обрано спливаючий список, його можна побачити на рисунку 3.3.

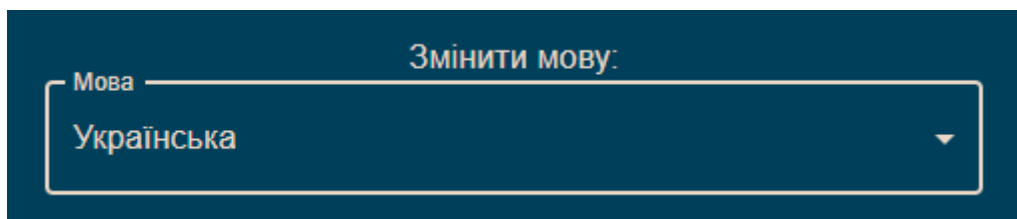


Рисунок 3.3 – Елемент вибору мову

Також для відображення інформації було використано таблицю для відображення популярності мов програмування. Цю таблицю можна побачити на рисунку 3.4.

Список мов програмування та їх популярність

Мова програмування ↓	Пишуть для роботи зараз, %	Зміни	Пишуть для роботи зараз	Використовують як додаткові	Використовують в своїх проектах	Індекс вподобання
JavaScript	18.8	+0.17	1601	3385	1089	49.8%
C#	14.7	-0.02	1252	346985	455	81.3%

Рисунок 3.4 – Таблиця мов програмування

Панель адміністратора виконана у вигляді вікон, між якими користувач перемикається для доступу до тієї ділянки інформації яка потрібна. Деталі вікон відображаються як елементи, які треба відкривати. Зовнішній вигляд панелі адміністратора можна побачити на рисунку 3.5.

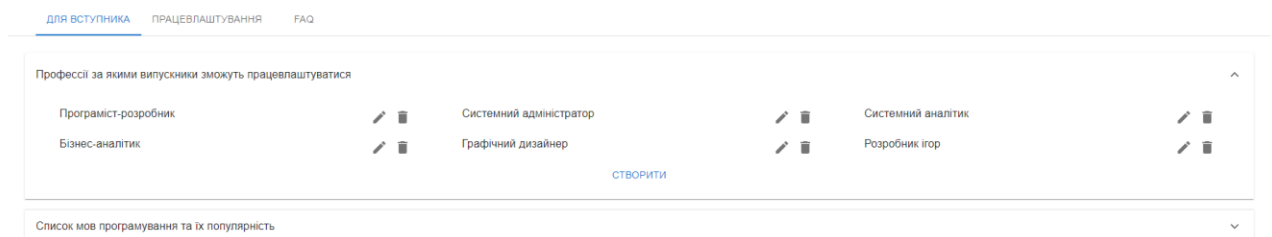


Рисунок 3.5 – Панель адміністратора

3.3 Структура проєкту

Структура проєкту досить велика, через наявність великої кількості файлів у front-end частині.

Структуру front-end частини проєкту можна побачити на рисунку 3.6.

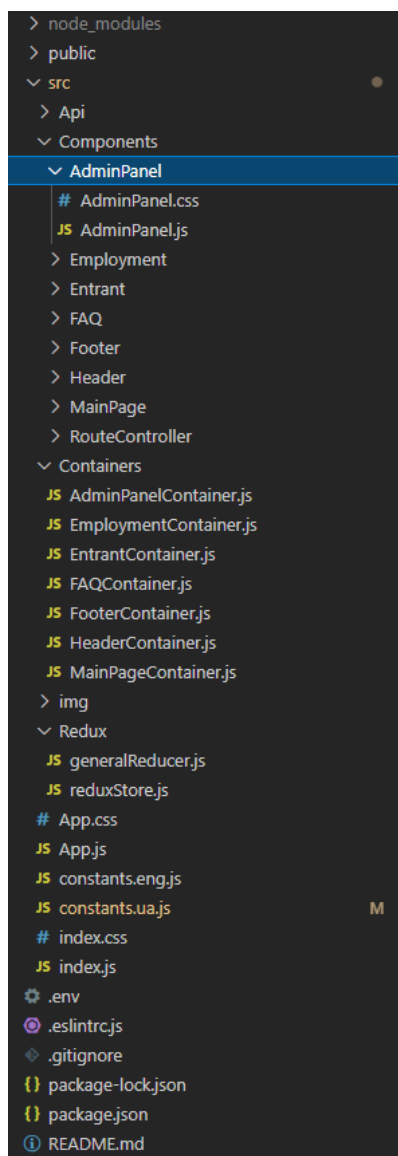


Рисунок 3.6 – Структура проєкту front-end

У корені проєкту front-end є тека `src` з усіма компонентами, контейнерами тощо, `public` з початковим файлом `html` та картинками для вікна та файли налаштувань такі як `.env`, `package.json` `eslintrc.js` тощо.

У теці `src` знаходяться теки `Components` з усіма компонентами та стилями, `Containers` з контейнерами компонент, `img` з картинками та `Redux` з ред'юсером та ініціалізаційним файлом `Redux`'у. Окрім тек є ще файли з перекладами та головні файли розмітки `index.js` та `App.js`, які викликають усі інші.

Структура back-end частини проєкту вийшла значно меншою, через меншу кількість файлів, структуру можна побачити на рисунку 3.7.

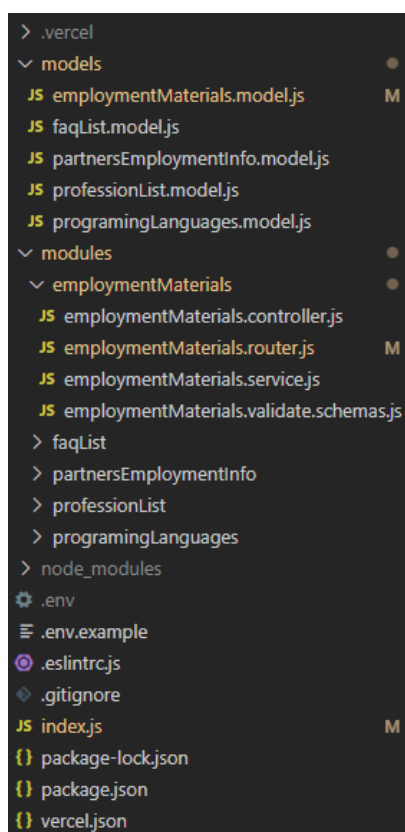


Рисунок 3.7 – Структура проєкту back-end

У корені є 4 теки, 2 з них згенеровані автоматично, це `.vercel` та `node_modules`. Тека `.vercel` створена однойменною бібліотекою, завдяки якій стало можливим викласти серверну частину на хостинг, а `node_modules` – це тека, у якій знаходяться усі бібліотеки та модулі, які використовує застосунок.

Тека `models` містить у собі моделі(схеми) БД, а `modules` – усі роутери та методи подальшого виконання запитів.

3.4 Використані алгоритми

Під час розробки проєкту було написано велика кількість алгоритмів, які спрощували та робили більш динамічним рендерінг елементів сторінки.

Наприклад для відображення професій, за якими студенти зможуть працевлаштуватися було використано наступний алгоритм(рисунок 3.8):

- беремо масив з професіями;
- через метод `map` обходимо весь масив;

- для кожного значення масиву створюємо елемент професії(приклад елемента можна побачити на рисунку 3.9);
- у результаті отримуємо деяку кількість елементів з різною інформацією про різні професії.

Схожий алгоритм було використано для багатьох елементів, бо використання такого алгоритму дозволяло системі бути динамічною та відображати стільки елементів, скільки необхідно без потреби змінювати розмітку.

```
{proffesions.map((prof) => <Grid item xs={12} md={6} className='partMainPadding' key={prof.name}>
  <Grid container className='containerFullHeight' direction={'column'} justifyContent='space-between'>
    <Typography align='center'>
      <a href={prof.link} className='link'>{prof[`name${props.language === 2 ? 'En' : ''}`]}</a>
    </Typography>
    <ul>
      {prof[`requirements${props.language === 2 ? 'En' : ''}`].map(req => <li key={req}>
        <Typography>{req}</Typography>
      </li>)}
    </ul>
    <Grid container>
      <Grid item xs={12} md={4}>
        <Typography align='center'>{phrases['AVERAGE-PAYMENT']} {prof.averagePayment}</Typography>
      </Grid>
      <Grid item xs={12} md={4}>
        <Typography align='center'>{phrases['MAX-PAYMENT']} {prof.maxPayment}</Typography>
      </Grid>
      <Grid item xs={12} md={4}>
        <Typography align='center'>{phrases['VACANCIES-NUMBER']} {prof.vacanciesNumber}</Typography>
      </Grid>
    </Grid>
    <Divider />
  </Grid>
</Grid>)}</Grid>}
```

Рисунок 3.8 – Приклад використання алгоритму з динамічною розміткою

Програміст-розробник

- Знання необхідних технологій та мов програмування, залежно від вакансії
- Уміння писати та правильно читати технічне завдання (ТЗ)
- Знання англійської мови на технічному рівні
- Soft skills: постійний саморозвиток, аналітичний склад розуму, відповідальність, комунікабельність, робота в команді.

Середня заробітна плата: 2500\$

Гранична заробітна плата: 5300\$

Кількість вакансій: 2400

Рисунок 3.9 – Блок «професії» на сторінці вступника

Ще одним алгоритмом реалізовано перехід між мовами. Для цього було створено два файли, у яких зберігалися фрази або текст на українській та англійській мовах(рисунок 3.10). Файл мав вигляд звичайного JS об'єкту, а кожен текст в обох файлах мав однакову назву. На початку формування розмітки з файлів cookie дістається обрана мова, якщо така є, та обирається відповідний файл з текстом, у разі, якщо мова не була обрана, по замовчуванню обирається українська. Таким чином не залежно від того, яка мова обрана, зі змінної, у яку записали об'єкт, з обраного файлу, дістається елемент об'єкту з текстом.

```
export const translateUkr = {
  'DEPARTMENT-NAME': 'Кафедра програмних засобів',
  'ACTIVITY-START-YEAR': 'Рік початку діяльності',
  'ADMINISTRATION': 'Керівництво',
  'ADMINISTRATION-OF-KAFEDRA': 'Завідувач кафедри: ',
  'HEAD-OF-THE-DEPARTMENT': 'д.т.н., професор Субботін Сергій Олександрович',
  'TEACHERS-LINK-TITLE': 'Перелік викладачів',
  'DESCRIPTION-1': 'Під час навчання на кафедрі студенти практикують та вивчаю',
  'DESCRIPTION-2': 'Окрім того студенти навчаються використовувати ОС Windows',
  'DESCRIPTION-3': 'Також буде можливість отримати навички програмування сучас',
  'SPECIALITIES': 'Спеціальності',
  'SPECIALITY-1': '121 – «Інженерія програмного забезпечення», освітні програм',
  'SPECIALITY-1-STUDYPROGRAM': '«Інженерія програмного забезпечення»',
  'SPECIALITY-1-DESCRIPTION': 'Підготовка фахівців, робота яких пов'язана з п',
  'SPECIALITY-2': '122 – «Комп'ютерні науки», освітні програми',
  'SPECIALITY-2-STUDYPROGRAM-1': '«Комп'ютерні науки»',
  'SPECIALITY-2-STUDYPROGRAM-2': '«Системи штучного інтелекту»',
  'SPECIALITY-2-DESCRIPTION': 'Підготовка фахівців з проектування й реалізації',
  'HEADER-FAQ': 'FAQ',
  'HEADER-SPECIALITIES': 'Спеціальності',
  'HEADER-MAIN': 'Головна',
  'HEADER-STUDY-ABROAD': 'Навчання за кордоном',
  'HEADER-FOR-ABITURIENT': 'Для вступника',
  'HEADER-WORK': 'Працевлаштування',
  'FOOTER-ADRESS-TITLE': 'Адреса:',
  'FOOTER-PHONE-TITLE': 'Телефон:',
  'FOOTER-EMAIL-TITLE': 'Електронна пошта:',
  'FOOTER-ADRESS': 'вул. Жуковського 64, м. Запоріжжя, Україна, 69063',
  'FOOTER-PHONE': '+380(61)7698267',
  'FOOTER-EMAIL': 'kafedra_pz@zntu.edu.ua',
  'FOOTER-LANGUAGE-CHANGE': 'Змінити мову:',
  'FOOTER-LANGUAGE-TITLE': 'Мова',
  'STUDY-PRICE-TITLE': 'Вартість навчання',
}
```

Рисунок 3.10 – Частина перекладу тексту

3.5 Використання бібліотек та фреймворків

У другому розділі була зазначена деяка кількість бібліотек та фреймворків, які використовувалися під час розробки. У цьому розділі можна буде побачити детальніше призначення багатьох фреймворків та бібліотек, використаних у проєкті.

3.5.1 Бібліотека React

React це бібліотека, завдяки якій було розроблено всю front-end частину. Початок роботи з бібліотекою відбувається одразу після виконання команди `prx-create-react-app`. Ця команда створює усе необхідне для мінімальної роботи проєкту. Створює головний файл та перший компонент.

Компонент створений бібліотекою може мати вигляд функції або класу. Оскільки розробники бібліотеки намагаються відмовитися від класів – усі компоненти у проєкті мають вигляд функцій(рисунок 3.11).

```
const FAQ = (props) => {
  const phrases = props.language === 1 ? translateUkr : translateEng;

  return <Paper elevation={10} className='paper'>
    <Typography align='center' variant='h5'>{phrases['FAQ-TITLE']}

```

Рисунок 3.11 – Функціональна компонента React

React має вбудовані методи, які дозволяють виконувати різні дії після або до того як компонент буде відображено. Таким чином у проєкті відбувається завантаження даних з БД(рисунок 3.12) та виявлення у Cookie файлах обраної раніше мови.

```
useEffect(() => {
  if (props.isAuthenticated)
    props.getAllData(props.setAllData);
}, [props.isAuthenticated]);
```

Рисунок 3.12 – Завантаження даних з бази до відображення компоненти

Окрім того React має зовнішню бібліотеку react-dom, завдяки якій можна виконувати адресування між компонентами, в залежності від шляху(рисунок 3.13). У разі якщо не знайдено компоненту за вказаним шляхом, буде зроблено перенаправлення на стартову сторінку. Компонент NavLink дозволяє виконувати переходи між посиланнями сайту без перезавантаження сторінки, завдяки чому переходи між елементами сайту стають дуже плавними та зручними.

```
const RouteController = () => (
  <Box style={{ padding: '20px' }}>
    <Routes>
      <Route exact path={'/home'} element={<MainPageContainer />} />
      <Route exact path={'/forentrant'} element={<EntrantContainer />} />
      <Route exact path={'/employment'} element={<EmploymentContainer />} />
      <Route exact path={'/faq'} element={<FAQContainer />} />
      <Route exact path={'/admin'} element={<AdminPanelContainer />} />
      <Route path="*" element={<Navigate to="/home" />} />
    </Routes>
  </Box>
);
```

Рисунок 3.13 – Компонент маршрутизації

3.5.2 Фреймворк Express

Express став основою back-end частини проєкту. Завдяки ньому проводиться ініціалізація сервера, визначення ендпоінтів, обробка помилок та запитів, авторизація запитів.

Основна логіка роботи фреймворку знаходиться у головному файлі, бо перше, що має робити back-end застосунок – це приймати запити та відповідати на них. Для визначення ендпоінтів запитів Express має універсальний інструмент `.use()`. Цей метод використовується для визначення ендпону, для обробки помилок, для додавання бібліотек у роботу з запитами тощо. У проєкті цим методом реалізовано визначення ендпоінтів, додання сторонніх бібліотек, обробка помилок та авторизація запиту. Реалізацію цих елементів можна побачити на рисунку 3.14.

```
app.use(express.json());
app.use(cors());

app.use((req, res, next) => {
  console.log(req.body, req.url, req.method);
  if(req.url === '/test') next();
  const authToken = req.headers.authorization;
  if (!!authToken || authToken !== process.env.AUTH_TOKEN) && req.method !== 'OPTIONS')
    throw new Error(JSON.stringify({ message: 'Not valid auth token', status: 'Forbidden', code: 403 }));
  next();
});

app.use('/programmingLanguages', programmingLanguagesRouter);
app.use('/professionList', professionListRouter);
app.use('/partnersEmploymentInfo', partnersEmploymentInfoRouter);
app.use('/faqList', faqListRouter);
app.use('/employmentMaterials', employmentMaterialsRouter);

app.use(function (err, req, res, next) {
  console.log(err);
  let errorContent;
  try {
    errorContent = JSON.parse(err.message);
  }
  catch (e) {
    const isJoiError = !!err?.error?.details;
    const joiMessage = isJoiError ?
      err.error.details.map(detail => detail.message,) : null;
    errorContent = { message: joiMessage || err.message };
  }
  res.status(500).send({ status: 'Internal Server Error', ...errorContent });
  next();
});
```

Рисунок 3.14 – Використання методу `.use()` у проєкті

Для відрізнєння ендпоїнтів та логіки від основного файлу було створено так звані роутери, тобто файли, які містять визначення ендпоїнтів для окремої частини. Один з роутерів можна побачити на рисунку 3.15.

```
const router = express.Router();
const validator = createValidator();

router.get('/:id', controller.findOne);
router.get('/', controller.findAll);
router.post('/', validator.body(employmentMaterialsCreateSchema, {passError: true}), controller.create);
router.put('/:id', validator.body(employmentMaterialsUpdateSchema), controller.update);
router.delete('/:id', controller.delete);

module.exports = router;
```

Рисунок 3.15 – Роутер

3.5.3 Бібліотека Joi

Бібліотека Joi – це бібліотека для валідації даних. Вона використовується у back-end частині для валідації тіла запиту. Так наприклад для створення у БД запису про мову програмування необхідно, щоб у запиті була назва цієї мови та інші обов’язкові поля, у разі їх відсутності у запиті повернеться помилка з текстом про відсутні поля. Для кожного запиту, де необхідна валідація створюється окрема схема, або якщо це можливо використовується вже існуюча. Приклад схеми валідації можна побачити на рисунку 3.16.

```
const Joi = require('joi');

module.exports = {
  employmentMaterialsCreateSchema: Joi.object({
    info: Joi.string().required(),
    infoEn: Joi.string().optional(),
    link: Joi.string().required(),
  }).options({ abortEarly: false }),
  employmentMaterialsUpdateSchema: Joi.object({
    info: Joi.string().optional(),
    infoEn: Joi.string().optional(),
    link: Joi.string().optional(),
  }).options({ abortEarly: false }),
};
```

Рисунок 3.16 – Схема валідації запиту

Проте бібліотека Joi сама по собі не здатна працювати з фреймворком Express, тому, для налагодження їх взаємодії, використовується додаткова бібліотека `express-joi-validation`. Єдине призначення цієї бібліотеки – це передати у роутер Express функцію, яка буде перевіряти тіло запиту за тими критеріями, які вказані у схемі.

3.5.4 Бібліотека Mongoose

Бібліотека `mongoose` використовується у проєкті для взаємодії back-end частини з СКБД, якою є MongoDB. Головними функціями бібліотеки є підключення до БД, створення схем та взаємодія з БД, для отримання або відправки туди інформації.

Для підключення до БД необхідно мати лише строку `url`, за якою здійснюється підключення, далі бібліотека з MongoDB зроблять все самі.

Для взаємодії з окремими таблицями необхідно створити схеми цих таблиць. Не дивлячись на те, що БД не реляційна та не має чіткої структури таблиць, для взаємодії з ними все ж потрібно створити деяку структуру. Приклад структури можна побачити на рисунку 3.17.

```
const schema = new Schema({
  createdAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  updatedAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  deletedAt: {
    type: Date,
    default: null,
  },
  _id: {
    type: String,
    require: true
  },
  info: {
    type: String,
    require: true
  },
  infoEn: {
    type: String,
  },
  link: {
    type: String,
    require: true
  },
});

module.exports = model('employmentmaterials', schema);
```

Рисунок 3.17 – Схема таблиці Mongoose

3.5.5 Бібліотека Redux

Бібліотека Redux використовується на front-end частині для керування станом застосунку. Для цього створюється так званий ред'юсер, який зберігає в собі початковий стан застосунку та методи для його зміни. Ред'юсер можна побачити на рисунку 3.18.

```

> const initialState = { ...
  };

  const SET_LANGUAGE = 'SET_LANGUAGE',
        GET_ALL_DATA = 'GET_ALL_DATA',
        SET_AUTH = 'SET_AUTH',
        SET_NEW_ENTITY_DATA = 'SET_NEW_ENTITY_DATA',
        SET_UPDATED_ENTITY_DATA = 'SET_UPDATED_ENTITY_DATA',
        DELETE_ENTITY_DATA = 'DELETE_ENTITY_DATA';

  const generalReducer = (state = initialState, action) => {
    switch (action.type) {
>     case SET_LANGUAGE: { ...
      }
>     case GET_ALL_DATA: { ...
      }
>     case SET_NEW_ENTITY_DATA: { ...
      }
>     case SET_UPDATED_ENTITY_DATA: { ...
      }
>     case DELETE_ENTITY_DATA: { ...
      }
>     case SET_AUTH: { ...
      }
>     default: ...
    }
  };

> export const setLanguage = (language) => { ...
  };

> export const setAllData = (data) => { ...
  };

```

Рисунок 3.18 – Ред'юсер

Для отримання даних з ред'юсера у компонентах створюються контейнери, які викликаються замість компонент, отримують необхідні дані та викликають компоненту з цими даними. Приклад контейнера можна побачити на рисунку 3.19.

```
const EmploymentContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <Employment {...props} />
  );
};

const mapStateToProps = (state) => {
  return {
    language: state.general.language,
    partnersEmploymentInfo: state.general.partnersEmploymentInfo,
    employmentMaterials: state.general.employmentMaterials,
  };
};

export default connect(mapStateToProps, {})(EmploymentContainer);
```

Рисунок 3.19 – Контейнер компоненти

Проблемою бібліотеки Redux є те, що вона не може працювати з асинхронними операціями, тобто з усіма операціями, які виконуються не миттєво, наприклад, це запити до БД. Для отримання цієї інформації у ред'юсері та записі її до стану застосунку необхідна додаткова бібліотека, яка має назву Redux-thunk. Ця бібліотека дозволяє використовувати асинхронні операції у методах для зміну стану застосунку. Метод трохи змінює зовнішній вигляд, проте має все той же функціонал, а саме зміна інформації у стані.

3.5.6 Бібліотека MUI

Бібліотека MUI є основним постачальником готових компонент у проєкті. Красиві та водночас зручні компоненти, які надає ця бібліотека використані всюди. Бонусом до великої кількості компонент є можливість їх персоналізації, тобто за бажанням можна створити свою компоненту, взявши одну з існуючих за основу, або якщо зміни потрібні не такі кардинальні, а наприклад проста зміна кольору, то можна використати одну з існуючих тем кольору, або створити свою.

Основні компоненти бібліотеки які було використано та їх опис можна побачити на таблиці 3.2.

Таблиця 3.2 – Використані компоненти бібліотеки MUI

Назва	Опис
Accordion	Елемент, який має вигляд простого блоку з різним можливим змістом, при натисканні на який відкривається прихована частина блоку з додатковим текстом(рисунок 3.20)
Paper	Звичайний блок контейнер, особливістю якого є тінь, яка змінюється параметром elevation, що візуально підіймає або опускає елемент
Tab	Горизонтальна панель з різною кількістю вікон між якими можна переходити, для кожного вікна можна зробити окрему деталізацію, яка розташована нижче(рисунок 3.21)
Typography	Елемент для відображення тексту, при створенні має більш приємний до ок розмір, може вирівнюватися по горизонталі та вертикалі та змінювати елемент, від якого походить(h1, h2, span тощо)
Grid	Контейнер, особливістю якого є можливість регулювати розмір по ширині в залежності від розміру екрану. Екран поділяється на 12 блоків, а в деталях компонента можна вказати скільки блоків він має займати
IconButton	Кнопка круглої форми, призначена для відображення іконкою
Dialog	Компонент, що з'являється поверх інших для відображення критичної інформації або необхідності дій користувача(рисунок 3.22)
Button	Закруглена кнопка, має декілька стилів, у тому числі заблокований, при якому на неї не можна натиснути
TextField	Строка для введення тексту, має декілька стилів, може бути багаторядковою, також є функція, при якій строка займає всю можливу ширину

Продовження таблиці 3.2

Назва	Опис
Divider	Невеликий за шириною блок для візуального відділення однієї інформації від іншої
Table	Таблиця для відображення різної інформації, може мати сортування(рисунок 3.23)
Select	Впливаючий список з різними опціями(рисунок 3.24)

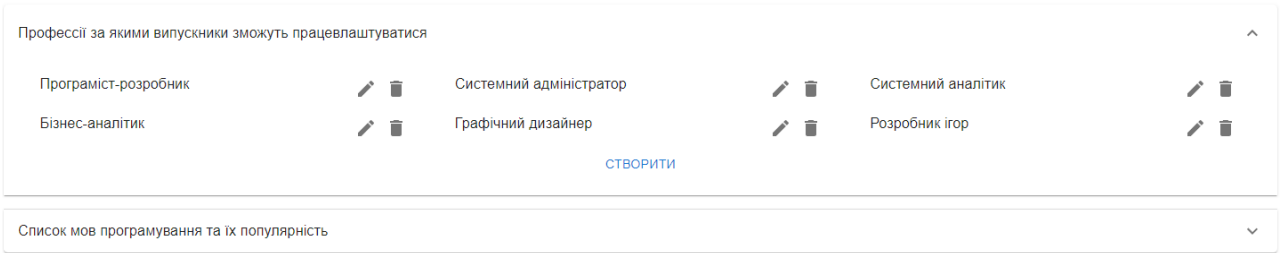


Рисунок 3.20 – Елемент Accordion

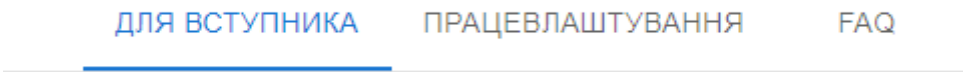


Рисунок 3.21 – Елемент Tab

Оновити об'єкт professionList

Назва
Програміст-розробник

Назва англійською
Developer

Вимоги
Знання необхідних технологій та мов програмування, залежно від вакансії
Уміння писати та правильно читати технічне завдання (ТЗ)
Знання англійської мови на технічному рівні
Soft skills: постійний саморозвиток, аналітичний склад розуму, відповідальність, комунікабельність, робота в команді.

Вимоги англійською
Knowledge of required technologies and programming languages, depending on the vacancy
Ability to write and correctly read the technical task (TOR)
Knowledge of English at the technical level
Soft skills: constant self-development, analytical mind, responsibility, sociability, teamwork.

Посилання
<https://www.work.ua/jobs-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BD%D0%B8%D0%BA/>

Середня заробітна плата
2500\$

Максимальна заробітна плата
5300\$

Кількість вакансій
2400

Рисунок 3.22 – Елемент Dialog

Список мов програмування та їх популярність

Мова програмування ↓	Пишуть для роботи зараз, % ↑	Зміни	Пишуть для роботи зараз	Використовують як додаткові	Використовують в своїх проектах	Індекс вподобання
JavaScript	18.8	+0.17	1601	3385	1089	49.8%
C#	14.7	-0.02	1252	346985	455	81.3%

Рисунок 3.23 – Елемент Table

Змінити мову:

Мова
Українська

Рисунок 3.24 – Елемент Select

Окрім цих компонентів було використано компонент з окремої бібліотеки. Цей компонент використовується для валідації даних при записі.

Окремою перевагою стало те, що базується компонент на TextField з бібліотеки MUI, через що добре вписується у загальний дизайн. Компонент перевіряє поле на задані характеристики та у разі не співпадіння відображає помилку(рисунок 3.25).

Назва

Це поле є обов'язковим

Назва англійською

Вимоги

n

Мінімальна довжина поля - 2 символи(ів)

Рисунок 3.25 – Компонент строки з валідацією

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Опис застосування

Розроблене програмне забезпечення, призначене для надання необхідної інформації потенційному абітурієнту, тобто має надати йому ту інформацію, яка буде корисна та водночас цікава та сприятиме до вступу.

Сайт виконує декілька задач:

- інформування користувача про деталі вступу та можливості подальшого розвитку та працевлаштування;
- надання необхідних контактних даних, які можуть знадобитися;
- підтримка інформації сайту актуальною адміністратором сайту;

Єдиною областю, у якій можна використовувати сайт є інформування вступників та агітація до вступу.

Обмеженням застосування є необхідність підключення інтернет з'єднання, оскільки це вебсайт і він буде розміщатися на хостингу, для доступу до нього необхідно мати інтернет.

4.2 Інструкція по експлуатації програми

Першим пунктом експлуатації вебсайту є її налаштування та запуск. Для налаштування сайту на front-end та back-end частинах є файл .env у ньому знаходяться налаштування проєкту. Змінні для налаштування з їх описом можна побачити на таблиці 4.1.

Таблиця 4.1 – Налаштування проєкту та опис змінних

Назва змінної	Опис	Приклад значення
AUTH_TOKEN	Зберігає строку, яка буде використовуватись як пароль на front-end частині для	v&?GLbcfU?mR3Lu*wFgV T-zt-hW8*zk@

Продовження таблиці 4.1

Назва змінної	Опис	Приклад значення
	доступу до панелі адміністратора та додаватися у заголовки запитів, а на back-end частині буде перевіряти заголовок запиту для його авторизації.	
DATABASE_CONNECTION_STRING	Зберігає посилання для підключення бази даних на back-end частині. Можна отримати на сайті MongoDB у налаштуваннях підключень.	mongodb+srv://Mainsed:<password>@cluster0.e4eoo.mongodb.net/?retryWrites=true&w=majority
BACKEND_URL	Зберігає посилання на back-end частину. Оскільки вони знаходяться у різних проєктах, то і на хостингу зберігаються під різними посиланнями	https://back-kd1ke34ax-mainsed.vercel.app/
PORT	Зберігає порт для локального запуску проєкту	3000

Після завершення налаштування та заповнення усіх необхідних змінних необхідно запустити проєкт. Для цього необхідно виконати команди `npm install` на обох частинах проєкту, після завершення роботи команди потрібно виконати `npm run start` на back-end частині та `npm start` на front-end частині. Команду `npm install` необхідно виконувати лише під час першого запуску для встановлення усіх необхідних модулів та бібліотек. Це може не знадобитися під час завантаження проєкту до хостингу, бо часто хостинг робить завантаження та

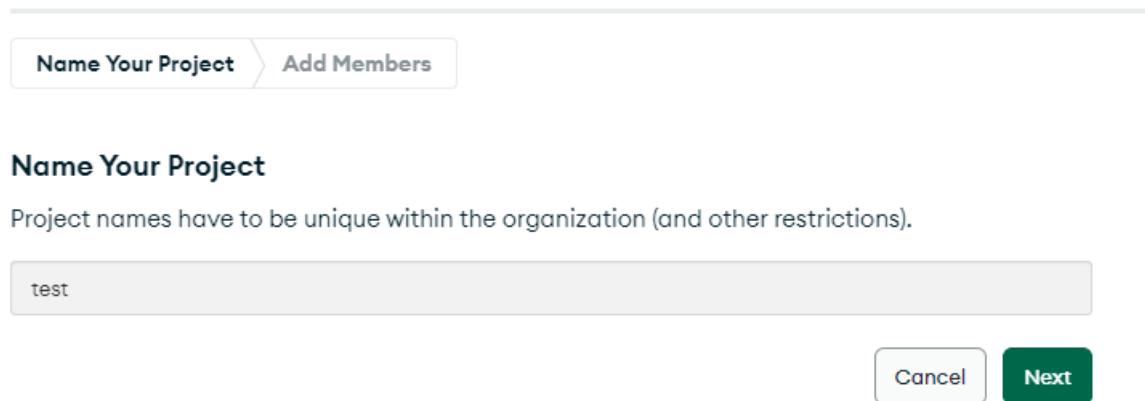
запуск проєкту сам, проте можуть знадобитися самі команди, щоб вписати їх то відповідних полів на сторінці хостингу.

4.3 Взаємодія з БД

Для початку взаємодії з MongoDB необхідно зайти на сайт з доступом до бази даних MongoDB[4], зареєструватися або увійти до свого аккаунту та обрати або створити проєкт(рисунок 4.1). Далі необхідно обрати кластер, на якому буде зберігатися БД. Від цього залежить можливості обсягу та обороту БД. Є як платні так і безплатні кластери у різних регіонах(рисунок 4.2). Після цього вже можна створювати або передивлятися існуючі таблиці у базі.

Подальша взаємодія з таблицями, тобто їх створення, запис даних, тощо відбуватиметься через back-end частину проєкту.

Create a Project



Name Your Project Add Members

Name Your Project

Project names have to be unique within the organization (and other restrictions).

test

Cancel Next

Рисунок 4.1 – Створення проєкту у MongoDB

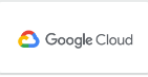
Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

PREVIEW Serverless
Dedicated
FREE Shared

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.
 No credit card required to start. Upgrade to dedicated clusters for full functionality.
 Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region AWS, Stockholm (eu-north-1) ▼


★ Recommended region ⓘ 🏷️ Dedicated tier region ⓘ

NORTH AMERICA	EUROPE	AUSTRALIA
 Oregon (us-west-2) ★	 Stockholm (eu-north-1) ★	 Sydney (ap-southeast-2) ★
 N. Virginia (us-east-1) ★	 Frankfurt (eu-central-1) ★	ASIA
 Ohio (us-east-2) ★ 🏷️	 Paris (eu-west-3) ★	 Singapore (ap-southeast-1) ★

Рисунок 4.2 – Створення кластера у MongoDB

4.4 Хостинг

Оскільки сайт було викладено на хостинг, то вибір та пошук хостингу зайняв досить великий проміжок часу. Остаточним варіантом став хостинг Vercel. Перш за все тому, що він надає послуги безкоштовного хостингу, а це було основним критерієм пошуку. Також досить важливим критерієм була можливість хостингу запускати у себе проєкти на фреймворках React та Express. Оскільки front-end та back-end на проєкті працюють окремо, то і хостити їх треба було окремо. Більшість хостингів, що пропонували умови безкоштовного хостингу не підтримували Express застосунки, або були умовно безкоштовними. Єдиний варіант, який задовільнив усі потреби – це Vercel.

4.5 Помилки

У проєкті є помилки, які зможе побачити не кожен користувач через місце їх відображення. Помилки запитів до back-end відображатимуться у панелі з інструментами для налагодження у розділі Network(рисунок 4.3). Там відображатимуться помилки, які повертає back-end. Помилки можуть бути пов'язані з непройденою авторизацією запиту(рисунок 4.4), валідацією тіла запиту(рисунок 4.5), або інші внутрішні помилки пов'язані з бібліотеками, які використовуються.

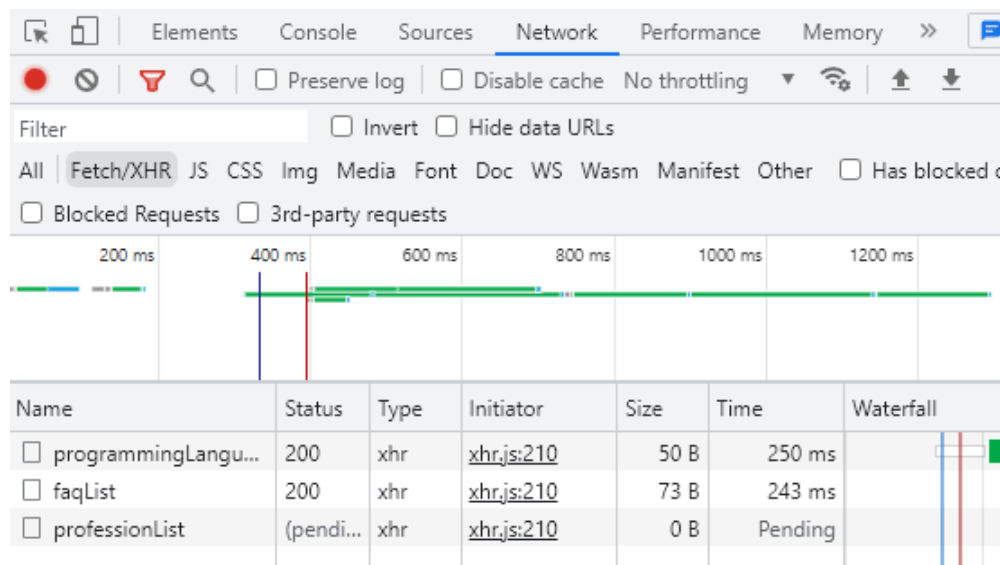


Рисунок 4.3 – Панель налагодження розділ Network

```

{
  "status": "Forbidden",
  "message": "Not valid auth token",
  "code": 403
}

```

Рисунок 4.4 – Помилка авторизації запиту

```
{
  "status": "Internal Server Error",
  "message": [
    "\"question\" is required",
    "\"answer\" is required"
  ]
}
```

Рисунок 4.5 – Помилка валідації тіла запиту

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення та умови виконання програми

Розроблене програмне забезпечення, призначене для надання необхідної інформації потенційному абітурієнту, тобто має надати йому ту інформацію, яка буде корисна та водночас цікава та сприятиме до вступу.

Для коректного функціонування вебсайту необхідно:

- у випадку викладення сайту на хостинг необхідно мати будь-який пристрій, здатний використовувати мережу інтернет та мати один з популярних браузерів останньої версії;
- для коректної роботи сайту на персональному комп'ютері(ПК) необхідно мати один з популярних браузерів останньої версії, стабільне інтернет з'єднання та ПК, здатний витримати навантаження, які дає браузер;
- для коректної роботи сайту на мобільному пристрої необхідно мати телефон з популярним браузером оновленим до останньої версії та стабільне інтернет з'єднання.

5.2 Виконання програми

Для запуску сайту необхідно перейти на сторінку сайту за посиланням, яке визначено на хостингу. Після цього можна починати роботу.

5.3 Повідомлення оператору

Повідомлення оператору здебільшого реалізовані у вигляді помилок валідації на front-end частині(рисунок 5.1).

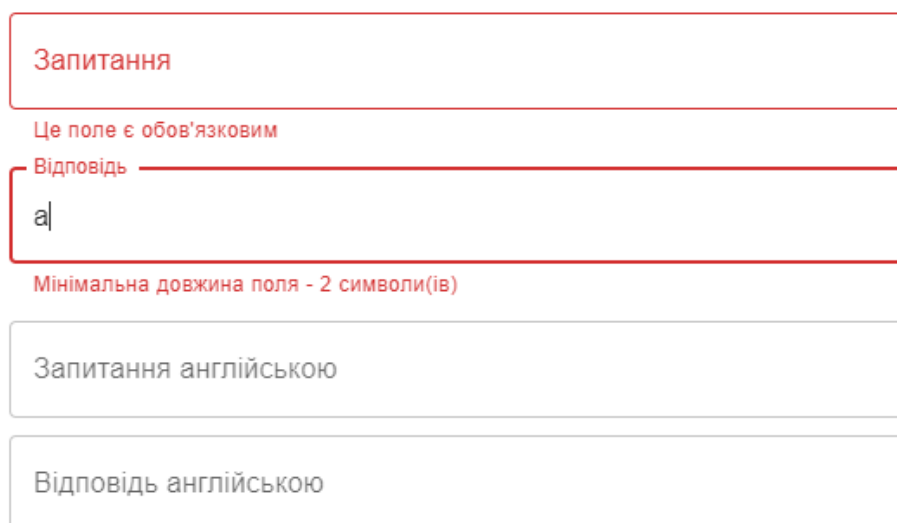


Рисунок 5.1 – Помилка валідації на front-end частині

Окрім повідомлень помилок, оператор отримує повідомлення з необхідністю підтвердження дій, або заповнення полів при створенні, змінінні або видаленні інформації у панелі адміністратора(рисунок 5.2).

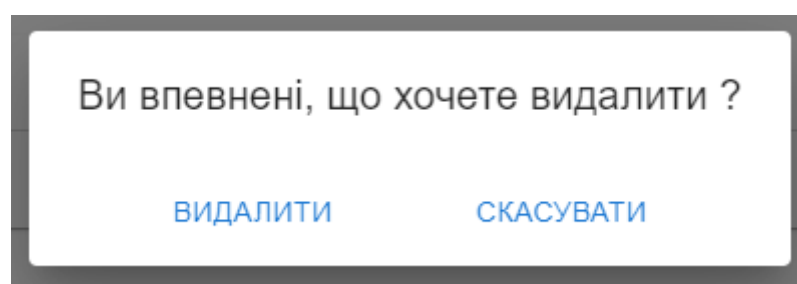


Рисунок 5.2 – Повідомлення про видалення

5.4 Користувачі

На сайті є 2 типи користувачів, перший – це звичайний користувач, йому для доступу на сайт не потрібно нічого, достатньо просто зайти та почати користуватися. Звичайному користувачу доступні наступні сторінки:

- Головна(рисунок 5.3);
- Для вступника(рисунок 5.4);
- Працевлаштування(рисунок 5.5);

- FAQ(рисунок 5.6).



Кафедра програмних засобів
Рік початку діяльності: 2002
Керівництво
Завідувач кафедри: д.т.н., професор Субботін Сергій Олександрович
Перелік викладачів



Під час навчання на кафедрі студенти практикують та вивчають різноманітні мови програмування та проектування програмного забезпечення, такі як C/C++/C#, Java, Python, PHP, PERL, JavaScript, Ruby, Matlab, R, SQL, Assembler.

Окрім того студенти навчаються використовувати ОС Windows, Linux, програми MS Office, Statistica, Mathcad, StatGraphics Plus, Adobe Photoshop, CorelDraw, продукти IBM, WEKA, Autodesk, AutoCAD, SolidEdge, Altium Designer, PTC Creo, Atmel Studio, 1C, SVN, Git, технології React, Vue, Angular, Node.js, ASP.net, Flutter, Laravel.

Також буде можливість отримати навички програмування сучасних мікроконтролерів (платформи Arduino, Raspberry Pi) та розроблення мобільних додатків для Android та iOS.

Спеціальності

121 – «Інженерія програмного забезпечення», освітні програми:
«Інженерія програмного забезпечення»

Підготовка фахівців, робота яких пов'язана з проектуванням, розробленням та супроводженням

122 – «Комп'ютерні науки», освітні програми:
«Комп'ютерні науки»
«Системи штучного інтелекту»

Підготовка фахівців з проектування й реалізації комп'ютерних програм, оптимізації, модернізації та

Рисунок 5.3 – Головна сторінка

Сертифікати ЗНО для вступу бакалаврам

Основні предмети: Українська мова, Математика
Предмети на вибір: Історія України | іноземна мова | біологія | географія | фізика | хімія

Фізика 23 травня (понеділок)	Хімія 26 травня (четвер)	Українська мова 31 травня (вівторок)
Українська мова та література 31 травня (вівторок)	Математика 03 червня (п'ятниця)	Іспанська мова 06 червня (понеділок)
Німецька мова 06 червня (понеділок)	Французька мова 06 червня (понеділок)	Англійська мова 07 червня (вівторок)
Історія України 10 червня (п'ятниця)	Географія 14 червня (вівторок)	Біологія 17 червня (п'ятниця)

Додаткова сесія

Додаткова сесія ЗНО для учасників, які не змогли взяти участь у основній сесії тестування з поважних причин, пройде у період з 23 червня до 15 липня 2022 року

[ЗАРЕЄСТРУВАТИСЯ НА ЗНО](#)

Документи необхідні для вступу до університету:

Документ, що посвідчує особу;
Військовий квиток або посвідчення про приписку – для військовозобов'язаних
Документ державного зразка (оригінал) про раніше здобутий освітній (освітньо-кваліфікаційний) рівень, на основі якого здійснюється вступ, і додаток до нього;
Сертифікат (сертифікати) зовнішнього незалежного оцінювання (для вступників на основі повної загальної середньої освіти);
Документи, які підтверджують право вступника на участь у конкурсі за результатами вступних іспитів, зарахування за співбесідою, зарахування за особливими умовами вступу, чотири кольорові фотокартки розміром 3 x 4 см.
Оригінали документів при участі в конкурсі на місця, що фінансуються за кошти державного бюджету, вступником подаються лише один раз при виборі місця навчання (виконання вимог до зарахування). Копії документів без пред'явлення оригіналів не розглядаються.

[ЯКЩО З'ЯВИЛИСЯ ПИТАННЯ](#)

[ДЛЯ АБІТУРІЄНТІВ З ПІЛЬГАМИ](#)

Професії за якими випускники зможуть працювати

Програміст-розробник - Знання необхідних технологій та мов програмування, залежно від вакансії - Уміння писати та правильно читати технічні завдання (ТЗ) - Знання англійської мови на технічному рівні - Soft skills: постійний саморозвиток, аналітичний склад розуму, відповідальність, комунікабельність, робота в команді. Середня заробітна плата: 2500\$ Гранична заробітна плата: 5300\$ Кількість вакансій: 2400	Системний адміністратор - Розуміти принципи роботи мережевого обладнання, мережевих протоколів - Знати і вміти налаштовувати операційні системи - Вміти працювати з доменними службами, поштовими службами - Володіти широким колом знань різних систем і технологій Середня заробітна плата: 700\$ Гранична заробітна плата: 4000\$ Кількість вакансій: 620
Системний аналітик - Знання персонального комп'ютера та пакетів Microsoft Office, SQL, XML та баз даних - Організація управління проектами з розробки та впровадження - Робота з Інтернет-технологіями та системами аналізу даних - Знання сучасних мов програмування Середня заробітна плата: 1000\$ Гранична заробітна плата: 5000\$ Кількість вакансій: 45	Бізнес-аналітик - Комунікативні навички - Увага до деталей - Уміння вирішувати проблеми - Готовність ухвалювати рішення Середня заробітна плата: 1000\$ Гранична заробітна плата: 6000\$ Кількість вакансій: 236
Графічний дизайнер	Розробник ігор

Рисунок 5.4 – Сторінка для вступника

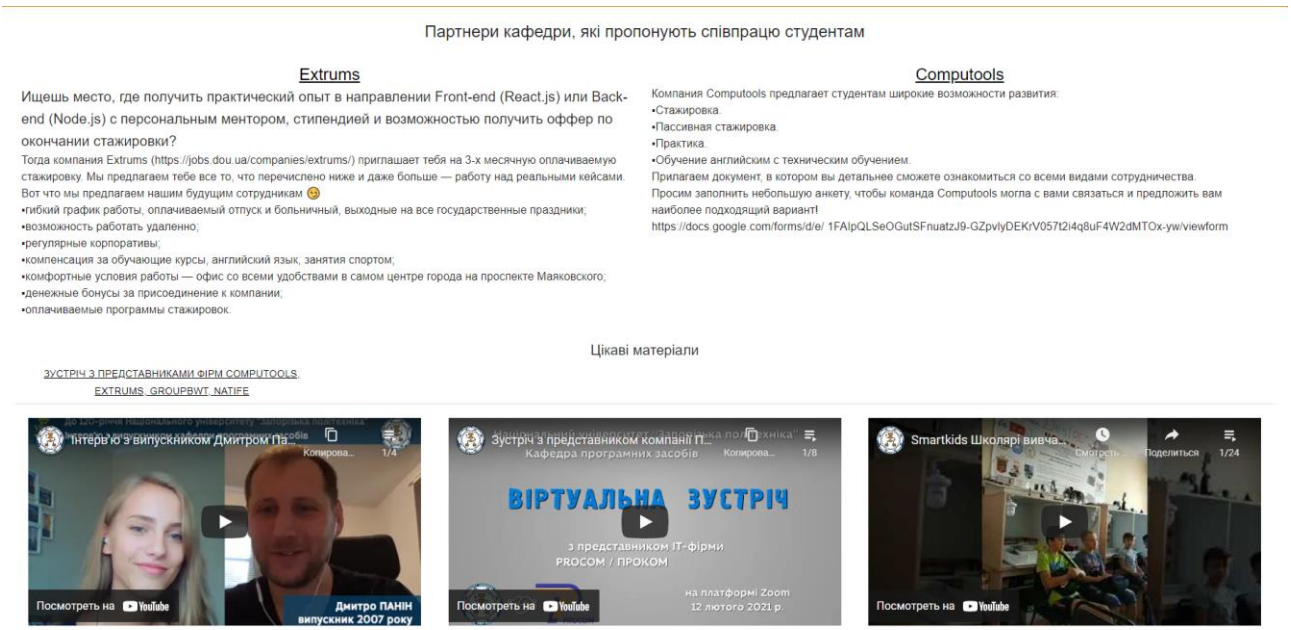


Рисунок 5.5 – Сторінка працевлаштування

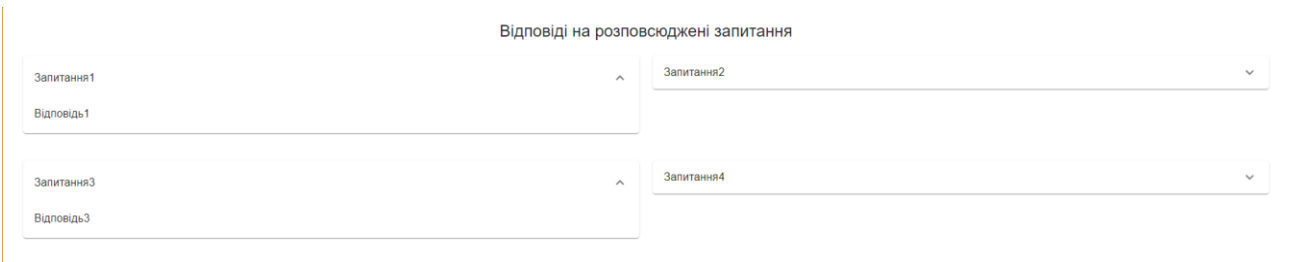


Рисунок 5.6 – Сторінка FAQ

Другий тип користувача – це адміністратор, у цілому це той же звичайний користувач. Відрізняються вони тим, що адміністратор знає пароль від панелі адміністратора та має до неї доступ. Тому адміністратору доступні ті самі сторінки, що й звичайному користувачі та сторінка адміністратора(рисунок 5.7).

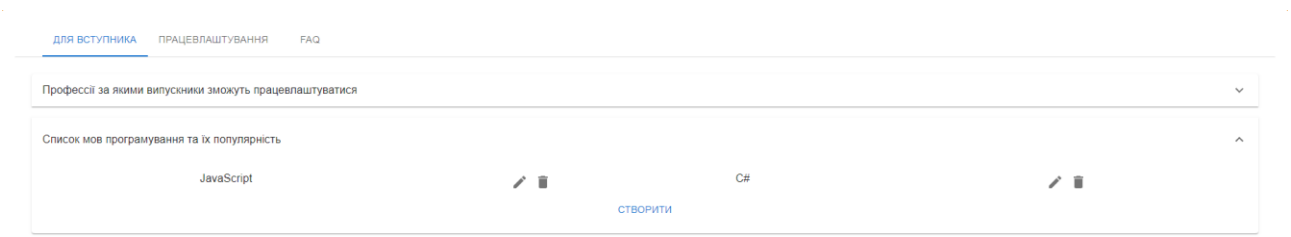


Рисунок 5.7 – Сторінка панелі адміністратора

5.5 Панель адміністратора

Після того, як користувач отримав доступ до панелі адміністратора, йому відкривається можливість змінювати динамічну інформацію сайту. До такої інформації належить:

- професії, за якими випускники зможуть працевлаштуватися;
- список мов програмування та їх популярність;
- партнери кафедри;
- цікаві матеріали;
- відповіді на розповсюджені запитання.

У користувача буде можливість додавати(рисунок 5.8), змінювати(рисунок 5.9) або видаляти(рисунок 5.10) записи за категоріями описаними вище.

Створити новий об'єкт professionList

Назва	test
Назва англійською	test
Вимоги	test
Вимоги англійською	test
Посилання	test
Середня заробітна плата	123
Максимальна заробітна плата	123
Кількість вакансій	123

СТВОРИТИ
СКАСУВАТИ

Рисунок 5.8 – Створення нового запису

Оновити об'єкт professionList

Назва	Програміст-розробник
Назва англійською	Developer
Вимоги	Знання необхідних технологій та мов програмування, залежно від вакансії Уміння писати та правильно читати технічне завдання (ТЗ) Знання англійської мови на технічному рівні Soft skills: постійний саморозвиток, аналітичний склад розуму, відповідальність, комунікабельність, робота в команді.
Вимоги англійською	Knowledge of required technologies and programming languages, depending on the vacancy Ability to write and correctly read the technical task (TOR) Knowledge of English at the technical level Soft skills: constant self-development, analytical mind, responsibility, sociability, teamwork.
Посилання	https://www.work.ua/jobs-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BD%D0%B8%D0%BA/
Середня заробітня плата	2500\$
Максимальна заробітня плата	5300\$
Кількість вакансій	2400

Рисунок 5.9 – Змінення запису

Ви впевнені, що хочете видалити ?

ВИДАЛИТИ
СКАСУВАТИ

Рисунок 5.10 – Видалення запису

ВИСНОВКИ

У результаті виконання дипломної роботи було проаналізовано об'єкт та предмет дослідження, а саме проведення інформаційної компанії в інтернеті для залучення абітурієнтів для вступу до закладу вищої освіти та розробка вебсайтів для залучення абітурієнтів. Було проаналізовано існуючі рішення, якими стали сайти Національного університету “Запорізька політехніка”, а саме основний сайт університету, сайт приймальної комісії та сайт кафедри програмних засобів. Було проаналізовано популярні мови програмування для розробки сайтів та обрано мову для розробки проєкту, нею стала мова програмування JavaScript. Також було проаналізовано різні СКБД та обрано ту, яка підходила більше, а саме MongoDB. Окрім того було описано бібліотеки та фреймворки, які використовувались під час розробки, для чого вони використовувалися у проєкті та приклади їх використання. Було наведено схеми бази даних з назвою поля, його значенням за замовчуванням, описом поля та необхідністю його заповнення. Було описано структуру проєкту з поясненням елементів структури, їх змістом та метою існування, а також алгоритми, які використовувались під час розробки. Також було описано метод взаємодії з БД програмістом, тобто створення бази та її використання. Окрім вище описаного, було названо два типи користувачів, а саме звичайний та адміністратор, сторінки, до яких вони мають доступ, та їх різницю. На останок було описано функціонал панелі адміністратора та динамічну інформацію, яка змінюється у панелі адміністратора.

Фінальним результатом виконання проєкту є розроблений вебсайт для залучення абітурієнтів, який можна використовувати як засіб для реклами університету, кафедри та для залучення нових студентів до вступу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Сайт національного університету “Запорізька політехніка [Електрон. ресурс]. – Режим доступу: ”<https://zp.edu.ua/>
2. Сайт кафедри програмних засобів [Електрон. ресурс]. – Режим доступу: <https://pz.zntu.net/>
3. Сайт приймальної комісії [Електрон. ресурс]. – Режим доступу: <https://pk.zp.edu.ua/>
4. MongoDB [Electronic resource]. – Access mode: <https://cloud.mongodb.com/>
5. React [Electronic resource]. – Access mode: <https://reactjs.org/>
6. Redux [Electronic resource]. – Access mode: <https://redux.js.org/>
7. Express [Electronic resource]. – Access mode: <https://expressjs.com/>
8. MUI [Electronic resource]. – Access mode: <https://mui.com/>
9. Joi [Electronic resource]. – Access mode: <https://joi.dev/>
10. Angular [Electronic resource]. – Access mode: <https://angular.io/>
11. Vue [Electronic resource]. – Access mode: <https://vuejs.org/>
12. JS [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
13. TS [Electronic resource]. – Access mode: <https://www.typescriptlang.org/>
14. Mongoose [Electronic resource]. – Access mode: <https://mongoosejs.com/>
15. WebStorm [Electronic resource]. – Access mode: <https://www.jetbrains.com/webstorm/>
16. Crockford, D. JavaScript: The good parts
17. Haverbeke, M. Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming

ДОДАТОК А
ТЕКСТ ПРОГРАМИ

Back-end

Файл index.js;

```

const express = require('express');
const                                programingLanguagesRouter
require('./modules/programingLanguages/programingLanguages.router');
const professionListRouter = require('./modules/professionList/professionList.router');
const                                partnersEmploymentInfoRouter
require('./modules/partnersEmploymentInfo/partnersEmploymentInfo.router');
const faqListRouter = require('./modules/faqList/faqList.router');
const                                employmentMaterialsRouter
require('./modules/employmentMaterials/employmentMaterials.router');
const cors = require('cors');
const mongoose = require('mongoose');

require('dotenv').config();

mongoose.connect(
  process.env.DATABASE_CONNECTION_STRING,
  {useNewUrlParser: true,
   useUnifiedTopology: true,
});

const app = express();

app.use(express.json());
app.use(cors());

app.use((req, res, next) => {
  console.log(req.body, req.url, req.method);
  if(req.url === '/test') next();
  const authToken = req.headers.authorization;
  if ((!authToken || authToken !== process.env.AUTH_TOKEN) && req.method !== 'OPTIONS')
    throw new Error(JSON.stringify({ message: 'Not valid auth token', status: 'Forbidden', code: 403
})));

  next();
});

app.use('/programmingLanguages', programingLanguagesRouter);
app.use('/professionList', professionListRouter);
app.use('/partnersEmploymentInfo', partnersEmploymentInfoRouter);
app.use('/faqList', faqListRouter);
app.use('/employmentMaterials', employmentMaterialsRouter);

app.use(function (err, req, res, next) {
  console.log(err);
  let errorContent;
  try {
    errorContent = JSON.parse(err.message);
  }
  catch (e) {
    const isJoiError = !!err?.error?.details;
    const joiMessage = isJoiError ?
      err.error.details.map(detail => detail.message,) : null;

    errorContent = { message: joiMessage || err.message };
  }
  res.status(500).send({ status: 'Internal Server Error', ...errorContent });

  next();
});

app.listen(process.env.PORT || 3000, () => console.log(`Server started at port ${process.env.PORT ||
3000}`));

```

Файл employmentMaterials.model.js;

```

const {Schema, model} = require('mongoose');

const schema = new Schema({
  createdAt: {
    type: Date,
    require: true,

```

```

    default: new Date(Date.now()),
  },
  updatedAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  deletedAt: {
    type: Date,
    default: null,
  },
  _id: {
    type: String,
    require: true
  },
  info: {
    type: String,
    require: true
  },
  infoEn: {
    type: String,
  },
  link: {
    type: String,
    require: true
  },
});

```

```
module.exports = model('employmentmaterials', schema);
```

Файл faqList.model.js;

```
const {Schema, model} = require('mongoose');
```

```

const schema = new Schema({
  createdAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  updatedAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  deletedAt: {
    type: Date,
    default: null,
  },
  _id: {
    type: String,
    primaryKey: true,
    require: true
  },
  question: {
    type: String,
    require: true
  },
  questionEn: {
    type: String,
  },
  answer: {
    type: String,
    require: true
  },
  answerEn: {
    type: String,
  },
});

```

```
module.exports = model('faqlist', schema);
```

Файл partnersEmploymentInfo.model.js;

```
const {Schema, model} = require('mongoose');
```

```

const schema = new Schema({
  createdAt: {

```

```

    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  updatedAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  deletedAt: {
    type: Date,
    default: null,
  },
  _id: {
    type: String,
    require: true,
  },
  name: {
    type: String,
    require: true,
  },
  link: {
    type: String,
  },
  headerText: {
    type: Array,
    require: true,
  },
  mainText: {
    type: Array,
    require: true,
  },
  headerTextEn: {
    type: Array,
  },
  mainTextEn: {
    type: Array,
  },
});

module.exports = model('partnersemploymentinfo', schema);

```

Файл professionList.model.js;

```

const {Schema, model} = require('mongoose');

const schema = new Schema({
  createdAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  updatedAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  deletedAt: {
    type: Date,
    default: null,
  },
  _id: {
    type: String,
    require: true
  },
  name: {
    type: String,
    require: true
  },
  nameEn: {
    type: String,
  },
  requirements: {
    type: Array,
    require: true
  },
  requirementsEn: {

```

```

    type: Array,
  },
  link: {
    type: String,
    require: true
  },
  averagePayment: {
    type: String,
    require: true
  },
  maxPayment: {
    type: String,
    require: true
  },
  vacanciesNumber: {
    type: Number,
    require: true
  },
}, {
  freezeTableName: true,
  tableName: 'professionlist'
});

```

```
module.exports = model('professionlist', schema);
```

Файл `programingLanguages.model.js`;

```
const {Schema, model} = require('mongoose');
```

```

const schema = new Schema({
  createdAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  updatedAt: {
    type: Date,
    require: true,
    default: new Date(Date.now()),
  },
  deletedAt: {
    type: Date,
    default: null,
  },
  _id: {
    type: String,
    require: true
  },
  name: {
    type: String,
    require: true,
    unique: true,
  },
  writeNowPerc: {
    type: Number,
    require: true,
  },
  changes: {
    type: Number,
    require: true
  },
  writeNow: {
    type: Number,
    require: true
  },
  useSec: {
    type: Number,
    require: true
  },
  usePrim: {
    type: Number,
    require: true
  },
  likeIndex: {
    type: Number,
    require: true
  },
},

```

```
});
```

```
module.exports = model('programminglanguages', schema);
```

Файл employmentMaterials.controller.js;

```
const employmentMaterialsService = require('./employmentMaterials.service');
```

```
const uuidv4 = require('uuid').v4;
```

```
const employmentMaterialsController = {
  findOne: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await employmentMaterialsService.findOne({ id }));
    } catch (e) { next(e); }
  },
  findAll: async (req, res, next) => {
    try {
      res.json(await employmentMaterialsService.findAll());
    } catch (e) { next(e); }
  },
  create: async (req, res, next) => {
    try {
      const entity = req.body;
      entity._id = uuidv4();
      res.json(await employmentMaterialsService.create(entity));
    } catch (e) { next(e); }
  },
  update: async (req, res, next) => {
    try {
      const id = req.params.id;
      const newEntity = req.body;
      res.json(await employmentMaterialsService.update(id, newEntity));
    } catch (e) { next(e); }
  },
  delete: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await employmentMaterialsService.delete(id));
    } catch (e) { next(e); }
  },
};
```

```
module.exports = employmentMaterialsController;
```

Файл employmentMaterials.router.js;

```
const controller = require('./employmentMaterials.controller');
const express = require('express');
const { createValidator } = require('express-joi-validation');
const {
  employmentMaterialsCreateSchema,
  employmentMaterialsUpdateSchema
} = require('./employmentMaterials.validate.schemas');

const router = express.Router();
const validator = createValidator();

router.get('/:id', controller.findOne);
router.get('/', controller.findAll);
router.post('/', validator.body(employmentMaterialsCreateSchema, {passError: true}),
controller.create);
router.put('/:id', validator.body(employmentMaterialsUpdateSchema, {passError: true}),
controller.update);
router.delete('/:id', controller.delete);
```

```
module.exports = router;
```

Файл employmentMaterials.service.js;

```
const EmploymentMaterials = require('../../models/employmentMaterials.model');
```

```
const employmentMaterialsService = {
  findOne: async (criteria) => {
    const employmentMaterial = await EmploymentMaterials.findOne({ ...criteria, deletedAt: null });
    if (!employmentMaterial) throw new Error(JSON.stringify(
```

```

    {
      message: 'Not found employment material',
      status: 'Not Found',
      code: 404
    }
  ));

  return employmentMaterial;
},
findAll: async () => {
  return EmploymentMaterials.find({ deletedAt: null });
},
create: async (entity) => {
  return EmploymentMaterials.create(entity);
},
update: async (_id, newEntity) => {
  return EmploymentMaterials.updateOne({ _id }, newEntity);
},
delete: async (_id) => {
  return EmploymentMaterials.updateOne({ _id }, { deletedAt: new Date(Date.now()) });
},
};

```

```
module.exports = employmentMaterialsService;
```

Файл employmentMaterials.validate.schemas.js;

```

const Joi = require('joi');

module.exports = {
  employmentMaterialsCreateSchema: Joi.object({
    info: Joi.string().required(),
    infoEn: Joi.string().optional(),
    link: Joi.string().required(),

  }).options({ abortEarly: false }),
  employmentMaterialsUpdateSchema: Joi.object({
    info: Joi.string().optional(),
    infoEn: Joi.string().optional(),
    link: Joi.string().optional(),
  }).options({ abortEarly: false }),
};

```

Файл faqList.controller.js;

```

const faqListService = require('./faqList.service');

const uuidv4 = require('uuid').v4;

const faqListController = {
  findOne: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await faqListService.findOne({ id }));
    } catch (e) { next(e); }
  },
  findAll: async (req, res, next) => {
    try {
      res.json(await faqListService.findAll());
    } catch (e) { next(e); }
  },
  create: async (req, res, next) => {
    try {
      const entity = req.body;
      entity._id = uuidv4();
      res.json(await faqListService.create(entity));
    } catch (e) { next(e); }
  },
  update: async (req, res, next) => {
    try {
      const id = req.params.id;
      const newEntity = req.body;
      res.json(await faqListService.update(id, newEntity));
    } catch (e) { next(e); }
  },
  delete: async (req, res, next) => {
    try {
      const id = req.params.id;

```

```

        res.json(await faqListService.delete(id));
    } catch (e) { next(e); }
  },
};

```

```
module.exports = faqListController;
```

Файл faqList.router.js;

```

const controller = require('./faqList.controller');
const express = require('express');
const { createValidator } = require('express-joi-validation');
const {
  faqListCreateSchema,
  faqListUpdateSchema,
} = require('./faqList.validate.schemas');

const router = express.Router();
const validator = createValidator();

router.get('/:id', controller.findOne);
router.get('/', controller.findAll);
router.post('/', validator.body(faqListCreateSchema, { passError: true }), controller.create);
router.put('/:id', validator.body(faqListUpdateSchema, { passError: true }), controller.update);
router.delete('/:id', controller.delete);

module.exports = router;

```

Файл faqList.service.js;

```

const FaqList = require('../models/faqList.model');

const faqListService = {
  findOne: async (criteria) => {
    const faqList = await FaqList.findOne({ ...criteria, deletedAt: null });

    if (!faqList) throw new Error(JSON.stringify(
      {
        message: 'Not found faq list',
        status: 'Not Found',
        code: 404
      }
    ));

    return faqList;
  },
  findAll: async () => {
    return FaqList.find({ deletedAt: null });
  },
  create: async (entity) => {
    return FaqList.create(entity);
  },
  update: async (_id, newEntity) => {
    return FaqList.updateOne({ _id }, newEntity);
  },
  delete: async (_id) => {
    return FaqList.updateOne({ _id }, { deletedAt: new Date(Date.now()) });
  },
};

```

```
module.exports = faqListService;
```

Файл faqList.validate.schemas.js;

```

const Joi = require('joi');

module.exports = {
  faqListCreateSchema: Joi.object({
    question: Joi.string().required(),
    questionEn: Joi.string().optional(),
    answer: Joi.string().required(),
    answerEn: Joi.string().optional(),
  }).options({ abortEarly: false }),
  faqListUpdateSchema: Joi.object({
    question: Joi.string().optional(),
    questionEn: Joi.string().optional(),
    answer: Joi.string().optional(),
  })

```

```

    answerEn: Joi.string().optional(),
  }).options({ abortEarly: false }),
};

```

Файл partnersEmploymentInfo.controller.js;

```
const partnersEmploymentInfoService = require('./partnersEmploymentInfo.service');
```

```
const uuidv4 = require('uuid').v4;
```

```
const partnersEmploymentInfoController = {
  findOne: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await partnersEmploymentInfoService.findOne({ id }));
    } catch (e) { next(e); }
  },
  findAll: async (req, res, next) => {
    try {
      res.json(await partnersEmploymentInfoService.findAll());
    } catch (e) { next(e); }
  },
  create: async (req, res, next) => {
    try {
      const entity = req.body;
      entity._id = uuidv4();
      res.json(await partnersEmploymentInfoService.create(entity));
    } catch (e) { next(e); }
  },
  update: async (req, res, next) => {
    try {
      const id = req.params.id;
      const newEntity = req.body;
      res.json(await partnersEmploymentInfoService.update(id, newEntity));
    } catch (e) { next(e); }
  },
  delete: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await partnersEmploymentInfoService.delete(id));
    } catch (e) { next(e); }
  },
};
```

```
module.exports = partnersEmploymentInfoController;
```

Файл partnersEmploymentInfo.router.js;

```
const controller = require('./partnersEmploymentInfo.controller');
const express = require('express');
const { createValidator } = require('express-joi-validation');
const {
  partnersEmploymentInfoCreateSchema,
  partnersEmploymentInfoUpdateSchema
} = require('./partnersEmploymentInfo.validate.schemas');
```

```
const router = express.Router();
const validator = createValidator();
```

```

router.get('/:id', controller.findOne);
router.get('/', controller.findAll);
router.post('/', validator.body(partnersEmploymentInfoCreateSchema, { passError: true }),
controller.create);
router.put('/:id', validator.body(partnersEmploymentInfoUpdateSchema, { passError: true }),
controller.update);
router.delete('/:id', controller.delete);

```

```
module.exports = router;
```

Файл partnersEmploymentInfo.service.js;

```
const PartnersEmploymentInfo = require('../../models/partnersEmploymentInfo.model');
```

```
const partnersEmploymentInfoService = {
  findOne: async (criteria) => {
```

```

    const partnersEmploymentInfo = await PartnersEmploymentInfo.findOne({ ...criteria, deletedAt:
null });

    if (!partnersEmploymentInfo) throw new Error(JSON.stringify(
    {
      message: 'Not found partners employment info',
      status: 'Not Found',
      code: 404
    }
    ));

    return partnersEmploymentInfo;
  },
  findAll: async () => {
    return PartnersEmploymentInfo.find({ deletedAt: null });
  },
  create: async (entity) => {
    return PartnersEmploymentInfo.create(entity);
  },
  update: async (_id, newEntity) => {
    return PartnersEmploymentInfo.updateOne({ _id }, newEntity);
  },
  delete: async (_id) => {
    return PartnersEmploymentInfo.updateOne({ _id }, { deletedAt: new Date(Date.now()) });
  },
};

module.exports = partnersEmploymentInfoService;

```

Файл partnersEmploymentInfo.validate.schemas.js;

```

const Joi = require('joi');

module.exports = {
  partnersEmploymentInfoCreateSchema: Joi.object({
    name: Joi.string().required(),
    link: Joi.string().required(),
    headerText: Joi.array().required(),
    mainText: Joi.array().required(),
    headerTextEn: Joi.array().optional(),
    mainTextEn: Joi.array().optional(),
  }).options({ abortEarly: false }),
  partnersEmploymentInfoUpdateSchema: Joi.object({
    name: Joi.string().optional(),
    link: Joi.string().optional(),
    headerText: Joi.array().optional(),
    mainText: Joi.array().optional(),
    headerTextEn: Joi.array().optional(),
    mainTextEn: Joi.array().optional(),
  }).options({ abortEarly: false }),
};

```

Файл professionList.controller.js;

```

const professionListService = require('./professionList.service');

const uuidv4 = require('uuid').v4;

const professionListController = {
  findOne: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await professionListService.findOne({ id }));
    } catch (e) { next(e); }
  },
  findAll: async (req, res, next) => {
    try {
      res.json(await professionListService.findAll());
    } catch (e) { next(e); }
  },
  create: async (req, res, next) => {
    try {
      const entity = req.body;
      entity._id = uuidv4();
      res.json(await professionListService.create(entity));
    } catch (e) { next(e); }
  },
  update: async (req, res, next) => {

```

```

    try {
      const id = req.params.id;
      const newEntity = req.body;
      res.json(await professionListService.update(id, newEntity));
    } catch (e) { next(e); }
  },
  delete: async (req, res, next) => {
    try {
      const id = req.params.id;
      res.json(await professionListService.delete(id));
    } catch (e) { next(e); }
  },
};

module.exports = professionListController;

```

Файл professionList.router.js;

```

const controller = require('./professionList.controller');
const express = require('express');
const { createValidator } = require('express-joi-validation');
const {
  professionListCreateSchema,
  professionListUpdateSchema
} = require('./professionList.validate.schemas');

const router = express.Router();
const validator = createValidator();

router.get('/:id', controller.findOne);
router.get('/', controller.findAll);
router.post('/', validator.body(professionListCreateSchema, { passError: true }), controller.create);
router.put('/:id', validator.body(professionListUpdateSchema, { passError: true }),
controller.update);
router.delete('/:id', controller.delete);

module.exports = router;

```

Файл professionList.service.js;

```

const ProfessionList = require('../../models/professionList.model');

const professionListService = {
  findOne: async (criteria) => {
    const professionList = await ProfessionList.findOne({ ...criteria, deletedAt: null });

    if (!professionList) throw new Error(JSON.stringify(
      {
        message: 'Not found profession list',
        status: 'Not Found',
        code: 404
      }
    ));

    return professionList;
  },
  findAll: async () => {
    return ProfessionList.find({ deletedAt: null });
  },
  create: async (entity) => {
    return ProfessionList.create(entity);
  },
  update: async (_id, newEntity) => {
    return ProfessionList.updateOne({ _id }, newEntity);
  },
  delete: async (_id) => {
    return ProfessionList.updateOne({ _id }, { deletedAt: new Date(Date.now()) });
  },
};

module.exports = professionListService;

```

Файл professionList.validate.schemas.js;

```

const Joi = require('joi');

module.exports = {
  professionListCreateSchema: Joi.object({

```

```

    name: Joi.string().required(),
    nameEn: Joi.string().optional(),
    requirements: Joi.array().required(),
    requirementsEn: Joi.array().optional(),
    link: Joi.string().required(),
    averagePayment: Joi.string().required(),
    maxPayment: Joi.string().required(),
    vacanciesNumber: Joi.string().required(),
  }).options({ abortEarly: false }),
  professionListUpdateSchema: Joi.object({
    name: Joi.string().optional(),
    nameEn: Joi.string().optional(),
    requirements: Joi.array().optional(),
    requirementsEn: Joi.array().optional(),
    link: Joi.string().optional(),
    averagePayment: Joi.string().optional(),
    maxPayment: Joi.string().optional(),
    vacanciesNumber: Joi.string().optional(),
  }).options({ abortEarly: false }),
};

```

Файл programmingLanguages.controller.js;

```

const programmingLanguagesService = require('./programmingLanguages.service');

const uuidv4 = require('uuid').v4;

const programmingLanguagesController = {
  findOne: async (req, res, next) => {
    try {
      const name = req.params.name;
      res.json(await programmingLanguagesService.findOne({ name }));
    } catch (e) { next(e); }
  },
  findAll: async (req, res, next) => {
    try {
      res.json(await programmingLanguagesService.findAll());
    } catch (e) { next(e); }
  },
  create: async (req, res, next) => {
    try {
      const entity = req.body;
      entity._id = uuidv4();
      res.json(await programmingLanguagesService.create(entity));
    } catch (e) { next(e); }
  },
  update: async (req, res, next) => {
    try {
      const name = req.params.name;
      const newEntity = req.body;
      res.json(await programmingLanguagesService.update(name, newEntity));
    } catch (e) { next(e); }
  },
  delete: async (req, res, next) => {
    try {
      const name = req.params.name;
      res.json(await programmingLanguagesService.delete(name));
    } catch (e) { next(e); }
  },
};

```

```
module.exports = programmingLanguagesController;
```

Файл programmingLanguages.router.js;

```

const controller = require('./programmingLanguages.controller');
const express = require('express');
const { createValidator } = require('express-joi-validation');
const {
  programmingLanguagesCreateSchema,
  programmingLanguagesUpdateSchema
} = require('./programmingLanguages.validate.schemas');

const router = express.Router();
const validator = createValidator();

router.get('/:name', controller.findOne);
router.get('/', controller.findAll);

```

```

    router.post('/', validator.body(programingLanguagesCreateSchema, { passError: true })),
    controller.create);
    router.put('/:name', validator.body(programingLanguagesUpdateSchema, { passError: true })),
    controller.update);
    router.delete('/:name', controller.delete);

```

```
module.exports = router;
```

Файл programingLanguages.service.js;

```
const ProgramingLanguages = require('../../models/programingLanguages.model');
```

```

const programingLanguagesService = {
  findOne: async (criteria) => {
    const programingLanguages = await ProgramingLanguages.findOne({ ...criteria, deletedAt: null });

    if (!programingLanguages) throw new Error(JSON.stringify({
      message: 'Not found programing languages',
      status: 'Not Found',
      code: 404
    }));

    return programingLanguages;
  },
  findAll: async () => {
    return ProgramingLanguages.find({ deletedAt: null });
  },
  create: async (entity) => {
    return ProgramingLanguages.create(entity);
  },
  update: async (_id, newEntity) => {
    return ProgramingLanguages.updateOne({ _id }, newEntity);
  },
  delete: async (_id) => {
    return ProgramingLanguages.updateOne({ _id }, { deletedAt: new Date(Date.now()) });
  },
};

```

```
module.exports = programingLanguagesService;
```

Файл programingLanguages.validate.schemas.js;

```
const Joi = require('joi');
```

```

module.exports = {
  programingLanguagesCreateSchema: Joi.object({
    name: Joi.string().required().max(36),
    writeNowPerc: Joi.string().required(),
    changes: Joi.string().required(),
    writeNow: Joi.string().required(),
    useSec: Joi.string().required(),
    usePrim: Joi.string().required(),
    likeIndex: Joi.string().required(),
  }).options({ abortEarly: false }),
  programingLanguagesUpdateSchema: Joi.object({
    name: Joi.string().optional().max(36),
    writeNowPerc: Joi.string().optional(),
    changes: Joi.string().optional(),
    writeNow: Joi.string().optional(),
    useSec: Joi.string().optional(),
    usePrim: Joi.string().optional(),
    likeIndex: Joi.string().optional(),
  }).options({ abortEarly: false }),
};

```

Front-end

Файл index.js;

```

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './Redux/reduxStore';

```

```
ReactDOM.render(
  <React.StrictMode>
    <BrowserRouter>
      <Provider store={store}>
        <App />
      </Provider>
    </BrowserRouter>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Файл App.js;

```
import React from 'react';
import './App.css';
import RouteController from './Components/RouteController/RouteController';
import FooterContainer from './Containers/FooterContainer';
import HeaderContainer from './Containers/HeaderContainer';
import { Grid } from '@mui/material';
const App = () => <Grid container className='app'>
  <Grid item xs={12}><HeaderContainer /></Grid>
  <Grid item xs={12}>
    <Grid container className='app' alignItems='center'>
      <div className='appWrap'>
        <RouteController />
      </div>
    </Grid>
  </Grid>
  <Grid item xs={12}>
    <Grid container className='app' alignItems='flex-end'>
      <div className='appWrap'>
        <FooterContainer />
      </div>
    </Grid>
  </Grid>
</Grid>;
```

```
export default App;
```

Файл constants.ua.js;

```
/* eslint-disable max-len */
export const translateUkr = {
  'DEPARTMENT-NAME': 'Кафедра програмних засобів',
  'ACTIVITY-START-YEAR': 'Рік початку діяльності',
  'ADMINISTRATION': 'Керівництво',
  'ADMINISTRATION-OF-KAFEDRA': 'Завідувач кафедри: ',
  'HEAD-OF-THE-DEPARTMENT': 'д.т.н., професор Субботін Сергій Олександрович',
  'TEACHERS-LINK-TITLE': 'Перелік викладачів',
  'DESCRIPTION-1': 'Під час навчання на кафедрі студенти практикують та вивчають різноманітні мови програмування та проектування програмного забезпечення, такі як C/C++/C#, Java, Python, PHP, PERL, JavaScript, Ruby, Matlab, R, SQL, Assembler',
  'DESCRIPTION-2': 'Окрім того студенти навчаються використовувати ОС Windows, Linux, програми MS Office, Statistica, Mathcad, StatGraphics Plus, Adobe Photoshop, CorelDraw, продукти IBM, WEKA, Autodesk, AutoCAD, SolidEdge, Altium Designer, PTC Creo, Atmel Studio, 1C, SVN, Git, технології React, Vue, Angular, Node.js, ASP.net, Flutter, Laravel',
  'DESCRIPTION-3': 'Також буде можливість отримати навички програмування сучасних мікроконтролерів (платформи Arduino, Raspberry Pi) та розроблення мобільних додатків для Android та iOS.',
  'SPECIALITIES': 'Спеціальності',
  'SPECIALITY-1': '121 – «Інженерія програмного забезпечення», освітні програми',
  'SPECIALITY-1-STUDYPROGRAM': '«Інженерія програмного забезпечення»',
  'SPECIALITY-1-DESCRIPTION': 'Підготовка фахівців, робота яких пов'язана з проектуванням, розробленням та супроводженням програмного забезпечення різноманітного призначення',
  'SPECIALITY-2': '122 – «Комп'ютерні науки», освітні програми',
  'SPECIALITY-2-STUDYPROGRAM-1': '«Комп'ютерні науки»',
  'SPECIALITY-2-STUDYPROGRAM-2': '«Системи штучного інтелекту»',
  'SPECIALITY-2-DESCRIPTION': 'Підготовка фахівців з проектування й реалізації комп'ютерних програм, оптимізації, модернізації та розвитку наявного ПЗ, застосування інформаційних технологій у різних сферах',
  'HEADER-FAQ': 'FAQ',
  'HEADER-SPECIALITIES': 'Спеціальності',
  'HEADER-MAIN': 'Головна',
  'HEADER-STUDY-ABROAD': 'Навчання за кордоном',
  'HEADER-FOR-ABITURIENT': 'Для вступника',
  'HEADER-WORK': 'Працевлаштування',
  'FOOTER-ADRESS-TITLE': 'Адреса:',
  'FOOTER-PHONE-TITLE': 'Телефон:',
  'FOOTER-EMAIL-TITLE': 'Електронна пошта:'
```

```

'FOOTER-ADRESS': 'вул. Жуковського 64, м. Запоріжжя, Україна, 69063',
'FOOTER-PHONE': '+380(61)7698267',
'FOOTER-EMAIL': 'kafedra_pz@zntu.edu.ua',
'FOOTER-LANGUAGE-CHANGE': 'Змінити мову:',
'FOOTER-LANGUAGE-TITLE': 'Мова',
'STUDY-PRICE-TITLE': 'Вартість навчання',
'STUDY-PRICE-1': '7600 грн./семестр + індексація',
'STUDY-PRICE-2': '7000 грн./семестр + індексація',
'FREE-PLACES-TITLE': 'Кількість бюджетних місць',
'FREE-PLACES-1': '30 для молодших спеціалістів та 31 для інших',
'FREE-PLACES-2': '20 для молодших спеціалістів та 26 для інших',
'PARTNERS-TITLE': 'Партнери кафедри',
'PARTNERS': 'SoftServe, Computools, Freshcode, Incode Group, Infocom LTD, Light-IT, Vimeo, Absolute
Web, DoneIT, Drum\N\Code, Ukrsoft Group, GroupBWT, Kozak Group, OpenGeeksLab, S-PRO, CHI Software, Extrums,
CUDEV, GBKSOFT, Koitechs, Logix ITS, Lumighost, Revolut LTD, Rezer Uran, Urich, ПАТ "Карлсберг Україна",
Золотий вік, ТОВ "Свізна Країна", АТ "Залк", АК КБ "ПриватБанк", ДТЕК Сервіс, ПАТ "Запоріжсталь", ПАТ "Мотор
Січ", Fozzy-group',
'ENTRANT-EXAM-CERTIFICATES': 'Сертифікати ЗНО для вступу бакалаврам',
'ENTRANT-EXAM-MAIN-SUBJECTS': 'Основні предмети: Українська мова, Математика',
'ENTRANT-EXAM-ADDITIONAL-SUBJECTS': 'Предмети на вибір: Історія України | іноземна мова | біологія
| географія | фізика | хімія',
'ENTRANT-EXAM-SCHEDULE-TITLE': 'Графік проведення ЗНО',
'ENTRANT-EXAM-SCHEDULE': 'Фізика 23 травня (понеділок), Хімія 26 травня (четвер), Українська мова 31
травня (вівторок), Українська мова та література 31 травня (вівторок), Математика 03 червня (п'ятниця), Іспанська
мова 06 червня (понеділок), Німецька мова 06 червня (понеділок), Французька мова 06 червня (понеділок), Англійська
мова 07 червня (вівторок), Історія України 10 червня (п'ятниця), Географія 14 червня (вівторок), Біологія 17 червня
(п'ятниця)',
'ENTRANT-EXAM-EXTRA-SESSION-TITLE': 'Додаткова сесія',
'ENTRANT-EXAM-EXTRA-SESSION': 'Додаткова сесія ЗНО для учасників, які не змогли взяти участь у
основній сесії тестування з поважних причин, пройде у період з 23 червня до 15 липня 2022 року',
'EXAM-REGISTER': 'Зареєструватися на ЗНО',
'UNIVERSITY-ENTRY-DOCUMENT-LIST-TITLE': 'Документи необхідні для вступу до університету:',
'UNIVERSITY-ENTRY-DOCUMENT-LIST-ARRAY': [
  'Документ, що посвідчує особу;',
  'Військовий квиток або посвідчення про приписку – для військовозобов'язаних',
  'Документ державного зразка (оригінал) про раніше здобутий освітній (освітньо-кваліфікаційний)
рівень, на основі якого здійснюється вступ, і додаток до нього;',
  'Сертифікат (сертифікати) зовнішнього незалежного оцінювання (для вступників на основі повної
загальної середньої освіти);',
  'Документи, які підтверджують право вступника на участь у конкурсі за результатами вступних
іспитів, зарахування за співбесідою, зарахування за особливими умовами вступу; чотири кольорові фотокартки
розміром 3 x 4 см.',
  'Оригінали документів при участі в конкурсі на місця, що фінансуються за кошти державного бюджету,
вступником подаються лише один раз при виборі місця навчання (виконання вимог до зарахування). Копії документів
без пред'явлення оригіналів не розглядаються.'
],
'UNIVERSITY-ENTRY-BENEFITS': 'Для абітурієнтів з пільгами',
'UNIVERSITY-ENTRY-QUESTIONS': 'Якщо з'явилися питання',
'PROFESSIONS-LIST-TITLE': 'Професії за якими випускники зможуть працевлаштуватися',
'AVERAGE-PAYMENT': 'Середня заробітна плата:',
'MAX-PAYMENT': 'Гранична заробітна плата:',
'VACANCIES-NUMBER': 'Кількість вакансій:',
'EMPLOYMENT-PARTNERS-TITLE': 'Партнери кафедри, які пропонують співпрацю студентам',
'INTERESTING-INFO-TITLE': 'Цікаві матеріали',
'TABLE-ENTRANT-LANGUAGE': 'Мова програмування',
'TABLE-ENTRANT-WRITE-NOW-PERC': 'Пишуть для роботи зараз, %',
'TABLE-ENTRANT-CHANGE': 'Зміни',
'TABLE-ENTRANT-WRITE-NOW': 'Пишуть для роботи зараз',
'TABLE-ENTRANT-USE-SECONDARY': 'Використовують як додаткові',
'TABLE-ENTRANT-USE-PRIMARY': 'Використовують в своїх проектах',
'TABLE-ENTRANT-LIKE-INDEX': 'Індекс вподобання',
'TABLE-ENTRANT-TITLE': 'Список мов програмування та їх популярність',
'FAQ-TITLE': 'Відповіді на розповсюджені запитання',
'DIALOG-NAME-FIELD': 'Назва',
'DIALOG-LINK-FIELD': 'Посилання',
'DIALOG-HEADER-TEXT-FIELD': 'Заголовок',
'DIALOG-MAIN-TEXT-FIELD': 'Основний текст',
'DIALOG-HEADER-TEXT-EN-FIELD': 'Заголовок англійською',
'DIALOG-MAIN-TEXT-EN-FIELD': 'Основний текст англійською',
'DIALOG-INFO-FIELD': 'Інформація',
'DIALOG-INFO-EN-FIELD': 'Інформація англійською',
'DIALOG-QUESTION-FIELD': 'Запитання',
'DIALOG-ANSWER-FIELD': 'Відповідь',
'DIALOG-QUESTION-EN-FIELD': 'Запитання англійською',
'DIALOG-ANSWER-EN-FIELD': 'Відповідь англійською',
'DIALOG-NAME-EN-FIELD': 'Назва англійською',
'DIALOG-REQUIREMENTS-FIELD': 'Вимоги',

```

```

'DIALOG-REQUIREMENTS-EN-FIELD': 'Вимоги англійською',
'DIALOG-AVERAGE-PAYMENT-FIELD': 'Середня заробітна плата',
'DIALOG-MAX-PAYMENT-FIELD': 'Максимальна заробітна плата',
'DIALOG-VACANCIES-NUMBER-FIELD': 'Кількість вакансій',
'DIALOG-WRITE-NOW-PERC-FIELD': 'Пишуть зараз %',
'DIALOG-CHANGE-FIELD': 'Зміна',
'DIALOG-WRITE-NOW-FIELD': 'Пишуть зараз',
'DIALOG-USE-SEC-FIELD': 'Використовують як другу',
'DIALOG-USE-PRIM-FIELD': 'Використовують як основну',
'DIALOG-INDEX-FIELD': 'Індекс оцінки',
'ADMIN-CREATE-NEW': 'Створити новий об\'єкт',
'ADMIN-CREATE-CONFIRM': 'Створити',
'ADMIN-CREATE-CANCEL': 'Скасувати',
'ADMIN-UPDATE': 'Оновити об\'єкт',
'ADMIN-UPDATE-CONFIRM': 'Оновити',
'ADMIN-UPDATE-CANCEL': 'Скасувати',
'ADMIN-DELETE-CONFIRM-TITLE': 'Ви впевнені, що хочете видалити?',
'ADMIN-DELETE-CONFIRM': 'Видалити',
'ADMIN-DELETE-CANCEL': 'Скасувати',
'ADMIN-AUTH-TITLE': 'Введіть авторизаційний ключ для доступу до панелі адміністратора',
'ADMIN-AUTH-CONFIRM': 'Підтвердити',
'ADMIN-CREATE-BUTTON-TEXT': 'Створити',
'ADMIN-ERROR-REQUIRED': 'Це поле є обов\'язковим',
'ADMIN-ERROR-MIN-FIELD-LENGTH': (value) => `Мінімальна довжина поля - ${value} символи(ів)`,
'ADMIN-ERROR-MIN-FIELD-VALUE': (value) => `Мінімальне значення поля - ${value}`,
'ADMIN-ERROR-MAX-FIELD-VALUE': (value) => `Максимальне значення поля - ${value}`,
};

```

Файл constants.eng.js;

```

export const translateEng = {
  'ACTIVITY-START-YEAR': 'Year of the beginning of activity',
  'ADMINISTRATION': 'Leadership',
  'ADMINISTRATION-OF-KAFEDRA': 'Head of the department: ',
  'HEAD-OF-THE-DEPARTMENT': 'd.t.s., Professor Sergey Alexandrovich Subbotin',
  'TEACHERS-LINK-TITLE': 'Teachers list',
  'DESCRIPTION-1': 'While studying at the department, students practice and learn various programming
languages and software design, such as C / C ++ / C #, Java, Python, PHP, PERL, JavaScript, Ruby, Matlab, R,
SQL, Assembler',
  'DESCRIPTION-2': 'In addition, students will learn to use Windows, Linux, MS Office, Statistica,
Mathcad, StatGraphics Plus, Adobe Photoshop, CorelDraw, IBM, WEKA, Autodesk, AutoCAD, SolidEdge, Altium
Designer, PTC Creo, Atmel Studio, 1C, SVN, Git, Technology React, Vue, Angular, Node.js, ASP.net, Flutter,
Laravel',
  'DESCRIPTION-3': 'There will also be an opportunity to gain skills in programming modern
microcontrollers (Arduino platforms, Raspberry Pi) and developing mobile applications for Android and iOS.',
  'SPECIALITIES': 'Specialties',
  'SPECIALITY-1': '121 - «Software Engineering», educational programs',
  'SPECIALITY-1-STUDYPROGRAM': '«Software engineering»',
  'SPECIALITY-1-DESCRIPTION': 'Training of specialists whose work is related to the design,
development and maintenance of software for various purposes',
  'SPECIALITY-2': '122 - «Computer Science», educational programs',
  'SPECIALITY-2-STUDYPROGRAM-1': '«Computer Science»',
  'SPECIALITY-2-STUDYPROGRAM-2': '«Artificial intelligence systems»',
  'SPECIALITY-2-DESCRIPTION': 'Training of specialists in the design and implementation of computer
programs, optimization, modernization and development of existing software, the use of information technology
in various fields',
  'HEADER-FAQ': 'FAQ',
  'HEADER-SPECIALITIES': 'Specialties',
  'HEADER-MAIN': 'Home',
  'HEADER-STUDY-ABROAD': 'Studying abroad',
  'HEADER-FOR-ABITURIENT': 'For the entrant',
  'HEADER-WORK': 'Employment',
  'FOOTER-ADDRESS-TITLE': 'Address:',
  'FOOTER-PHONE-TITLE': 'Phone:',
  'FOOTER-EMAIL-TITLE': 'E-mail:',
  'FOOTER-ADDRESS': 'str. Zhukovsky 64, Zaporozhye, Ukraine, 69063',
  'FOOTER-PHONE': '+380(61)7698267',
  'FOOTER-EMAIL': 'kafedra_pz@zntu.edu.ua',
  'FOOTER-LANGUAGE-CHANGE': 'Change the language:',
  'FOOTER-LANGUAGE-TITLE': 'Language',
  'STUDY-PRICE-TITLE': 'Cost of education',
  'STUDY-PRICE-1': 'UAH 7600 / semester + indexation',
  'STUDY-PRICE-2': 'UAH 7000 / semester + indexation',
  'FREE-PLACES-TITLE': 'Number of state-financed places.',
  'FREE-PLACES-1': '30 for junior specialists and 31 for others',
  'FREE-PLACES-2': '20 for junior specialists and 26 for others',
  'PARTNERS-TITLE': 'Partners of the department',

```

'PARTNERS': 'SoftServe, Computools, Freshcode, Incode Group, Infocom LTD, Light-IT, Vimeo, Absolute Web, DoneIT, Drum\N\Code, Ukrsoft Group, GroupBWT, Kozak Group, OpenGeeksLab, S-PRO, CHI Software, Extrums, CUDEV, GBKSOFT, Koitechs, Logix ITS, Lumighost, Revolut LTD, Rezer Uran, Urich, PJSC "Carlsberg Ukraine", Golden Age, LLC "Silver Country", JSC "ZaK", JSC "ZaK", JSC CB "PrivatBank", DTEK Service, PJSC "Zaporizhstal", PJSC "Motor Sich", Fozzy-group',

'ENTRANT-EXAM-CERTIFICATES': 'EIT certificates for admission to bachelors',

'ENTRANT-EXAM-MAIN-SUBJECTS': 'Main subjects: Ukrainian language, Mathematics',

'ENTRANT-EXAM-ADDITIONAL-SUBJECTS': 'Electives: History of Ukraine foreign language biology | geography physics chemistry',

'ENTRANT-EXAM-SCHEDULE-TITLE': 'EIT evaluation schedule',

'ENTRANT-EXAM-SCHEDULE': 'Physics May 23 (Monday), Chemistry May 26 (Thursday), Ukrainian Language May 31 (Tuesday), Ukrainian Language and Literature May 31 (Tuesday), Mathematics June 3 (Friday), Spanish Language June 06 (Monday), German language June 06 (Monday), French language June 6 (Monday), English language June 7 (Tuesday), History of Ukraine June 10 (Friday), Geography June 14 (Tuesday), Biology June 17 (Friday)',

'ENTRANT-EXAM-EXTRA-SESSION-TITLE': 'Additional session',

'ENTRANT-EXAM-EXTRA-SESSION': 'An additional EIT session for participants who were unable to participate in the main testing session due to valid reasons will be held from June 23 to July 15, 2022.',

'EXAM-REGISTER': 'Register for external evaluation',

'UNIVERSITY-ENTRY-DOCUMENT-LIST-TITLE': 'Documents required for admission to the university:',

'UNIVERSITY-ENTRY-DOCUMENT-LIST-ARRAY': [

'Identity document;',

'Military ID or certificate of enlistment - for conscripts',

'State standard document (original) on the previously obtained educational (educational and qualification) level on the basis of which the admission is made, and the appendix to it;',

'Certificate (certificates) of external independent evaluation (for entrants on the basis of complete general secondary education);',

'Documents confirming the entrant\'s right to participate in the competition based on the results of entrance exams, enrollment by interview, enrollment under special conditions of admission, four color photographs measuring 3 x 4 cm.',

'The originals of documents when participating in the competition for places financed from the state budget, the applicant submits only once when choosing a place of study (fulfillment of enrollment requirements). Copies of documents without presentation of originals are not considered.'

],

'UNIVERSITY-ENTRY-BENEFITS': 'For entrants with benefits',

'UNIVERSITY-ENTRY-QUESTIONS': 'If you have any questions',

'PROFESSIONS-LIST-TITLE': 'Professions for which graduates will be able to find employment',

'AVERAGE-PAYMENT': 'Average salary:',

'MAX-PAYMENT': 'Marginal wages:',

'VACANCIES-NUMBER': 'Number of vacancies:',

'EMPLOYMENT-PARTNERS-TITLE': 'Partners of the department who offer cooperation to students',

'INTERESTING-INFO-TITLE': 'Interesting materials',

'TABLE-ENTRANT-LANGUAGE': 'Programming language',

'TABLE-ENTRANT-WRITE-NOW-PERC': 'Write to work now, %',

'TABLE-ENTRANT-CHANGE': 'Changes',

'TABLE-ENTRANT-WRITE-NOW': 'Write to work now',

'TABLE-ENTRANT-USE-SECONDARY': 'Used as an add-on',

'TABLE-ENTRANT-USE-PRIMARY': 'Use in their projects',

'TABLE-ENTRANT-LIKE-INDEX': 'Favorite Index',

'TABLE-ENTRANT-TITLE': 'List of programming languages and their popularity',

'FAQ-TITLE': 'Answers to Frequently Asked Questions',

'DIALOG-NAME-FIELD': 'Name',

'DIALOG-LINK-FIELD': 'Link',

'DIALOG-HEADER-TEXT-FIELD': 'Header text',

'DIALOG-MAIN-TEXT-FIELD': 'Main text',

'DIALOG-HEADER-TEXT-EN-FIELD': 'Header text on english',

'DIALOG-MAIN-TEXT-EN-FIELD': 'Main text on english',

'DIALOG-INFO-FIELD': 'Information',

'DIALOG-INFO-EN-FIELD': 'Information on english',

'DIALOG-QUESTION-FIELD': 'Question',

'DIALOG-ANSWEAR-FIELD': 'Answear',

'DIALOG-QUESTION-EN-FIELD': 'Question on english',

'DIALOG-ANSWEAR-EN-FIELD': 'Answear on english',

'DIALOG-NAME-EN-FIELD': 'Name on english',

'DIALOG-REQUIREMENTS-FIELD': 'Requirements',

'DIALOG-REQUIREMENTS-EN-FIELD': 'Requirements on english',

'DIALOG-AVERAGE-PAYMENT-FIELD': 'Average payment',

'DIALOG-MAX-PAYMENT-FIELD': 'Max payment',

'DIALOG-VACANCIES-NUMBER-FIELD': 'Number of the vacancies',

'DIALOG-WRITE-NOW-PERC-FIELD': 'Write now, %',

'DIALOG-CHANGE-FIELD': 'Change',

'DIALOG-WRITE-NOW-FIELD': 'Write now',

'DIALOG-USE-SEC-FIELD': 'Use as secondary',

'DIALOG-USE-PRIM-FIELD': 'Use as primary',

'DIALOG-INDEX-FIELD': 'Like index',

'ADMIN-CREATE-NEW': 'Create new',

'ADMIN-CREATE-CONFIRM': 'Create',

'ADMIN-CREATE-CANCEL': 'Cancel',

```

'ADMIN-UPDATE': 'Update object',
'ADMIN-UPDATE-CONFIRM': 'Update',
'ADMIN-UPDATE-CANCEL': 'Cancel',
'ADMIN-DELETE-CONFIRM-TITLE': 'Are you sure you want to delete ?',
'ADMIN-DELETE-CONFIRM': 'Delete',
'ADMIN-DELETE-CANCEL': 'Cancel',
'ADMIN-AUTH-TITLE': 'Please enter the authorisation code to get access to admin panel',
'ADMIN-AUTH-CONFIRM': 'Confirm',
'ADMIN-CREATE-BUTTON-TEXT': 'Create',
'ADMIN-ERROR-REQUIRED': 'This field is required',
'ADMIN-ERROR-MIN-FIELD-LENGTH': (value) => `Min field length - ${value} symbols`,
'ADMIN-ERROR-MIN-FIELD-VALUE': (value) => `Min field value - ${value}`,
'ADMIN-ERROR-MAX-FIELD-VALUE': (value) => `Max field value - ${value}`,
};

```

Файл Api.js;

```

import * as axios from 'axios';
import env from 'react-dotenv';

const instance = axios.create({
  baseURL: env.BACKEND_URL,
  headers: {
    'Authorization': env.AUTH,
  }
});

export const getAllData = async () => {
  const programmingLanguages = (await instance.get('programmingLanguages')).data;
  const faqList = (await instance.get('faqList')).data;
  const professionList = (await instance.get('professionList')).data;
  const partnersEmploymentInfo = (await instance.get('partnersEmploymentInfo')).data;
  const employmentMaterials = (await instance.get('employmentMaterials')).data;
  return { programmingLanguages, faqList, professionList, partnersEmploymentInfo, employmentMaterials };
};

export const createNewEntity = async (entity, entityName) => {
  const newEntity = (await instance.post(entityName, entity)).data;

  return newEntity;
};

export const updateEntity = async (_id, entity, entityName) => {
  const updatedEntity = (await instance.put(`${entityName}/${_id}`, entity)).data;
  return { ...updatedEntity, entity: { _id, ...entity, }, entityName };
};

export const deleteEntity = async (id, entityName) => {
  const deletedEntity = (await instance.delete(`${entityName}/${id}`)).data;

  return deletedEntity;
};

```

Файл AdminPanel.js;

```

import {
  Accordion,
  AccordionDetails,
  AccordionSummary,
  Box,
  Paper,
  Tab,
  Tabs,
  Typography,
  Grid,
  IconButton,
  Dialog,
  DialogTitle,
  DialogActions,
  DialogContent,
  Button,
  TextField
} from '@mui/material';
import React, { useEffect, useState } from 'react';
import './AdminPanel.css';
import { translateUkr } from '../constants.ua';
import { translateEng } from '../constants.eng';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';

```

```

import EditIcon from '@mui/icons-material/Edit';
import DeleteIcon from '@mui/icons-material/Delete';
import env from 'react-dotenv';
import { ValidatorForm } from 'react-material-ui-form-validator';
import { TextValidator } from 'react-material-ui-form-validator';

const AdminPanel = (props) => {
  useEffect(() => {
    if (props.isAuth)
      props.getAllData(props.setAllData);
  }, [props.isAuth]);

  const phrases = props.language === 1 ? translateUkr : translateEng;

  function TabPanel(props) {
    const { children, value, index } = props;

    return (
      <div>
        {value === index && (
          <Box sx={{ p: 3 }}>
            <Typography>{children}</Typography>
          </Box>
        )}
      </div>
    );
  }

  const [createOpen, setCreateOpen] = React.useState(false);
  const [updateOpen, setUpdateOpen] = React.useState({ name: false, entity: {} });
  const [deleteOpen, setDeleteOpen] = React.useState({ isOpen: false, id: '' });

  const entitiesSchema = {
    partnersEmploymentInfo: [
      {
        fieldName: 'name',
        type: 'string',
        validationOnCreate: ['required', 'minStringLength:2'],
        validationOnUpdate: ['minStringLength:2'],
        errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
        errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
      },
      {
        fieldName: 'link',
        type: 'string',
        validationOnCreate: ['required', 'minStringLength:2'],
        validationOnUpdate: ['minStringLength:2'],
        errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
        errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
      },
      {
        fieldName: 'headerText',
        type: 'array',
        validationOnCreate: ['required', 'minStringLength:2'],
        validationOnUpdate: ['minStringLength:2'],
        errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
        errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
      },
      {
        fieldName: 'mainText',
        type: 'array',
        validationOnCreate: ['required', 'minStringLength:2'],
        validationOnUpdate: ['minStringLength:2'],
        errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
        errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
      },
      {
        fieldName: 'headerTextEn',
        type: 'array',
      },
      {
        fieldName: 'mainTextEn',
        type: 'array',
      }
    ]
  };

```

```

    },
  ],
  employmentMaterials: [
    {
      fieldName: 'info',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'infoEn',
      type: 'string',
    },
    {
      fieldName: 'link',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
  ],
  faqList: [
    {
      fieldName: 'question',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'answer',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'questionEn',
      type: 'string',
    },
    {
      fieldName: 'answerEn',
      type: 'string',
    },
  ],
  professionList: [
    {
      fieldName: 'name',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'nameEn',
      type: 'string',
    },
    {
      fieldName: 'requirements',
      type: 'array',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
  ],

```

```

    {
      fieldName: 'requirementsEn',
      type: 'array',
    },
    {
      fieldName: 'link',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'averagePayment',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'maxPayment',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'vacanciesNumber',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0'],
      validationOnUpdate: ['minNumber:0'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0)]
    },
  ],
  programmingLanguages: [
    {
      fieldName: 'name',
      type: 'string',
      validationOnCreate: ['required', 'minStringLength:2'],
      validationOnUpdate: ['minStringLength:2'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-LENGTH'](2)]
    },
    {
      fieldName: 'writeNowPerc',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0', 'maxNumber:100'],
      validationOnUpdate: ['minNumber:0', 'maxNumber:100'],
      errorsOnCreate: [
        phrases['ADMIN-ERROR-REQUIRED'],
        phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0),
        phrases['ADMIN-ERROR-MAX-FIELD-VALUE'](100)
      ],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0), phrases['ADMIN-ERROR-MAX-FIELD-VALUE'](100)]
    },
    {
      fieldName: 'changes',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0', 'maxNumber:100'],
      validationOnUpdate: ['minNumber:0', 'maxNumber:100'],
      errorsOnCreate: [
        phrases['ADMIN-ERROR-REQUIRED'],
        phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0),
        phrases['ADMIN-ERROR-MAX-FIELD-VALUE'](100)
      ],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0), phrases['ADMIN-ERROR-MAX-FIELD-VALUE'](100)]
    },
  ],

```

```

    {
      fieldName: 'writeNow',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0'],
      validationOnUpdate: ['minNumber:0'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
VALUE'](0)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0)]
    },
    {
      fieldName: 'useSec',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0'],
      validationOnUpdate: ['minNumber:0'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
VALUE'](0)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0)]
    },
    {
      fieldName: 'usePrim',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0'],
      validationOnUpdate: ['minNumber:0'],
      errorsOnCreate: [phrases['ADMIN-ERROR-REQUIRED'], phrases['ADMIN-ERROR-MIN-FIELD-
VALUE'](0)],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0)]
    },
    {
      fieldName: 'likeIndex',
      type: 'number',
      validationOnCreate: ['required', 'minNumber:0', 'maxNumber:100'],
      validationOnUpdate: ['minNumber:0', 'maxNumber:100'],
      errorsOnCreate: [
        phrases['ADMIN-ERROR-REQUIRED'],
        phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0),
        phrases['ADMIN-ERROR-MAX-FIELD-VALUE'](100)
      ],
      errorsOnUpdate: [phrases['ADMIN-ERROR-MIN-FIELD-VALUE'](0), phrases['ADMIN-ERROR-MAX-FIELD-
VALUE'](100)]
    },
  ],
};

const fieldNameLabel = {
  name: phrases['DIALOG-NAME-FIELD'],
  link: phrases['DIALOG-LINK-FIELD'],
  headerText: phrases['DIALOG-HEADER-TEXT-FIELD'],
  mainText: phrases['DIALOG-MAIN-TEXT-FIELD'],
  headerTextEn: phrases['DIALOG-HEADER-TEXT-EN-FIELD'],
  mainTextEn: phrases['DIALOG-MAIN-TEXT-EN-FIELD'],
  info: phrases['DIALOG-INFO-FIELD'],
  infoEn: phrases['DIALOG-INFO-EN-FIELD'],
  question: phrases['DIALOG-QUESTION-FIELD'],
  answer: phrases['DIALOG-ANSWEAR-FIELD'],
  questionEn: phrases['DIALOG-QUESTION-EN-FIELD'],
  answerEn: phrases['DIALOG-ANSWEAR-EN-FIELD'],
  nameEn: phrases['DIALOG-NAME-EN-FIELD'],
  requirements: phrases['DIALOG-REQUIREMENTS-FIELD'],
  requirementsEn: phrases['DIALOG-REQUIREMENTS-EN-FIELD'],
  averagePayment: phrases['DIALOG-AVERAGE-PAYMENT-FIELD'],
  maxPayment: phrases['DIALOG-MAX-PAYMENT-FIELD'],
  vacanciesNumber: phrases['DIALOG-VACANCIES-NUMBER-FIELD'],
  writeNowPerc: phrases['DIALOG-WRITE-NOW-PERC-FIELD'],
  changes: phrases['DIALOG-CHANGE-FIELD'],
  writeNow: phrases['DIALOG-WRITE-NOW-FIELD'],
  useSec: phrases['DIALOG-USE-SEC-FIELD'],
  usePrim: phrases['DIALOG-USE-PRIM-FIELD'],
  likeIndex: phrases['DIALOG-INDEX-FIELD'],
};

const handleCreateClickOpen = (e) => {
  setCreateOpen(e.target.name);
};

const handleUpdateClickOpen = (entity) => (e) => {
  setUpdateOpen({ name: e.currentTarget.name, entity });
};

```

```

const handleDeleteClickOpen = (id, dbname) => () => {
  setDeleteOpen({ openedEntity: dbname, id });
};

const handleCreateClose = () => {
  setCreateOpen(null);
};

const handleUpdateClose = () => {
  setUpdateOpen({ name: false, entity: {} });
};

const handleDeleteClose = () => {
  setDeleteOpen({ openedEntity: '', id: '' });
};

const handleCreateSubmit = (entityName, entity, fields) => () => {
  const createdEntity = Object.keys(entity).reduce((acc, cur) => {
    if (fields.find(field => {
      return field.fieldName === cur;
    }).type === 'array')
      acc[cur] = entity[cur].split('\n');
    else acc[cur] = entity[cur];
    return acc;
  }, {});
  props.createNewEntityThunk(createdEntity, entityName);
  setCreateOpen(null);
};

const handleDeleteConfirm = (entityName, id) => () => {
  handleDeleteClose();
  props.deleteEntityThunk(id, entityName);
};

const handleUpdateSubmit = (entityName, entity, fields, id) => () => {
  const updatedEntity = Object.keys(entity).reduce((acc, cur) => {
    if (fields.find(field => {
      return field.fieldName === cur;
    }).type === 'array')
      acc[cur] = entity[cur].split('\n');
    else acc[cur] = `${entity[cur]}`;
    return acc;
  }, {});
  props.updateEntityThunk(id, updatedEntity, entityName);
  setUpdateOpen({ name: false, entity: {} });
};

function CreateDialog(dialogProps) {
  const [form, setForm] = useState(Object.fromEntries(dialogProps.fields.map(field =>
    ([field.fieldName, '']))));
  const handleChange = (e) => {
    setForm({ ...form, [e.target.id]: e.target.value });
  };
  return <Dialog open={createOpen === dialogProps.dbname} onClose={handleCreateClose}
    {...dialogProps}>
    <ValidatorForm
      onSubmit={handleCreateSubmit(dialogProps.dbname, form, dialogProps.fields)}
      className='validatorForm'
    >
    <DialogTitle align='center'>{phrases['ADMIN-CREATE-NEW']} {dialogProps.dbname}</DialogTitle>
    <DialogContent className='adminDialogContent'>
      {dialogProps.fields.map((field, i) => {
        return <TextValidator
          className='adminDialogRow'
          key={field.fieldName}
          autoFocus={i === 0}
          value={form[field.fieldName]}
          id={field.fieldName}
          validators={field.validationOnCreate}
          errorMessages={field.errorsOnCreate}
          multiline
          maxRows={20}
          label={fieldNameLabel[field.fieldName]}
          fullWidth
          variant="outlined"
          onChange={handleChange}

```

```

    />;
  }}}

</DialogContent>
<DialogActions align='center'>
  <Grid container justifyContent={'space-evenly'}>
    <Button type='submit'>
      {phrases[ 'ADMIN-CREATE-CONFIRM' ]}
    </Button>
    <Button onClick={handleCreateClose}>{phrases[ 'ADMIN-CREATE-CANCEL' ]}</Button>
  </Grid>
</DialogActions>
</ValidatorForm>
</Dialog>;
}

function UpdateDialog(dialogProps) {
  const [form, setForm] = useState(Object.fromEntries(dialogProps.fields.map(field =>
    ([field.fieldName, Array.isArray(updateOpen.entity[field.fieldName]) ?
      updateOpen.entity[field.fieldName].join('\n') :
      updateOpen.entity[field.fieldName]]))));
  const handleChange = (e) => {
    setForm({ ...form, [e.target.id]: e.target.value });
  };
  return <Dialog open={updateOpen.name === dialogProps.dbname} onClose={handleUpdateClose}
{...dialogProps}>
  <ValidatorForm
    onSubmit={handleUpdateSubmit(dialogProps.dbname, form, dialogProps.fields,
updateOpen.entity._id)}
    className='validatorForm'
  >
    <DialogTitle align='center'>{phrases[ 'ADMIN-UPDATE' ]} {dialogProps.dbname}</DialogTitle>
    <DialogContent className='adminDialogContent'>
      {dialogProps.fields.map((field) => {
        return <TextValidator
          className='adminDialogRow'
          key={field.fieldName}
          value={form[field.fieldName]}
          id={field.fieldName}
          validators={field.validationOnUpdate}
          errorMessages={field.errorsOnUpdate}
          multiline
          maxRows={20}
          label={fieldNameLabel[field.fieldName]}
          fullWidth
          variant="outlined"
          onChange={handleChange}
        />;
      })}
    </DialogContent>
    <DialogActions align='center'>
      <Grid container justifyContent={'space-evenly'}>
        <Button type='submit'>{phrases[ 'ADMIN-UPDATE-CONFIRM' ]}</Button>
        <Button onClick={handleUpdateClose}>{phrases[ 'ADMIN-UPDATE-CANCEL' ]}</Button>
      </Grid>
    </DialogActions>
  </ValidatorForm>
</Dialog>;
}

function DeleteDialog(dialogProps) {
  return <Dialog open={deleteOpen.openedEntity === dialogProps.dbname} onClose={handleDeleteClose}>
    <DialogTitle align='center'>{phrases[ 'ADMIN-DELETE-CONFIRM-TITLE' ]}</DialogTitle>
    <DialogActions align='center'>
      <Grid container justifyContent={'space-evenly'}>
        <Button
          onClick={handleDeleteConfirm(
            dialogProps.dbname,
            deleteOpen.id,
          )}
        >{phrases[ 'ADMIN-DELETE-CONFIRM' ]}</Button>
        <Button onClick={handleDeleteClose}>{phrases[ 'ADMIN-DELETE-CANCEL' ]}</Button>
      </Grid>
    </DialogActions>
  </Dialog>;
}

```

```

const [value, setValue] = React.useState(0);

const tabs = [
  {
    name: phrases['HEADER-FOR-ABITURIENT'],
    changeableFields: [
      {
        name: phrases['PROFESSIONS-LIST-TITLE'],
        fieldName: 'professionList'
      },
      {
        name: phrases['TABLE-ENTRANT-TITLE'],
        fieldName: 'programmingLanguages'
      }
    ]
  },
  {
    name: phrases['HEADER-WORK'],
    changeableFields: [
      {
        name: phrases['EMPLOYMENT-PARTNERS-TITLE'],
        fieldName: 'partnersEmploymentInfo'
      },
      {
        name: phrases['INTERESTING-INFO-TITLE'],
        fieldName: 'employmentMaterials'
      }
    ]
  },
  {
    name: phrases['HEADER-FAQ'],
    changeableFields: [
      {
        name: phrases['FAQ-TITLE'],
        fieldName: 'faqList'
      }
    ]
  }
];

const getNameField = (fieldName) => {
  let name = '';
  switch (fieldName) {
    case 'programmingLanguages': name = 'name'; break;
    case 'partnersEmploymentInfo': name = 'name'; break;
    case 'employmentMaterials': name = `info${props.language === 2 ? 'En' : ''}`; break;
    case 'faqList': name = `question${props.language === 2 ? 'En' : ''}`; break;
    default: name = `name${props.language === 2 ? 'En' : ''}`;
  }
  return name;
};

const handleChange = (event, newValue) => {
  setValue(newValue);
};

const [error, setError] = useState('');
const [auth, setAuth] = useState('');

const handleAuthChange = (event) => {
  setAuth(event.target.value);
};

const validateAuth = () => {
  const authFromEnv = env.AUTH;
  return auth === authFromEnv;
};

const handleEnterAuthKey = () => {
  const isValidated = validateAuth();
  if (isValidated) {
    props.setAuthSuccess();
    return;
  }
  setError('Incorrect authorization code');
};

```

```

{
  return props.isAuth ? <Paper elevation={10} className='paper'>
    <Box sx={{ borderBottom: 1, borderColor: 'divider' }}>
      <Tabs
        value={value}
        onChange={handleChange}
        variant="scrollable"
        scrollButtons
        allowScrollButtonsMobile
      >
        {tabs.map(tab => <Tab label={tab.name} key={tab.name} />)}
      </Tabs>
    </Box>
    {tabs.map((tab, i) => <TabPanel value={value} index={i} key={tab.name + i}>
      {tab.changeableFields.map(field => <Accordion key={field.fieldName + i}>
        <AccordionSummary
          expandIcon={<ExpandMoreIcon />}
        >
          <Typography component='span'>{field.name}</Typography>
        </AccordionSummary>
        <AccordionDetails>
          <Grid container justifyContent='space-evenly'>
            {props[field.fieldName].map(detail =>
              <Grid item xs={12} md={6} lg={4}
                className='adminElementMarginBottom'
                key={Object.values(detail).join('')}
              >
                <Grid container justifyContent='space-evenly'>
                  <Grid item xs={12} sm={9}>
                    <Typography component='span'>
                      {detail[getNameField(field.fieldName)]}
                    </Typography>
                  </Grid>
                  <div>
                    <IconButton
                      size='small'
                      onClick={handleUpdateClickOpen(detail)}
                      name={field.fieldName}
                    >
                      <EditIcon />
                    </IconButton>
                    <IconButton
                      size='small'
                      onClick={handleDeleteClickOpen(detail._id, field.fieldName)}
                    ><DeleteIcon /></IconButton>
                  </div>
                </Grid>
              </Grid>)}
            <Grid item xs={12}>
              <Grid container justifyContent='center'>
                <Button onClick={handleCreateClickOpen} name={field.fieldName}>
                  {phrases['ADMIN-CREATE-BUTTON-TEXT']}
                </Button>
              </Grid>
            </Grid>
          </Grid>
        </AccordionDetails>
        <CreateDialog fields={entitiesSchema[field.fieldName]} dbname={field.fieldName} />
        <UpdateDialog fields={entitiesSchema[field.fieldName]} dbname={field.fieldName} />
        <DeleteDialog dbname={field.fieldName} />
      </Accordion>)}
    </TabPanel>)}
  </Paper>
  :
  <Paper elevation={10} className='paper'>
    <Typography variant='h4' align='center'>{phrases['ADMIN-AUTH-TITLE']}</Typography>
    <TextField fullWidth onChange={handleAuthChange} />
    <Typography align='center'>
      <Button onClick={handleEnterAuthKey}>
        {phrases['ADMIN-AUTH-CONFIRM']}
      </Button>
    </Typography>
    {error !== '' ?
      <Typography color='error' align='center'>
        {error}
      </Typography>
    }
  </Paper>
}

```

```

      : ''}
    </Paper>;
  }
};

```

```
export default AdminPanel;
```

Файл Employment.js;

```

import { Button, Divider, Grid, Paper, Typography } from '@mui/material';
import React from 'react';
import './Employment.css';
import { translateUkr } from '../../constants.ua';
import { translateEng } from '../../constants.eng';

const Employment = (props) => {
  const phrases = props.language === 1 ? translateUkr : translateEng;

  const partnersInfo = props.partnersEmploymentInfo;
  const employmentMaterials = props.employmentMaterials;

  const YouTubeLists = [
    'PLCyBjMS1_FyDqDAzxwO2f7xcnqGbhT107',
    'PLCyBjMS1_FyCtE6MvJamVijwCwQMd5RIF',
    'PLCyBjMS1_FyCh31XNIFscAluS59kgyQQF'
  ];

  return <Paper elevation={10} className='paper'>
    <Typography variant='h5' align='center'>{phrases['EMPLOYMENT-PARTNERS-TITLE']}

```

```
export default Employment;
```

Файл Entrant.js;

```

import {
  Button,
  Divider,

```

```

Grid,
Paper,
Typography,
TableContainer,
TableRow,
TableHead,
Table,
TableCell,
TableBody,
TableSortLabel
} from '@mui/material';
import React from 'react';
import './Entrant.css';
import { translateUkr } from '../../constants.ua';
import { translateEng } from '../../constants.eng';

const createData = (name, writePercent, change, writeNum, useSecondary, useNow, rate) => {
  return {
    name,
    writePercent,
    change,
    writeNum,
    useSecondary,
    useNow,
    rate,
  };
};

const Entrant = (props) => {
  const phrases = props.language === 1 ? translateUkr : translateEng;

  const rows = props.programmingLanguages.map(lang => createData(
    lang.name,
    lang.writeNowPerc,
    lang.changes,
    lang.writeNow,
    lang.useSec,
    lang.usePrim,
    lang.likeIndex
  ));

  const [order, setOrder] = React.useState('desc');
  const [orderBy, setOrderBy] = React.useState('name');

  const handleSort = (property) => {
    const isAsc = orderBy === property && order === 'asc';
    setOrder(isAsc ? 'desc' : 'asc');
    setOrderBy(property);
  };

  const createSortHandler = (property) => (event) => {
    handleSort(property, event);
  };

  const headCells = [
    {
      id: 'name',
      label: phrases['TABLE-ENTRANT-LANGUAGE'],
    },
    {
      id: 'writePercent',
      label: phrases['TABLE-ENTRANT-WRITE-NOW-PERC'],
    },
    {
      id: 'change',
      label: phrases['TABLE-ENTRANT-CHANGE'],
    },
    {
      id: 'writeNum',
      label: phrases['TABLE-ENTRANT-WRITE-NOW'],
    },
    {
      id: 'useSecondary',
      label: phrases['TABLE-ENTRANT-USE-SECONDARY'],
    },
    {
      id: 'useNow',

```

```

    label: phrases['TABLE-ENTRANT-USE-PRIMARY'],
  },
  {
    id: 'rate',
    label: phrases['TABLE-ENTRANT-LIKE-INDEX'],
  },
];

const professions = props.professionList;
const exams = phrases['ENTRANT-EXAM-SCHEDULE'].split(',');
return <Paper elevation={10}>
  <Grid container justifyContent={'space-evenly'}>
    <Grid item xs={12} md={5.99} className='entrantElement'>
      <Grid container direction={'column'} justifyContent='space-evenly'
className='containerFullHeight'>
        <Typography align='center' variant='h6'>{phrases['ENTRANT-EXAM-
CERTIFICATES']}</Typography>
        <div className='entrantListElement'>
          <Typography>{phrases['ENTRANT-EXAN-MAIN-SUBJECTS']}</Typography>
          <Typography>{phrases['ENTRANT-EXAN-ADDITIONAL-SUBJECTS']}</Typography>
        </div>
        <Typography align='center'>{phrases['ENTRANT-EXAM-SCHEDULE-TITLE']}</Typography>
        <Grid container className='entrantListElement'>
          {exams.map((exam, index) => {
            return <Grid item xs={12} sm={6} lg={4} key={exam + index}>
              <Typography align='center'>
                {exam}
              </Typography>
            </Grid>;
          })}
        </Grid>
        <Typography align='center'>{phrases['ENTRANT-EXAM-EXTRA-SESSION-TITLE']}</Typography>
        <Typography
          className='entrantListElement'>{phrases['ENTRANT-EXAM-EXTRA-
SESSION']}</Typography>
        <Grid container justifyContent={'center'}>
          <Button size='large'>
            <a href="https://zno.testportal.com.ua/registration" className='link'>{phrases['EXAM-
REGISTER']}</a>
          </Button>
        </Grid>
      </Grid>
    </Grid>
    <Grid item xs={0}><Divider orientation='vertical' /></Grid>
    <Grid item xs={12} md={5.99} className='entrantElement'>
      <Grid container direction={'column'} justifyContent='space-evenly'
className='containerFullHeight'>
        <Typography align='center' variant='h6'>{phrases['UNIVERSITY-ENTRY-DOCUMENT-LIST-
TITLE']}</Typography>
        {phrases['UNIVERSITY-ENTRY-DOCUMENT-LIST-ARRAY'].map(
          item => <Typography className='entrantListElement' key={item}>
            {item}
          </Typography>)}
        <Grid container justifyContent={'space-evenly'}>
          <Button className='entrantListElement' size='large'>
            <a href="https://pk.zp.edu.ua/pravyla-pryjomu/abituriyentam-pilgovyh-kategorij"
              className='link'>{phrases['UNIVERSITY-ENTRY-QUESTIONS']}</a>
          </Button>
          <Button className='entrantListElement' size='large'>
            <a href="https://pk.zp.edu.ua/" className='link'>{phrases['UNIVERSITY-ENTRY-
BENEFITS']}</a>
          </Button>
        </Grid>
      </Grid>
    </Grid>
    <Grid item xs={0} md={12}><Divider /></Grid>
    <Grid item xs={12}>
      <Typography align='center' variant='h6' className='entrantElement'>
        {phrases['PROFESSIONS-LIST-TITLE']}
      </Typography>
      <Grid container>
        {professions.map((prof) => <Grid item xs={12} md={6} className='partMainPadding'
key={prof.name}>
          <Grid container className='containerFullHeight' direction={'column'}
justifyContent='space-between'>
            <Typography align='center'>

```

```

        <a href={prof.link} className='link'>{prof[`name${props.language} === 2 ? 'En' :
        ''`]}</a>
      </Typography>
    <ul>
      {prof[`requirements${props.language} === 2 ? 'En' : ''`].map(req => <li key={req}>
        <Typography>{req}</Typography>
      </li>)}
    </ul>
    <Grid container>
      <Grid item xs={12} md={4}>
        <Typography align='center'>{phrases[ 'AVERAGE-PAYMENT' ]}
        {prof.averagePayment}</Typography>
      </Grid>
      <Grid item xs={12} md={4}>
        <Typography align='center'>{phrases[ 'MAX-PAYMENT' ]} {prof.maxPayment}</Typography>
      </Grid>
      <Grid item xs={12} md={4}>
        <Typography align='center'>{phrases[ 'VACANCIES-NUMBER' ]}
        {prof.vacanciesNumber}</Typography>
      </Grid>
    </Grid>
    <Divider />
  </Grid>
</Grid>
<Typography className='entrantElement' variant='h6'>{phrases[ 'TABLE-ENTRANT-
TITLE' ]}</Typography>
<TableContainer component={Paper}>
  <Table sx={{ minWidth: 650 }} aria-label="simple table">
    <TableHead>
      <TableRow>
        {headCells.map(cell => <TableCell align='center' {...cell} key={cell.id}>
          <TableSortLabel
            active={orderBy === cell.id}
            direction={orderBy === cell.id ? order : 'asc'}
            onClick={createSortHandler(cell.id)}
          >
            {cell.label}
          </TableSortLabel>
        </TableCell>)}
      </TableRow>
    </TableHead>
    <TableBody>
      {rows.sort((a, b) => {
        return orderBy === 'name' ?
          +(a[orderBy] < b[orderBy]) * (order === 'asc' ? -1 : 1) :
          (b[orderBy] - a[orderBy]) * (order === 'asc' ? -1 : 1);
      }).map((row) => (
        <TableRow
          key={row.name}
        >
          {Object.entries(row).map(entry => {
            return <TableCell align='center' key={entry}>
              {entry[1] + (entry[0] === 'rate' ? '%' : '')}
            </TableCell>;
          })}
        </TableRow>
      ))}
    </TableBody>
  </Table>
</TableContainer>
</Grid>
</Paper>;
};

```

export default Entrant;

Файл FAQ.js;

```

import { Accordion, AccordionDetails, AccordionSummary, Grid, Paper, Typography } from
'@mui/material';
import React from 'react';
import './FAQ.css';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';
import { translateUkr } from '../constants.ua';
import { translateEng } from '../constants.eng';

```

```

const FAQ = (props) => {
  const phrases = props.language === 1 ? translateUkr : translateEng;

  return <Paper elevation={10} className='paper'>
    <Typography align='center' variant='h5'>{phrases['FAQ-TITLE']}

```

export default FAQ;

Файл Footer.js;

```

import React, { useEffect, useState } from 'react';
import { createTheme, FormControl, Grid, InputLabel, MenuItem, Select, Typography, ThemeProvider,
Divider } from '@mui/material';
import './Footer.css';
import '../App.css';
import { translateUkr } from '../constants.ua';
import { translateEng } from '../constants.eng';
import { makeStyles } from '@mui/styles';

import Cookies from 'universal-cookie';

const theme = createTheme({
  palette: {
    error: { main: '#ebd9c8' },
    secondary: { main: '#ebd9c8' }
  },
});

const useStyles = makeStyles({
  divider: {
    backgroundColor: '#ebd9c8',
  },
  label: {
    color: '#ebd9c8',
    '&.Mui-focused': {
      color: '#ebd9c8',
    },
  },
  select: {
    color: '#ebd9c8',
    '& .MuiSvgIcon-root': {
      color: '#ebd9c8',
    },
  },
});

const Footer = (props) => {
  useEffect(() => {
    setLanguage(props.language);
  });

  const classes = useStyles();

  const handleLanguageChange = (event) => {
    setLanguage(event.target.value);
    props.setLanguage(event.target.value);
  };

```

```

    const cookies = new Cookies();
    cookies.set('language', event.target.value, { path: '/' });
  };

  const [language, setLanguage] = useState(1);

  const phrases = props.language === 1 ? translateUkr : translateEng;

  return (
    <Grid container justifyContent={'space-evenly'} className='mainFooterContainer'>
      <Grid item xs={12} md={4}>
        <Grid container justifyContent={'center'} className='containerFullHeight'
direction={'column'}>
          <Typography textAlign={'center'}>{phrases['FOOTER-ADRESS-TITLE']}</Typography>
          <Typography textAlign={'center'}>{phrases['FOOTER-ADRESS']}</Typography>
        </Grid>
      </Grid>
      <Grid item>
        <Divider orientation='vertical' className={classes.divider} />
      </Grid>
      <Grid item xs={12} md={4}>
        <Grid container justifyContent={'center'} className='containerFullHeight'
direction={'column'}>
          <Typography textAlign={'center'}>{phrases['FOOTER-EMAIL-TITLE']} {phrases['FOOTER-
EMAIL']}</Typography>
          <Typography textAlign={'center'}>{phrases['FOOTER-PHONE-TITLE']} {phrases['FOOTER-
PHONE']}</Typography>
        </Grid>
      </Grid>
      <Grid item>
        <Divider orientation='vertical' className={classes.divider} />
      </Grid>
      <Grid item xs={12} md={3}>
        <Grid container justifyContent={'center'} className='containerFullHeight'
direction={'column'}>
          <Typography textAlign={'center'}>{phrases['FOOTER-LANGUAGE-CHANGE']}</Typography>
          <ThemeProvider theme={theme}>
            <FormControl>
              <InputLabel color={'secondary'} id="language-label" className={classes.label}>
                {phrases['FOOTER-LANGUAGE-TITLE']}
              </InputLabel>
              <Select
                color='secondary'
                labelId="language-label"
                label={phrases['FOOTER-LANGUAGE-TITLE']}
                value={language}
                error
                className={classes.select}
                onChange={handleLanguageChange}
              >
                <MenuItem value={1}>Українська</MenuItem>
                <MenuItem value={2}>English</MenuItem>
              </Select>
            </FormControl>
          </ThemeProvider>
        </Grid>
      </Grid>
    </Grid>
  );
};

```

export default Footer;

Файл Header.js;

```

import React, { useEffect, useState } from 'react';
import { Button, createTheme, Grid, IconButton, Typography } from '@mui/material';
import './Header.css';
import { NavLink } from 'react-router-dom';
import { translateUkr } from '../constants.ua';
import { translateEng } from '../constants.eng';
import Cookies from 'universal-cookie';
import {
  Menu as MenuIcon,
  ArrowBack as ArrowBackIcon
} from '@mui/icons-material';

```

```

import { ThemeProvider } from '@emotion/react';

const Header = (props) => {
  useEffect(() => {
    const cookies = new Cookies();
    const language = +cookies.get('language');
    language && props.setLanguage(language);

    const header = document.querySelector('.headerMobile');

    const scrollUp = 'scroll-up';
    const scrollDown = 'scroll-down';
    let lastScroll = 0;

    window.addEventListener('scroll', (e) => {
      const currentScroll = window.pageYOffset;
      if (currentScroll <= 0) {
        header.classList.remove(scrollUp);
        header.style.position = 'static';
        return;
      }

      header.style.position = 'fixed';

      if (currentScroll > lastScroll && !header.classList.contains(scrollDown)) {
        // down
        header.classList.remove(scrollUp);
        header.classList.add(scrollDown);
      } else if (
        currentScroll < lastScroll &&
        header.classList.contains(scrollDown)
      ) {
        // up
        header.classList.remove(scrollDown);
        header.classList.add(scrollUp);
      }
      lastScroll = currentScroll;
    });
  });

  const phrases = props.language === 1 ? translateUkr : translateEng;

  const [showBar, setShowBar] = useState(false);

  const headers = [
    { text: phrases['HEADER-FOR-ABITURIENT'], url: 'forentrant' },
    // { text: phrases['HEADER-SPECIALITIES'], url: 'speciality' },
    { text: phrases['HEADER-MAIN'], url: 'home' },
    // { text: phrases['HEADER-STUDY-ABROAD'], url: 'abroad' },
    { text: phrases['HEADER-WORK'], url: 'employment' },
    { text: phrases['HEADER-FAQ'], url: 'faq' },
  ];

  const theme = createTheme({
    palette: {
      secondary: { main: '#ebd9c8' }
    },
  });

  const handleMenuClick = () => {
    setShowBar(!showBar);
  };

  const SideBar = props => {
    return (
      <div {...props}>
        <nav>
          <Typography align='right'>
            <Button className={classes.Closer} onClick={handleMenuClick}>
              <ArrowBackIcon color='secondary' />
            </Button>
          </Typography>
          {headers.map(title =>
            <Grid item xs={12} key={title.url}>
              <Grid container justifyItems='center' style={{ height: '100%' }}>
                <NavLink to={title.url}
                  className={({ isActive }) =>

```

```

        isActive ? 'nav titleNavActive' : 'nav'
      }
    >
      {title.text}
    </NavLink>
  </Grid>
</Grid>
  )}
</nav>
</div>
);
};

return (
  <div>
    <Grid container className='header'>
      {headers.map(title =>
        <Grid item xs={3} key={title.url} className='title'>
          <Grid container justifyItems={'center'} style={{ height: '100%' }}>
            <NavLink to={title.url}
              className={{ isActive }} =>
                isActive ? 'nav titleNavActive' : 'nav'
            >
              {title.text}
            </NavLink>
          </Grid>
        </Grid>
      )}
    </Grid>
    <Grid container className={`header headerMobile ${showBar ? 'headerMobileHide' : ''}`}>
      <ThemeProvider theme={theme}>
        <IconButton
          size="large"
          edge="start"
          color="secondary"
          aria-label="menu"
          sx={{ mr: 2 }}
          onClick={handleMenuClick}
          className='menu'
        >
          <MenuIcon />
        </IconButton>
      </ThemeProvider>
    </Grid>
    <Grid container className={`header headerMobile ${showBar ? 'sideBar' : 'hideSideBar'}`}>
      <ThemeProvider theme={theme}>
        <SideBar />
      </ThemeProvider>
    </Grid>
  </div>
);
};

```

export default Header;

Файл MainPage.js;

```

import { Grid, Typography, Paper, Divider } from '@mui/material';
import React from 'react';
import { ReactComponent as Logo } from '../img/logo.svg';
import { ReactComponent as Nuzp } from '../img/NUZP-GERB.svg';
import './MainPage.css';
import { translateUkr } from '../constants.ua';
import { translateEng } from '../constants.eng';

const MainPage = (props) => {
  const phrases = props.language === 1 ? translateUkr : translateEng;

  return <Paper elevation={10} className='paper'>
    <Grid container justifyContent={'space-evenly'} alignItems={'center'}>
      <Grid item xs={3}>
        <a href="https://zp.edu.ua/"><Nuzp className='logo' /></a>
      </Grid>
      <Grid item xs={6}>
        <Grid container alignItems={'center'} direction={'column'}>
          <a href="https://zp.edu.ua/kafedra-programnih-zasobiv" className='link'>

```

```

        <Typography variant='h5'>{phrases[ 'DEPARTMENT-NAME' ]}</Typography>
      </a>
      <Typography>{phrases[ 'ACTIVITY-START-YEAR' ]}: 2002</Typography>
      <Typography>{phrases.ADMINISTRATION}</Typography>
      <Typography>{phrases[ 'ADMINISTRATION-OF-KAFEDRA' ]}
        <a href="https://zp.edu.ua/?q=node/664" className='link'>{phrases[ 'HEAD-OF-THE-
DEPARTMENT' ]}</a>
      </Typography>
      <Typography><a href='https://pz.zntu.net/kafedra/vikladachi/'
        className='link'>{phrases[ 'TEACHERS-LINK-TITLE' ]}</a>
      </Typography>
    </Grid>
  </Grid>
  <Grid item xs={3}>
    <a href="https://zp.edu.ua/kafedra-programnih-zasobiv"><Logo className='logo' /></a>
  </Grid>
  <Grid item xs={12}>
    <Divider className='divider'></Divider>
  </Grid>
  <Grid container justifyContent='space-evenly' justifyItems='flex-start'
className='additionalContent'>
    <Grid item xs={12} className='partMargin'>
      <Typography>{phrases[ 'DESCRIPTION-1' ]}</Typography>
      <br />
      <Typography>{phrases[ 'DESCRIPTION-2' ]}</Typography>
      <br />
      <Typography>{phrases[ 'DESCRIPTION-3' ]}</Typography>
    </Grid>
    <Grid item xs={12}>
      <Typography align='center' className='partMargin'
variant='h6'>{phrases.SPECIALITIES}</Typography>
    </Grid>
    <Grid item xs={5}>
      <Grid container className='containerFullHeight' direction='{column}'
justifyContent='space-between'>
        <Grid item>
          <Typography align='center' variant='h6'>{phrases[ 'SPECIALITY-1' ]}</Typography>
          <Typography align='center' className='studyProgram'>{phrases[ 'SPECIALITY-1-
STUDYPROGRAM' ]}</Typography>
        </Grid>
        <Typography className='partMargin'>{phrases[ 'SPECIALITY-1-DESCRIPTION' ]}</Typography>
      </Grid>
    </Grid>
    <Grid item>
      <Divider orientation='vertical' />
    </Grid>
    <Grid item xs={5}>
      <Grid container className='containerFullHeight' direction='{column}'
justifyContent='space-between'>
        <Grid item>
          <Typography align='center' variant='h6'>{phrases[ 'SPECIALITY-2' ]}</Typography>
          <Typography align='center' className='studyProgram'>{phrases[ 'SPECIALITY-2-
STUDYPROGRAM-1' ]}</Typography>
          <Typography align='center' className='studyProgram'>{phrases[ 'SPECIALITY-2-
STUDYPROGRAM-2' ]}</Typography>
        </Grid>
        <Typography className='partMargin'>{phrases[ 'SPECIALITY-2-DESCRIPTION' ]}</Typography>
      </Grid>
    </Grid>
    <Grid item xs={12}>
      <Typography align='center' variant='h6'>{phrases[ 'STUDY-PRICE-TITLE' ]}</Typography>
    </Grid>
    <Grid item xs={5}>
      <Typography align='center' className='partMargin'>{phrases[ 'STUDY-PRICE-1' ]}</Typography>
    </Grid>
    <Grid item>
      <Divider orientation='vertical' />
    </Grid>
    <Grid item xs={5}>
      <Typography align='center' className='partMargin'>{phrases[ 'STUDY-PRICE-2' ]}</Typography>
    </Grid>
    <Grid item xs={12}>
      <Typography align='center' variant='h6'>{phrases[ 'FREE-PLACES-TITLE' ]}</Typography>
    </Grid>
    <Grid item xs={5}>
      <Typography align='center' className='partMargin'>{phrases[ 'FREE-PLACES-1' ]}</Typography>

```

```

    </Grid>
    <Grid item>
      <Divider orientation='vertical' />
    </Grid>
    <Grid item xs={5}>
      <Typography align='center' className='partMargin'>{phrases[ 'FREE-PLACES-2' ]}</Typography>
    </Grid>
    <Grid item xs={12}>
      <Typography align='center' variant='h6'>{phrases[ 'PARTNERS-TITLE' ]}</Typography>
      <Typography className='partMargin'>{phrases[ 'PARTNERS' ]}</Typography>
    </Grid>
  </Grid>
</Grid>
</Paper>;
};

```

```
export default MainPage;
```

Файл RouteController.jsx;

```

import { Route, Routes, Navigate } from 'react-router-dom';
import React from 'react';
import { Box } from '@mui/material';
import MainPageContainer from '../Containers/MainPageContainer';
import EntrantContainer from '../Containers/EntrantContainer';
import EmploymentContainer from '../Containers/EmploymentContainer';
import FAQContainer from '../Containers/FAQContainer';
import AdminPanelContainer from '../Containers/AdminPanelContainer';

const RouteController = () => (
  <Box style={{ padding: '20px' }}>
    <Routes>
      <Route exact path='/home' element={MainPageContainer} /> />
      <Route exact path='/forentrant' element={EntrantContainer} /> />
      <Route exact path='/employment' element={EmploymentContainer} /> />
      <Route exact path='/faq' element={FAQContainer} /> />
      <Route exact path='/admin' element={AdminPanelContainer} /> />
      <Route path="*" element={Navigate to="/home"} /> />
    </Routes>
  </Box>
);

```

```
export default RouteController;
```

Файл AdminPanelContainer.js;

```

import { connect } from 'react-redux';
import React, { useEffect } from 'react';
import AdminPanel from '../Components/AdminPanel/AdminPanel';
import {
  getAllDataThunk,
  createNewEntityThunk,
  setAuthSuccess,
  updateEntityThunk,
  deleteEntityThunk
} from '../Redux/generalReducer';

const AdminPanelContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <AdminPanel {...props} />
  );
};

const mapStateToProps = (state) => {
  return {
    language: state.general.language,
    partnersEmploymentInfo: state.general.partnersEmploymentInfo,
    employmentMaterials: state.general.employmentMaterials,
    faqList: state.general.faqList,
    professionList: state.general.professionList,
    programmingLanguages: state.general.programmingLanguages,
    isAuth: state.general.isAuth,
  };
};

```

```
export default connect(mapStateToProps, {
  getAllData: getAllDataThunk,
  createNewEntityThunk,
  setAuthSuccess,
  updateEntityThunk,
  deleteEntityThunk,
})(AdminPanelContainer);
```

Файл EmploymentContainer.js;

```
import { connect } from 'react-redux';
import React, { useEffect } from 'react';
import Employment from '../Components/Employment/Employment';

const EmploymentContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <Employment {...props} />
  );
};

const mapStateToProps = (state) => {
  return {
    language: state.general.language,
    partnersEmploymentInfo: state.general.partnersEmploymentInfo,
    employmentMaterials: state.general.employmentMaterials,
  };
};
```

```
export default connect(mapStateToProps, {})(EmploymentContainer);
```

Файл EntrantContainer.js;

```
import { connect } from 'react-redux';
import React, { useEffect } from 'react';
import Entrant from '../Components/Entrant/Entrant';

const EntrantContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <Entrant {...props} />
  );
};

const mapStateToProps = (state) => {
  return {
    language: state.general.language,
    professionList: state.general.professionList,
    programmingLanguages: state.general.programmingLanguages,
  };
};
```

```
export default connect(mapStateToProps, {})(EntrantContainer);
```

Файл FAQContainer.js;

```
import { connect } from 'react-redux';
import React, { useEffect } from 'react';
import FAQ from '../Components/FAQ/FAQ';

const FAQContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <FAQ {...props} />
  );
};

const mapStateToProps = (state) => {
  return {
    language: state.general.language,
    faqList: state.general.faqList,
  };
};
```

```
export default connect(mapStateToProps, {})(FAQContainer);
```

Файл FooterContainer.js;

```
import { connect } from 'react-redux';
import React, { useEffect } from 'react';
import Footer from '../Components/Footer/Footer';
import { setLanguage } from '../Redux/generalReducer';
```

```
const FooterContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <Footer {...props} />
  );
};
```

```
const mapStateToProps = (state) => {
  return {
    language: state.general.language,
  };
};
```

```
export default connect(mapStateToProps, {setLanguage})(FooterContainer);
```

Файл HeaderContainer.js;

```
import { connect } from 'react-redux';
import React, { useEffect } from 'react';
import Header from '../Components/Header/Header';
import { setLanguage } from '../Redux/generalReducer';
```

```
const HeaderContainer = (props) => {
  useEffect(() => {
  }, []);
  return (
    <Header {...props} />
  );
};
```

```
const mapStateToProps = (state) => {
  return {
    language: state.general.language,
  };
};
```

```
export default connect(mapStateToProps, {setLanguage})(HeaderContainer);
```

Файл MainPageContainer.js;

```
import { connect } from 'react-redux';
import React from 'react';
import MainPage from '../Components/MainPage/MainPage';
```

```
const MainPageContainer = (props) => {
  return (
    <MainPage {...props} />
  );
};
```

```
const mapStateToProps = (state) => {
  return {
    language: state.general.language,
  };
};
```

```
export default connect(mapStateToProps, {})(MainPageContainer);
```

Файл generalReducer.js;

```
import {
  createNewEntity,
  deleteEntity,
```

```

    getAllData,
    updateEntity
  } from '../Api/Api';

const initialState = {
  language: 1,
  partnersEmploymentInfo: [
    {
      name: 'Extrums',
      link: 'https://extrums.com/',
      headerText: ['Ищешь место, где получить практический опыт в направлении Front-end (React.js) или Back-end (Node.js) с персональным ментором, стипендией и возможностью получить оффер по окончании стажировки?'],
      mainText: ['Тогда компания Extrums (https://jobs.dou.ua/companies/extrums/) приглашает тебя на 3-х месячную оплачиваемую стажировку. Мы предлагаем тебе все то, что перечислено ниже и даже больше – работу над реальными кейсами.',
        'Вот что мы предлагаем нашим будущим сотрудникам 😊',
        '■гибкий график работы, оплачиваемый отпуск и больничный, выходные на все государственные праздники;',
        '■возможность работать удаленно;',
        '■регулярные корпоративы;',
        '■компенсация за обучающие курсы, английский язык, занятия спортом;',
        '■комфортные условия работы – офис со всеми удобствами в самом центре города на проспекте Маяковского;',
        '■денежные бонусы за присоединение к компании;',
        '■оплачиваемые программы стажировок.'],
      headerTextEn: ['Are you looking for a place to gain practical experience in Front-end (React.js) or Back-end (Node.js) with a personal mentor, scholarship and the opportunity to receive an offer at the end of the internship?'],
      mainTextEn: ['Then Extrums (https://jobs.dou.ua/companies/extrums/) invites you to a 3-month paid internship. We offer you all that is listed below and even more - work on real cases.',
        'Here's what we offer our future employees 😊',
        '■flexible work schedule, paid leave and sick leave, weekends on all public holidays;',
        '■ability to work remotely;',
        '■regular corporations;',
        '■compensation for training courses, English language, sports;',
        '■comfortable working conditions - office with all amenities in the heart of the city on Mayakovsky Avenue;',
        '■money bonuses for joining the company;',
        '■paid internship programs.'],
    },
    {
      name: 'Computools',
      link: 'https://computools.com/',
      headerText: [''],
      mainText: ['Компания Computools предлагает студентам широкие возможности развития:',
        '■Стажировка.',
        '■Пассивная стажировка.',
        '■Практика.',
        '■Обучение английским с техническим обучением.',
        '■Прилагаем документ, в котором вы подробнее сможете ознакомиться со всеми видами сотрудничества.',
        '■Просим заполнить небольшую анкету, чтобы команда Computools могла с вами связаться и предложить вам наиболее подходящий вариант!',
        'https://docs.google.com/forms/d/e/1FAIpQLSeOGutSFnuatzJ9-GZpvlyDEKrv057t2i4q8uF4W2dMT0x-yw/viewform'],
      headerTextEn: [''],
      mainTextEn: ['Computools offers students a wide range of development opportunities:',
        '■Internship.',
        '■Passive internship.',
        '■Practice.',
        '■Englishing with technical training.',
        '■We enclose a document in which you will be able to get acquainted in more detail with all types of cooperation.',
        '■Please fill out a small form so that the Computools team can contact you and offer you the best option!',
        'https://docs.google.com/forms/d/e/1FAIpQLSeOGutSFnuatzJ9-GZpvlyDEKrv057t2i4q8uF4W2dMT0x-yw/viewform']
    }
  ],
  employmentMaterials: [
    {
      info: 'Зустріч з представниками фірм Computools, Extrums, GroupBWT, Natife',
      infoEn: 'Meeting with representatives of Computools, Extrums, GroupBWT, Natife',
    }
  ]
}

```

```

    link: 'https://youtu.be/LVw3gPTrmLA'
  }
],
faqList: [
  {
    id: '1',
    question: 'Запитання1',
    answer: 'Відповідь1',
    questionEn: 'question test1',
    answerEn: 'answer test1',
  },
  {
    id: '2',
    question: 'Запитання2',
    answer: 'Відповідь2',
    questionEn: 'question test2',
    answerEn: 'answer test2'
  },
  {
    id: '3',
    question: 'Запитання3',
    answer: 'Відповідь3',
    questionEn: 'question test3',
    answerEn: 'answer test3'
  },
  {
    id: '4',
    question: 'Запитання4',
    answer: 'Відповідь4',
    questionEn: 'question test4',
    answerEn: 'answer test4'
  },
],
professionList: [
  {
    name: 'Програміст-розробник',
    nameEn: 'Developer',
    requirements: [
      'Знання необхідних технологій та мов програмування, залежно від вакансії',
      'Уміння писати та правильно читати технічне завдання (ТЗ)',
      'Знання англійської мови на технічному рівні',
      'Soft skills: постійний саморозвиток, аналітичний склад розуму, відповідальність, комунікабельність, робота в команді.'
    ],
    requirementsEn: [
      'Knowledge of required technologies and programming languages, depending on the vacancy',
      'Ability to write and correctly read the technical task (TOR)',
      'Knowledge of English at the technical level',
      'Soft skills: constant self-development, analytical mind, responsibility, sociability, teamwork.'
    ],
    link: 'https://www.work.ua/jobs-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BD%D0%B8%D0%BA/',
    averagePayment: '2500$',
    maxPayment: '5300$',
    vacanciesNumber: '2400'
  },
  {
    name: 'Системний адміністратор',
    nameEn: 'System administrator',
    requirements: [
      'Розуміти принципи роботи мережевого обладнання, мережевих протоколів',
      'Знати і вміти налаштовувати операційні системи',
      'Вміти працювати з доменними службами, поштовими службами',
      'Володіти широким колом знань різних систем і технологій'
    ],
    requirementsEn: [
      'Understand the principles of operation of network equipment, network protocols',
      'Know and be able to configure operating systems',
      'Be able to work with domain services, postal services',
      'Have a wide range of knowledge of different systems and technologies'
    ],
    link: 'https://www.work.ua/jobs-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B8%D0%B9+%D0%B0%D0%B4%D0%BC%D1%96%D0%BD%D1%96%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80/',
    averagePayment: '700$',
    maxPayment: '4000$',
    vacanciesNumber: '620'
  }
]

```

```

    },
    {
      name: 'Системний аналітик',
      nameEn: 'Systems analyst',
      requirements: [
        'Знання персонального комп'ютера та пакетів Microsoft Office, SQL, XML та баз даних',
        'Організація управління проектами з розробки та впровадження',
        'Робота з Інтернет-технологіями та системами аналізу даних',
        'Знання сучасних мов програмування'
      ],
      requirementsEn: [
        'Knowledge of personal computer and Microsoft Office, SQL, XML and databases',
        'Organization of project management for development and implementation',
        'Working with Internet technologies and data analysis systems',
        'Knowledge of modern programming languages'
      ],
      link: 'https://www.work.ua/jobs-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B8%D0%B9+%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D0%BA/',
      averagePayment: '1000$',
      maxPayment: '5000$',
      vacanciesNumber: '45'
    },
    {
      name: 'Бізнес-аналітик',
      nameEn: 'Business analyst',
      requirements: [
        'Комунікативні навички',
        'Увага до деталей',
        'Уміння вирішувати проблеми',
        'Готовність ухвалювати рішення'
      ],
      requirementsEn: [
        'Communication skills',
        'Attention to detail',
        'Ability to solve problems',
        'Willingness to make decisions'
      ],
      link: 'https://www.work.ua/jobs-%D0%B1%D1%96%D0%B7%D0%BD%D0%B5%D1%81-%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D0%BA/',
      averagePayment: '1000$',
      maxPayment: '6000$',
      vacanciesNumber: '236'
    },
    {
      name: 'Графічний дизайнер',
      nameEn: 'Graphic designer',
      requirements: [
        'Володіння програмним забезпеченням для графічного дизайну',
        'Відчуття стилю, креативність, образне та об'ємно-просторове мислення',
        'Знання основ дизайну: основи живопису і малюнка, колористики(використання кольору) і композиції та ін.',
        'Спеціальні знання в області графічного дизайну'
      ],
      requirementsEn: [
        'Possession of software for graphic design',
        'Sense of style, creativity, figurative and three-dimensional thinking',
        'Knowledge of the basics of design: the basics of painting and drawing, color (use of color) and composition, etc.',
        'Special knowledge in the field of graphic design'
      ],
      link: 'https://www.work.ua/jobs-graphic+designer/',
      averagePayment: '700$',
      maxPayment: '4000$',
      vacanciesNumber: '205'
    },
    {
      name: 'Розробник ігор',
      nameEn: 'Game developer',
      requirements: [
        'Робота з мовами програмування',
        'Знання технічної англійської мови',
        'Розробка концепції гри та ігрових механік, жанру, сюжету, характерів персонажів тощо',
        'Знання Python, Lua або інших скриптових мовах',
        'Аналіз статистики та балансування гри'
      ],
      requirementsEn: [
        'Working with programming languages',

```

```

    'Knowledge of technical English',
    'Development of the concept of the game and game mechanics, genre, plot, characters, etc.',
    'Knowledge of Python, Lua or other scripting languages',
    'Analysis of statistics and game balancing',
  ],
  link: 'https://www.work.ua/jobs-game+developer/',
  averagePayment: '800$',
  maxPayment: '5000$',
  vacanciesNumber: '32'
},
],
programmingLanguages: [
  {
    name: 'JavaScript',
    writeNowPerc: '18.8',
    changes: '+0.17',
    writeNow: '1601',
    useSec: '3385',
    usePrim: '1089',
    likeIndex: '49.8',
  },
  {
    name: 'C#',
    writeNowPerc: '14.7',
    changes: '-0.02',
    writeNow: '1252',
    useSec: '346985',
    usePrim: '455',
    likeIndex: '81.3',
  }
],
isAuth: true,
};

const SET_LANGUAGE = 'SET_LANGUAGE',
      GET_ALL_DATA = 'GET_ALL_DATA',
      SET_AUTH = 'SET_AUTH',
      SET_NEW_ENTITY_DATA = 'SET_NEW_ENTITY_DATA',
      SET_UPDATED_ENTITY_DATA = 'SET_UPDATED_ENTITY_DATA',
      DELETE_ENTITY_DATA = 'DELETE_ENTITY_DATA';

const generalReducer = (state = initialState, action) => {
  switch (action.type) {
    case SET_LANGUAGE: {
      return {
        ...state,
        ...action.data
      };
    }
    case GET_ALL_DATA: {
      return {
        ...state,
        ...action.data
      };
    }
    case SET_NEW_ENTITY_DATA: {
      return {
        ...state,
        [action.data.entityName]: [...state[action.data.entityName], action.data.entity]
      };
    }
    case SET_UPDATED_ENTITY_DATA: {
      return {
        ...state,
        [action.data.entityName]: state[action.data.entityName].map(entry => {
          if (entry._id === action.data.entity._id)
            entry = action.data.entity;
          return entry;
        })
      };
    }
    case DELETE_ENTITY_DATA: {
      return {
        ...state,
        [action.data.entityName]: state[action.data.entityName].filter(entry => entry._id !==
action.data.id)
      };
    }
  }
};

```

```

    }
    case SET_AUTH: {
      return {
        ...state,
        isAuth: true,
      };
    }
    default:
      return state;
  }
};

export const setLanguage = (language) => {
  return { type: SET_LANGUAGE, data: { language } };
};

export const setAllData = (data) => {
  return { type: GET_ALL_DATA, data };
};

export const setNewEntityData = (data) => {
  return { type: SET_NEW_ENTITY_DATA, data };
};

export const setUpdatedEntityData = (data) => {
  return { type: SET_UPDATED_ENTITY_DATA, data };
};

export const deleteEntityData = (data) => {
  return { type: DELETE_ENTITY_DATA, data };
};

export const setAuthSuccess = () => {
  return { type: SET_AUTH };
};

export const getAllDataThunk = () => async (dispatch) => {
  const resp = await getAllData();
  dispatch(setAllData(resp));
};

export const createNewEntityThunk = (entity, entityName) => async (dispatch) => {
  const resp = await createNewEntity(entity, entityName);

  dispatch(setNewEntityData({entity: resp, entityName}));
};

export const updateEntityThunk = (id, entity, entityName) => async (dispatch) => {
  const resp = await updateEntity(id, entity, entityName);
  if (resp?.modifiedCount)
    dispatch(setUpdatedEntityData({
      entity: resp?.entity,
      entityName: resp?.entityName,
    }));
};

export const deleteEntityThunk = (id, entityName) => async (dispatch) => {
  const resp = await deleteEntity(id, entityName);
  if (resp?.modifiedCount)
    dispatch(deleteEntityData({ id, entityName }));
};

export default generalReducer;
Файл reduxStore.js
import {applyMiddleware, combineReducers, createStore} from 'redux';
import generalReducer from './generalReducer';
import ReduxThunk from "redux-thunk";

const reducers = combineReducers({
  general: generalReducer,
});

const store = createStore(reducers, applyMiddleware(ReduxThunk));

```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Слайди презентації на рисунках Б.1 – Б.14

Міністерство освіти і науки України
Національний університет "Запорізька політехніка"
Кафедра програмних засобів

ПРЕЗЕНТАЦІЯ ДО ДИПЛОМНОЇ РОБОТИ**РОЗРОБКА ВЕБСАЙТУ ДЛЯ ЗАЛУЧЕННЯ АБІТУРІЄНТІВ**

Виконав: студ. гр КНТ-118
Шевердінкін І. А.

Керівник:
Каплієнко Т.І., к.т.н, доцент

Рисунок Б.1 – 1 слайд презентації

Постановка задачі

Об'єкт дослідження – проведення інформаційної кампанії в інтернеті для залучення абітурієнтів для вступу до закладу вищої освіти.

Предмет дослідження – розробка вебсайтів для залучення абітурієнтів.

Мета роботи – створення вебсайту для залучення абітурієнтів, наповненого актуальною та корисною інформацією у зручній формі.

Рисунок Б.2 – 2 слайд презентації

Структура сайту

Сайт складається з двох частин: back-end, front-end.

Back-end – частина, яка взаємодіє з базою даних та іншими сторонніми бібліотека для виконання задач не пов'язаних з зовнішньою складовою сайту.

Front-end – частина, яка генерує сторінки, обробляє усі дії на сторінках та робить запити до back-end.

Зовнішній вигляд сайту складається з 3 частин: верхня частина, її називають “header” англійською, основна частина з інформацією, та нижня частина – “footer” англійською.

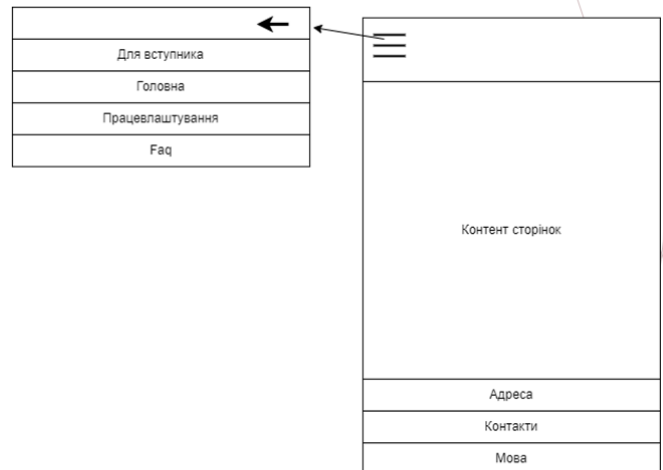


5

Рисунок Б.5 – 5 слайд презентації

Структура сайту

Окрім стандартного інтерфейсу потрібно було продумати як сайт буде виглядати на мобільних пристроях, адже через менший розмір екрану такі елементи як header та footer не могли залишитися у тому самому вигляді. Основним рішенням цієї проблеми для header стало створення меню, яке часто називають “бургер” меню. Для footer все вирішилося трохи простіше, адже через не дуже великий обсяг тексту потрібно було лише розділити текст на декілька ліній.

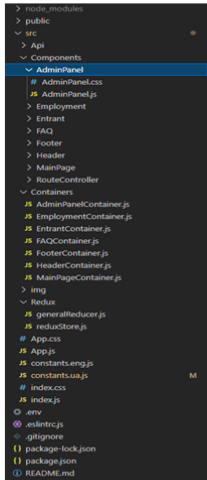


6

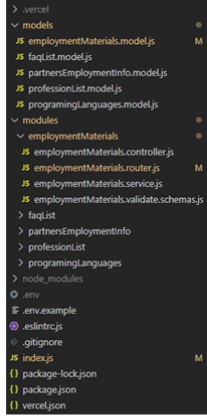
Рисунок Б.6 – 6 слайд презентації

Структура проєкту

Front-end



Back-end



У корені проєкту front-end є тека `src` з усіма компонентами, контейнерами тощо, `public` з початковим файлом `html` та картинками для вікна та файли налаштувань такі як `.env`, `package.json` `eslinttrc.js` тощо.

У корені є 4 теки, 2 з них згенеровані автоматично, це `.vercel` та `node_modules`. Тека `.vercel` створена однойменною бібліотекою, завдяки якій стало можливим викласти серверну частину на хостинг, а `node_modules` – це тека, у якій знаходяться усі бібліотеки та модулі, які використовує застосунок.

7

Рисунок Б.7 – 7 слайд презентації

Схема даних БД

Кожна таблиця у БД має поля `createdAt`, `updatedAt`, `deletedAt`, `id`. Усі ці поля, окрім `deletedAt`, є обов'язковими для створення запису та мають значення за замовчуванням.

Таблиця EmploymentMaterials	Таблиця PartnersEmploymentInfo	Таблиця ProfessionList	Таблиця ProgrammingLanguages	Таблиця FaqList
Додаткові поля:	Додаткові поля	Додаткові поля:	Додаткові поля:	Додаткові поля:
- info	- name	- name	- name	- question
- infoEn	- link	- nameEn	- writeNowPerc	- questionEn
- link	- headerText	- requirements	- changes	- answer
	- mainText	- requirementsEn	- writeNow	- answerEn
	- headerTextEn	- Link	- useSec	
	- mainTextEn	- averagePayment	- usePrim	
		- maxPayment	- likeIndex	
		- vacanciesNumber		

8

Рисунок Б.8 – 8 слайд презентації

Вибір мови програмування

Найпопулярнішими мовами програмування для розробки вебсайтів та серверних застосунків є JS, TS та PHP. PHP є більш старою та стає менш популярною щороку, навідміну від JS або TS, які стають популярнішими. Особливо значну популярність отримав TS, який є нащадком JS з явною типізацією. Мовою програмування для проекту було обрано JS.

Назва	Переваги	Недоліки
JS	1. Відсутність явної типізації прискорює розробку. 2. Мова має велику кількість додаткових бібліотек та вбудованих функцій.	1. Відсутність явної типізації ускладнює розробку великих проєктів через неоднозначність об'єктів що може повертати функція та типів даних елементів об'єкту
TS	1. Присутність явної типізації робить розробку великих проєктів зручнішою 2. Мова має велику кількість додаткових бібліотек, вбудованих функцій та інструментів	1. Присутність явної типізації ускладнює та робить довгою початкову стадію розробку через необхідність створення інтерфейсів.
PHP	1. Мова не конфліктує з розміткою html, що робить можливим використання php частини одразу під час розмітки. 2. Мова є досить простою, завдяки чому використовувати її може будь-хто	1. Мова стає менш популярною через більш зручних конкурентів 2. Додатки написані мовою PHP є менш захищеними ніж додатки написані іншими мовами

9

Рисунок Б.9 – 9 слайд презентації

Використані бібліотеки

React

React це бібліотека, завдяки якій було розроблено всю front-end частину. Бібліотека малює усі сторінки сайту, відправляє запити до серверної частини.

```
const FAQ = (props) => {
  const phrases = props.language === 1 ? translateEng : translateUkr;
  return (
    <Paper elevation={10} className='paper'>
      <Typography align='center' variant='h5'>{phrases['FAQ-TITLE']}</Typography>
      <Grid container>
        {props.faqList.map(faq => {
          return (
            <Grid item xs={12} md={6} className='faqElement' key={faq.question + faq.id}>
              <AccordionSummary
                expandIcon={ExpandMoreIcon} />
              <Typography>{faq.question}</Typography>
              <AccordionDetails>
                <Typography>{faq.answer}</Typography>
              </AccordionDetails>
            </Grid>
          )
        })}
      </Grid>
    </Paper>
  );
};
```

Express

Express – фреймворк на базі NodeJS, який є основою серверної частини. Його задачею є створення застосунку, визначення ендпоінтів, приймання запитів та відправка відповідей.

```
const express = require('express');
const app = express();

app.use((req, res, next) => {
  console.log(req.body, req.url, req.method);
  if (req.url === '/test') next();
});

const authToken = req.headers.authorization;
if (!authToken) {
  throw new Error('Unauthorized');
}

app.use((req, res, next) => {
  console.log('Authenticated');
  next();
});

app.use('/programming-languages', programmingLanguagesRouter);
app.use('/partnership-info', partnershipInfoRouter);
app.use('/partnership-info', partnershipInfoRouter);
app.use('/partnership-info', partnershipInfoRouter);
app.use('/partnership-info', partnershipInfoRouter);
app.use('/partnership-info', partnershipInfoRouter);

app.use(function (err, req, res, next) {
  console.log(err);
  res.status(500).send({ status: 'Internal Server Error', ...err });
});
```

Mongoose

Mongoose – бібліотека для взаємодії з БД. Оскільки обраною СКБД стала MongoDB то Mongoose є незамінною бібліотекою для взаємодії з таблицями.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const userSchema = new Schema({
  type: String,
  require: true,
  default: new Date(Date.now()),
});

const userSchema = new Schema({
  type: String,
  require: true,
  default: new Date(Date.now()),
});

const userSchema = new Schema({
  type: String,
  require: true,
  default: new Date(Date.now()),
});

const userSchema = new Schema({
  type: String,
  require: true,
  default: new Date(Date.now()),
});

const userSchema = new Schema({
  type: String,
  require: true,
  default: new Date(Date.now()),
});

module.exports = mongoose.model('user', userSchema);
```

10

Рисунок Б.10 – 10 слайд презентації

Інтерфейс та екранні форми

Сторінки

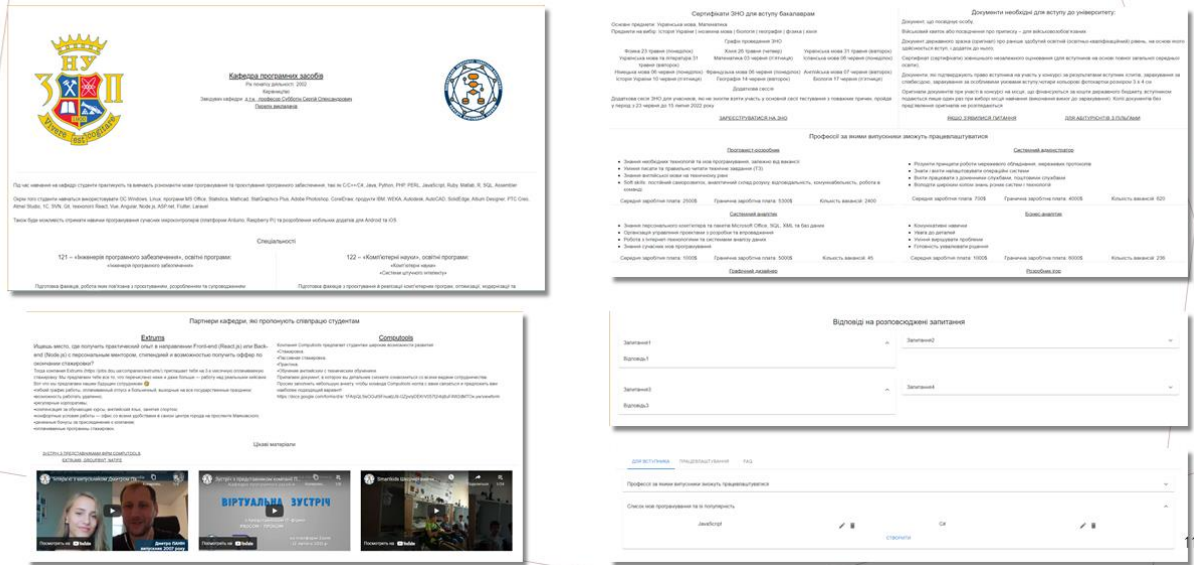


Рисунок Б.11 – 11 слайд презентації

Інтерфейс та екранні форми

Форми

Створити новий об'єкт professionList

Назва

test

Назва англійською

test

Вимоги

test

Вимоги англійською

test

Посилання

test

Середня заробітна плата

123

Максимальна заробітна плата

123

Кількість вакансій

123

СТВОРИТИ

СКАСУВАТИ

Оновити об'єкт professionList

Назва

Програміст-розробник

Назва англійською

Developer

Вимоги

Знання необхідних технологій та мов програмування, залежно від вакансії
Умінь писати та правильно читати технічне завдання (ТЗ)
Знання англійської мови на технічному рівні
Soft skills: постійний саморозвиток, аналітичний склад розуму, відповідальність, комунікабельність, робота в команді

Вимоги англійською

Knowledge of required technologies and programming languages, depending on the vacancy
Ability to write and correctly read the technical task (TOR)
Knowledge of English at the technical level
Soft skills: constant self-development, analytical mind, responsibility, sociability, teamwork

Посилання

https://www.work.ua/jobs-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BD%D0%B8%D0%BA/

Середня заробітна плата

2500\$

Максимальна заробітна плата

5300\$

Кількість вакансій

2400

Ви впевнені, що хочете видалити ?

ВИДАЛИТИ

СКАСУВАТИ

Рисунок Б.12 – 12 слайд презентації

Висновки

У результаті виконання дипломної роботи було проаналізовано об'єкт та предмет дослідження, а саме проведення інформаційної компанії в інтернеті для залучення абітурієнтів для вступу до закладу вищої освіти та розробка вебсайтів для залучення абітурієнтів. Було проаналізовано існуючі рішення, якими стали сайти Національного університету “Запорізька політехніка”, а саме основний сайт університету, сайт приймальної комісії та сайт кафедри програмних засобів.

Було проаналізовано популярні мови програмування для розробки сайтів та обрано мову для розробки проекту, нею стала мова програмування JavaScript. Також було проаналізовано різні бази даних та обрано ту, яка підходила більше, а саме MongoDB.

Було наведено схеми бази даних, описано алгоритми, які використовувались під час розробки.

Розроблений сайт можна використовувати як засіб для реклами університету, кафедри та для залучення нових студентів до вступу.

Рисунок Б.13 – 13 слайд презентації