КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

ДОПОМІЖНИЙ ДОДАТОК ДЛЯ HR СПЕЦІАЛІСТІВ

Виконав студент 4-го курсу
Капленко Олександр Юрійович

(підпис)

Науковий керівник:
асистент, кандидат фіз.-мат. наук
Федорова Марія Вікторівна

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«1» червня 2022 р.,

протокол № 10
Завідувач кафедри
М. С. Нікітченко

(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 41 сторінка, 42 ілюстрацій, 0 таблиць, 24 джерел посилань.

ІНСТРУКЦІЯ КОРИСТУВАЧА, ІНТЕРФЕЙС ПРОГРАМИ, ОДНОСТОРІНКОВИЙ WEB-ДОДАТОК, УПРАВЛІННЯ БАЗАМИ ДАНИХ, ANGULAR, ASP.NET CORE, HR-СПЕЦІАЛІСТ, MICROSOFT SQL SERVER.

Об’єктом розроблення допоміжного додатку для HR-спеціалістів є процеси зчитування інформації, її обробка та управління базами даних за допомогою програмного засобу.

Метою кваліфікаційної роботи є створення допоміжного технологічного додатку для HR-спеціалістів.

Методи розроблення. В якості базового інструменту створення програмного засобу було обрано комбінацію технологій ASP.NET Core (виступає у ролі серверної частини) та Angular (за допомогою якої реалізована клієнтська частина проєкту). У процесі реалізації розробки залучено використання інтегрованого середовища Visual Studio 2022, технологію MS SQL Server, пакет Selenium, платформу PdfTron, технологію Blob сховища Azure, систему Syncfusion Schedule, сервіс MatDialog, а також застосовано Web Animations, Toastr, Azure Active Directory B2C.

Результати роботи: визначено план роботи над продуктом та обрано сферу його функціонування; поглиблено знання з технологій розробки додатків та визначено ті з них, які будуть застосовані у роботі; систематизовано вимоги до функціонування відповідного додатку та втілена його практична реалізація; здійснено опис структурних частин програми, презентовано його інтерфейс; для забезпечення успішного функціонування подано інструкцію користувача.

Програмний продукт створено безпосередньо для потреб застосування HR-спеціалістами, перед якими сьогодні постає гостра вимога відмінного розуміння сучасних HR-практик та володіння різними інструментами (зокрема, й IT-технологіями) для побудови ефективних процесів.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта розробки. Сучасні практики повсякдення HR-спеціаліста набувають тісного взаємозв'язку із IT-технологіями. Традиційні схеми та наративи трансформуються у новітні форми спілкування із «людським ресурсом». Переважна більшість компаній будує свій бізнес на основі сучасних технологій, даних та аналітики, що саме по собі вимагає мобільної обробки цих даних, їх аналітичного й швидкого опрацювання.

Усі ці фактори спонукають до відповідних розробок зручних та практичних технологічних додатків. Станом на сьогодні можна говорити про активний процес нарощення кількісних та якісних показників у цій справі. Конкуренція враховує як суто функціональні, так і естетичні (дизайнерські) характеристики відповідних IT-розробок.

Можна навіть говорити про спробу внутрішньої класифікації пропонованих інструментів для автоматизації HR-процесів [1]. Так, великим попитом користуються платформи для тестування та оцінки навичок (HackerRank [2], Pymetrics [3], Self Management Group [4]); програми, які використовують штучний інтелект (Ideal [5], Textio [6], Zoom.ai); системи відстеження кандидатів (Bullhorn [7], Greenhouse [8], SAP SuccessFactors [9], iCIMS [10], Jobvite [11], Workable [12]); програмне забезпечення для відео-інтерв'ю та онлайн-співбесід (ConveyIQ [13], Wepow [14]); сервіси з управління персоналом (BambooHR, Zenefits [15], OpenTute [16], Ultimate Software) тощо.

Пошук нових проєктів активно стимулюється новими реаліями. Тяжіння до максимально зручних та універсальних форм продукує появу великої кількості гібридів, поєднання, різноманітних комбінацій програм. Відповідно, попит породжує нові пропозиції.

Актуальність роботи та підстави для її виконання. Фактор актуальності визначається широкою парадигмою вимог до популярної вакансії HR-спеціаліста в умовах сучасної ринкової ситуації. Перелік основних обов'язків відповідного фахівця у сфері HR вже не обмежується високими комунікативними здібностями чи навичками побудови ефективних ділових стосунків. Постає гостра вимога розробки та впровадження різноманітних мотиваційних програм залучення ключових спеціалістів; необхідність формування та підготовки кадрового резерву; відмінного розуміння сучасних HR-практик та володіння різними інструментами для побудови ефективних процесів.

Зрештою, потрібно усвідомлювати взаємооберненість процесу, адже перед фахівцями стоять вимоги розробки та реалізації програми найму й адаптації співробітників, з одного боку, та власної адаптації до мінливої ринкової ситуації, з іншого.

Відтак, продукування якісного та зручного технологічного додатку, який допоможе вирішити цілу низку проблемних моментів, є сьогодні на часі, чим і забезпечує актуальність нашої розробки.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення допоміжного додатку для HR-спеціалістів. Для досягнення цієї мети поставлено такі завдання.

- Визначити план роботи над продуктом.
- Поглибити знання з технологій розробки додатків.
- Розробити зручний користувацький інтерфейс додатку.
- Забезпечити інформаційне наповнення додатку та його успішне функціонування.
- Здійснити огляд використаних технологій.
- Описати структурні частини програми.
- Подати інструкцію користувача.

Об’єкт і методи розробки. Об’єктом розроблення допоміжного додатку для HR-спеціалістів є процеси зчитування інформації, її обробка та управління базами даних за допомогою програмного засобу.

В якості інструменту створення програмного засобу було обрано комбінацію технологій ASP.NET Core (виступає у ролі серверної частини) та Angular (за допомогою якої реалізована клієнтська частина проєкту).

Реалізація серверного блоку коду стала можливою завдяки використанню інтегрованого середовища розробки Visual Studio 2022 для автоматичного завершення коду, який розпізнає контекст; технології MS SQL Server для автоматичного пов’язування моделей нашої системи з таблицями у базі даних; пакета Selenium для зчитування інформації з сайту.

Реалізація клієнтської частини, що передбачає роботу з файлами, була здійснена шляхом використання платформи PdfTron для перегляду файлів; технології Blob сховища Azure для збереження даних; системи Syncfusion Schedule для календарного планування подій. Для забезпечення комфортної атмосфери та естетизації процесів було додано сервіс MatDialog, а також застосовано Web Animations, Toastr, Azure Active Directory B2C.

Можливі сфери застосування. Програмний продукт – допоміжний технологічний додаток – створено безпосередньо для потреб застосування HR-спеціалістами.

РОЗДІЛ 1 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Реалії сучасного світу вимагають створення швидкого, зручного та технологічного додатку. Він повинен бути не відірваним від всього, а гнучко реалізовувати взаємодію з вже існуючими системами. Також, система повинна створити щось особливе, щось нове. Саме це буде виокремлювати її з-поміж вже існуючих застосунків і привертати увагу клієнтів.

Щоб досягти цих амбіційних цілей, було обрано досить популярну і перевірену часом комбінацію технологій ASP.NET Core та Angular. Ми використовуємо її для створення односторінкового веб-додатку.

Чому саме ASP.NET Core? Ця структура проєктування дозволяє створювати додатки за допомогою різноманітних моделей програмування, серед яких ASP.NET Core Web API. Вона представляє реалізацію архітектури REST, згідно якої для кожного http-запиту призначений окремий ресурс. Подібні ресурси визначаються у вигляді методів контроллера Web API.

ASP.NET Core виступає у ролі серверної частини. Для написання цього блоку коду, ми використовували інтегроване середовище розробки Visual Studio 2022. Особливою перевагою цієї версії, для нас, став IntelliCode – потужний набір засобів автоматичного завершення коду, який розпізнає контекст [17]. Окрім цього, після оновлення стала доступна остання версія ASP.NET Core 6.0, яку і було вирішено взяти як основу нашого проекту.

Entity Framework Core це ORM технологія, що дозволяє автоматично пов’язувати моделі нашої системи з таблицями у базі даних. Вона підтримує більшість популярних систем управління базами даних. Ми вирішили використати MS SQL Server, адже він входить до трійки найпопулярніших на ринку систем (рисунок 1) і дуже зручний у використанні.

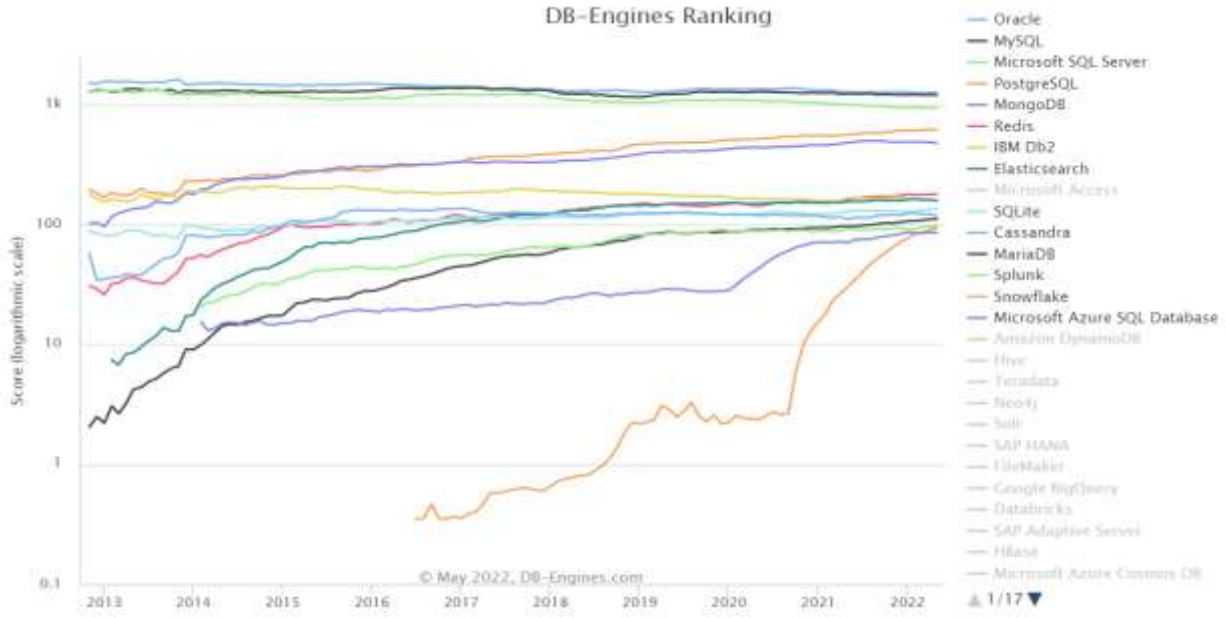

Рисунок 1 – Графік популярності систем управління базами даних

Однією з преваг нашого проекту є зчитування інформації з сайту LinkedIn. Це соціальна мережа, яка надає можливість користувачам створювати, підтримувати та розширювати список ділових контактів [18]. Однією з проблем на шляху реалізації цієї функції став екран завантаження сайту. Після ретельного пошуку шляхів вирішення цієї задачі, ми вирішили скористатися пакетом Selenium. З його допомогою можна дочекатися повного завантаження сторінки і вже потім зчитувати вміст. Крім того, несподіваним плюсом під час роботи з цим інструментом стала можливість використовувати різноманітні способи пошуку елемента на сторінці.

Клієнтська частина проєкту реалізована за допомогою Angular. Зручна зрозуміла структура додатку і наявність якісної документації. Саме це значно вплинуло на вибір технології. Адже для клієнтської частини обирався новий засіб, з яким раніше не доводилося працювати.

Angular проєкт складається блоків. Їх досить легко генерувати і пов’язувати між собою. Це дозволяє оновлювати лише ті елементи, які дійсно цього потребують, а не всю сторінку одразу. Також один елемент можна використовувати

декілька разів, що значно спрощує написання коду і допомагає уникнути дублювання.

Клієнтська частина передбачає роботу з файлами. Для перегляду файлів було обрано платформу PdfTron. Вона зручна, швидка і надійна. З її допомогою легко додавати нові функції для роботи з файлом, тож у результаті ми отримуємо не просте відображення файлу, а повноцінний самостійний інструмент.

Крім перегляду файлів наша програма дозволяє їх зберігати. Вони завантажуються до Blob сховища Azure. Це технологія компанії Microsoft для збереження великих об’ємів неструктурованих даних [19].

Також, додаток містить сторінку для планування. Загалом, це календар подій, який керує списком різноманітних заходів у різних доступних режимах перегляду. Ця функція реалізована за допомогою Syncfusion Schedule [20].

Інтерфейс став важливим елементом системи. Тому для роботи з ним було імпортовано значну кількість засобів.

Перш за все, додано сервіс MatDialog, який можна використовувати для відкриття модальних діалогів зі стилями та анімацією Material Design. Оскільки кількість тем була обмежена, то ми прийняли рішення обрати найбільш оптимальну і перевизначити частину її елементів.

Для того, щоб користувач не почував себе покинутим системою, потрібно сповіщати його про успішність або хибність його дій. Кожна вагома подія, запит та відповідь супроводжуються появою відповідного елементу на сторінці. Ці елементи реалізовані за допомогою Web Animations та структури Toastr.

Авторизація та автентифікація реалізовані за допомогою Azure Active Directory B2C. Це рішення для керування доступом до ідентифікації клієнтів (CIAM), яке здатне підтримувати мільйони користувачів і мільярди аутентифікацій на день. [21]

Технологія B2C дозволяє задіяти сторонні API типу Google та Facebook. Що ми успішно реалізували у програмі. Також було налаштовано додаткові параметри користувача, зміну пароля, надсилання кодів підтвердження електронною поштою та ще багато чого іншого.

РОЗДІЛ 2 ОПИС СТРУКТУРНИХ ЧАСТИН ПРОГРАМИ

2.1 Вимоги до програми

Вимоги до функціонування програми. Програма повинна реалізовувати авторизацію та автентифікацію користувача. Користувач повинен мати змогу ввійти в систему за допомогою Google та Facebook акаунтів. Під час реєстрації юзер повинен отримати код підтвердження на пошту і ввести його в спеціальне поле, щоб активувати акаунт.

Необхідно, щоб користувач міг змінити пароль, якщо раптом не згадає поточний.

Після авторизації, користувач повинен мати змогу переглядати відомості про обліковий запис.

Система повинна містити логічні блоки, серед яких були виділені блок проєктів, технологій та кандидатів.

Проєкти. Структурна одиниця, що обмежує доступ стороннім користувачам. Символізує певне завдання, що необхідно виконати обмеженою групою працівників.

Додаток повинен надавати функцію створення проєкту. Також юзер повинен мати змогу отримати усі проєкти, до яких він має доступ і перейти до обраного з них.

Головна сторінка проєкту повинна містити його назву, інформацію про того, ким він був створений, короткий опис.

Технологія. Ще один логічний блок, що виступає в ролі розділу для проєкту.

Проєкт, повинен містити список технологій, можливість його розширення та фільтрації за назвою технології.

Наступним логічним блоком нашої системи стали кандидати. Вони уособлюють потенційних працівників компанії. Саме цей блок є основним. Весь додаток націлений на те, щоб робота з кандидатами була максимально простою та продуктивною.

Кожна технологія містить у собі таблицю кандидатів. Пошук кандидата повинен відбуватися за іменем, прізвищем та поштою.

Список кандидатів повинен доповнюватися. Необхідно розробити функцію яка буде зчитувати інформацію з обраної LinkedIn сторінки і заповнювати потрібні поля нового кандидата. Якщо сторінка містить більше інформації, ніж обов'язковий мінімум, то повинні створюватися і заповнюватися додаткові поля.

Також, наша система повинна зберігати файли, та пов'язувати їх з певним кандидатом. Необхідно реалізувати таке сховище, щоб документ був доступний за посиланням поза нашою системою.

Файли можуть мати різні формати. Необхідно забезпечити коректне їх відображення і збереження. Попередній перегляд файлу обов'язкова умова для його завантаження до системи.

Крім того, учасники проєкту повинні мати змогу запрошувати своїх колег.

Кожен проєкт повинен мати розклад – список подій доступний усім його працівникам. Для зручності використання кожен юзер також повинен мати свій власний розклад, доступний лише йому.

Крім того, повинна бути реалізована можливість для створення Zoom конференції, отримання посилання власника та посилання гостя.

Для взаємодії зі сторонніми додатками система буде запитувати дозволи. Дозвіл повинен бути прицільним, тобто якщо ми хочемо створити Zoom конференцію, то дозвіл має обмежуватися лише цією функцією.

Додаток повинен містити темну та світлу тему. Кольори теми за замовчуванням досить світлі, тому можуть спричинити дискомфорт, якщо працювати у погано освітленому приміщені.

Система повинна коректно відображатися на різних пристроях. Необхідно створити інтерфейс для комп'ютерів, планшетів та телефонів. Повний функціонал додатку повинен зберігатися на всіх пристроях.

Технічні вимоги. Програма повинна запускатись на операційній системі Windows, починаючи з версії 7, до останньої 11 версії. Також, інформація повинна коректно відображатися у різних браузерах, серед яких Google Chrome.

2.2 Інтерфейс програми

Інтерфейс одна з найважливіших частин програми. Його мова – англійська, ми уникнули використання складних слів та сталих виразів. Ми поглянули на систему з точки зору користувача. Будь-який нетривіальний елемент має пояснення щодо його функцій та способу використання.

Додаток має просту структуру. Навігація по сайту реалізована у вигляді меню. Воно доступне на всіх сторінках додатку та містить мінімалістичні позначення сторінок на які ми можемо потрапити, серед яких:

- домашня сторінка – Home
- сторінка з інформацією про авторизованого користувача – Profile
- сторінка для перегляду поточних проєктів – Project
- сторінка для роботи з файлами – File
- сторінка розкладу – Schedule
- сторінка для проведення опитувань – Poll

Поточна сторінка позначається у меню кольором та маркером, крім того всі вони мають певну стилістичну особливість – унікальний колір певних елементів. Це зроблено для того щоб користувач міг краще орієнтуватися, бігаючи по сторінках нашого додатку.

Загалом, для того щоб користувач зручно почував себе на сайті, ми здійснили декілька важливих кроків.

Інтерактивні елементи виконані у одному стилі. Кожен тип інтерактивних елементів має свої унікальні візуальні особливості, тож користувач навіть під час першого входу на сайт зможе неймовірно швидко перейти до роботи, а не до спроб розібратися як працювати з додатком.

Сторінки, на нашу думку, пов’язані саме так як буде очікувати і потребувати того користувач.

Розташування елементів на сторінці. Досить кропітку роботу ми здійснили для найбільш зручного та інтуїтивно зрозумілого місця розташування кожного

елементу. Виникли певні проблеми з розташуванням об’єктів на смартфоні, адже принципово необхідно зробити так, щоб додаток мав однакову структуру кожної сторінки на всіх пристроях.

Ще однією особливістю є те, що наша система запобігає випадковій втраті інформації. Якщо користувач заповнював форму і випадково виконує дію, яка призведе втрати даних, то система запитає чи він дійсно цього хотів.

Також варто зазначити, що наша технологія досить доступна для кожного. Вона безкоштовна і не потребує великої кількості інформації, щоб зареєструватися. На диво, це дуже вигідно виокремлює систему з-поміж інших. Адже нам досить важко було відшукати аналоги з такими параметрами.

2.3 Структура програми

Структура нашої програми складається з трьох блоків (рисунок 2).

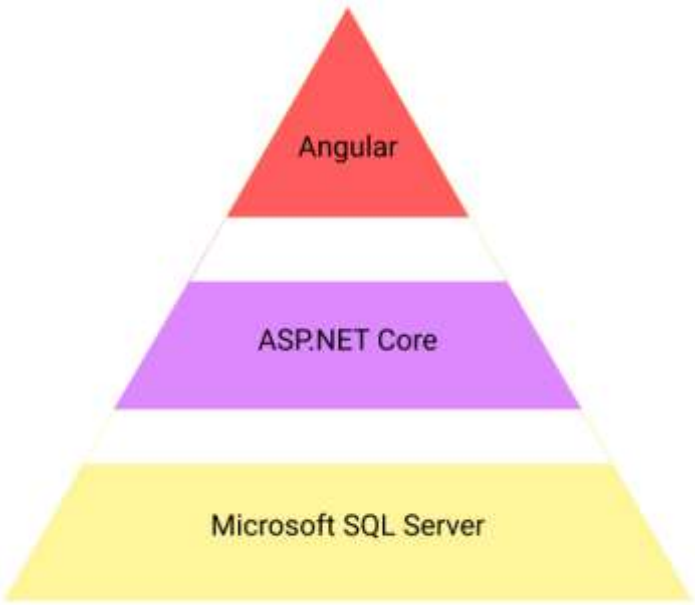

Рисунок 2 – Структура програми

2.3.1 Microsoft SQL Server

У нашій програмі було вирішено використовувати Code First сценарій програмування. Цей підхід дозволяє спочатку визначити наші моделі за допомогою класів мови програмування C#, а вже потім по ним створити базу даних з таблицями. Це дозволяє легше доповнювати таблиці новими полями у разі потреби. У результаті ми отримали базу даних зображену на рисунку 3.

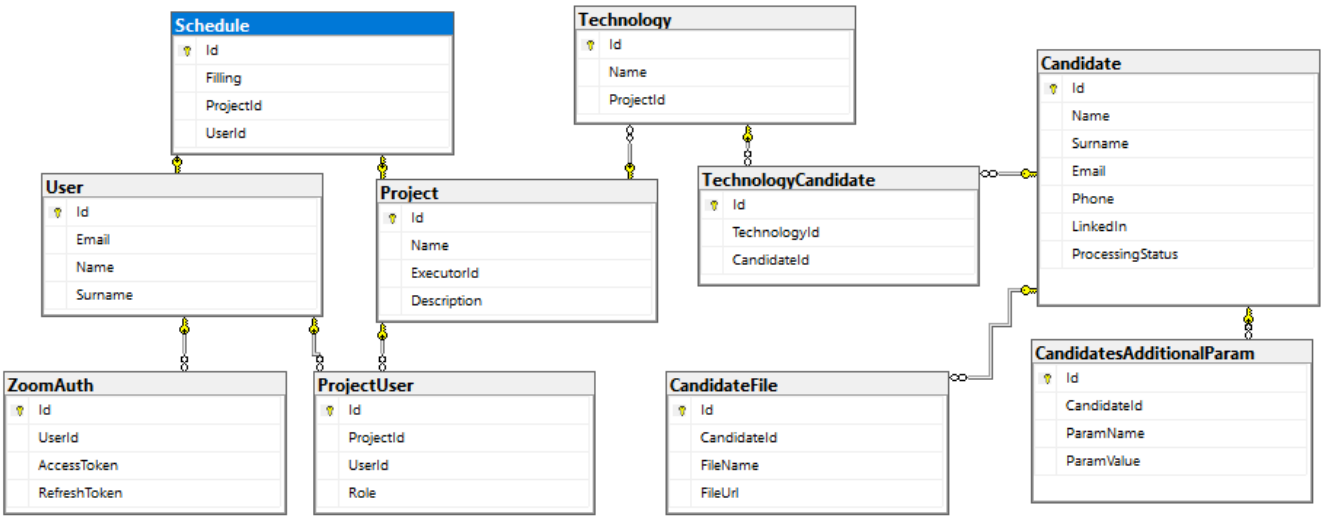

Рисунок 3 – База даних

2.3.2 ASP.NET Core

Використання шаблону програмування ASP.NET Web API передбачає створення моделей. Крім того, необхідно визначити контекст, який походить від System.Data.Entity.DbContext та надає введений DbSet<TEntity> для кожного класу в нашій моделі. Це дозволить нам робити запити та зберігати дані. [22]

Крім того, наша програма реалізує контроллери. Це елементи, які приймають запит, займаються його обробкою, за потреби звертаються до моделей та повертають відповідь. У нашому випадку методи контроллери реалізують REST архітектуру.

Також важливою структурною одиницею є сервіси. Вони дозволяють взаємодіяти нашому додатку зі сторонніми API. Наприклад одним з таких став додаток Zoom.

2.3.3 Angular

Структура клієнтьскої частини досить проста. Розробники Angular попіклувалися, щоб будь-який додаток мав достатньо чітке розділення на блоки. Взагалі, існує близько двадцяти типів блоків. Їх функціонал різнаманітний. У нашому проєкті використано чотири з них.

Блок component. Він може виступати у ролі частини або навіть цілої сторінки. Тобто на екрані зображено різномаїтна комбінація блоків компонент, у нашому проєкті лише цей тип блоків видно кінцевому користувачу, тому вся візуальна частина реалізована саме в них.

Цей блок складається з файлів:

- Каскадної таблиці стилів. Відповідає за розміження та візуальний вигляд елементів;
- Гіпертекстової розмітки сторінки. Відповідає за наповнення сторінки контекстом. Елементи саме цієї сторінки стилізуються каскадними таблицями стилів.
- Spec. Ці файли мають розширення .ts та використовуються для тестування нашого додатку.
- Тайпскрипт. Використовується для обробки подій, які користувач виконує на своїй сторінці.

Блок service. Наш проєкт використовує його для надсилання http запитів на сервер. Кожен цей блок пов’язано з відповідним контроллером з частини додатку, яка реалізована за допомогою ASP.NET Core Web API. Тобто кожен сервіс спілкується лише зі своїм контроллером. Їх спілкування означає, що метод з service надсилає запит, потім серверна частина його обробляє та надсилає відповідь.

Блок model. У цих блоках реалізовані інтерфейси та класи клієнтського додатку. Наш проєкт їх використовує щоб строго визначити параметри, які ми відправляємо у http запиті та витягуємо з відповіді.

Блок pipe. У нашому додатку використовується сортування. Воно реалізується саме в цих блоках. Тут ми описуємо як буде відбуватися сортування, що воно повинно повертати та інформація яка використовується для сортування.

2.4 Реалізація програми

2.4.1 ASP.NET Core Web API

Перш за все необхідно створити моделі нашої серверної частини. Адже саме за ними генерується база даних, саме інформацію про них необхідно заповнити в контексті, щоб саме з ними будуть працювати майбутні контроллери.

Тому переходимо до створення першої моделі. Вона обов’язково повинна містити ключове поле. За ним буде відбуватися ідентифікація її об’єктів, адже навіть сама база даних не дозволить нам внести два обєкти з однаковими значеннями ключових полів.

Далі створюємо сові поля необхідні для реалізації функціоналу додатку. Вони можуть бути обов’язковими (Required). Це означає, що об'єкт цієї моделі не допустить пропуску в заповнені цього поля.

Таблиці у базі даних можуть мати різний зв’язок. Це також реалізується у класах моделей нашого додатку. Вього використано три типи зв’язків.

Один до багатьох. Означає що об’єкту однієї моделі (головної) може ставиться у відповідність декільнка об’єктів іншої (залежної).

Головна модель повинна містити список елементів залежної. Приклад реалізації зображено на рисунку 4.

```
8 references
public class Candidate
{
    1 reference
    public Candidate()
    {
        CandidatesAdditionalParam = new List<CandidatesAdditionalParam>();
    }
    [Key]
    11 references
    public int Id { get; set; }
    3 references
    public string Name { get; set; }
    3 references
    public string Surname { get; set; }
    3 references
    public string Email { get; set; }
    3 references
    public string Phone { get; set; }
    3 references
    public string LinkedIn { get; set; }
    1 reference
    public virtual ICollection<CandidatesAdditionalParam> CandidatesAdditionalParam { get; set; }
```

Рисунок 4 – Клас головної моделі

Тоді як залежна модель повинна містити поле з ключовим параметром головної та доступ до її полів. Приклад реалізації зображено на рисунку 5.

```
7 references
public class CandidatesAdditionalParam
{
    [Key]
    5 references
    public int Id { get; set; }
    [Required]
    2 references
    public int CandidateId { get; set; }
    [Required]
    3 references
    public string ParamName { get; set; }
    [Required]
    3 references
    public string ParamValue { get; set; }

    0 references
    public virtual Candidate? Candidate { get; set; }
}
```

Рисунок 5 – Клас залежної моделі

Також наша система включає реалізацію зв’язку багато до багатьох. Для цього необхідно створити додатковий клас моделі (рисунок 6), який буде містити ідентифікатори тих моделей, які ми хочемо об’єднати таким типом зв’язку. Після

цього новоутворена модель реалізовує зв’язок один до багатьох з обраними моделями.

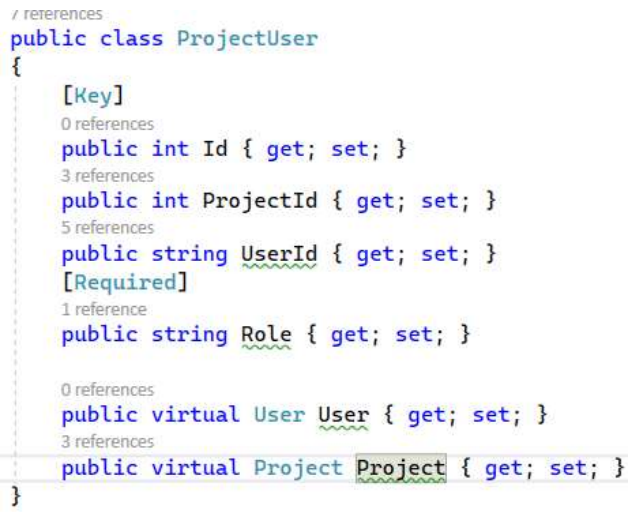

Рисунок 6 – Приклад реалізації таблиці зв’язку

Один до одного – останній тип зв’язку реалізований у додатку. Особливість його реалізації полягає у тому, що він не містить головної таблиці. Обидві таблиці – залежні.

Тепер реалізуємо контекст. У ньому перелічено всі створені класи моделей нашого додатку. Більшість з них має свій контроллер, який реалізує методи для роботи з моделлю.

Зараз можемо розглянути, як саме реалізуються деякі функції нашої системи.

Перейдемо одразу до методу, що дозволяє зчитувати інформацію зі сторінок сайту LinkedIn.

Складність цієї задачі полягала у тому, що додаток LinkedIn містить екран завантаження сторінки. І якщо не використовувати додаткові бібліотеки, то зчитати ми зможемо тільки його. Тому було обрано і підключено пакет Selenium.

Крім того, LinkedIn не впускає переглядати сторінки неавторизованим користувачам. Тому система повинна передавати файли cookie вже існуючого юзера їх додатку. Для цього ми зареєстрували сторінку нашої системи у додатку LinkedIn та скопіювали значення поля li_at (рисунок 7) у браузері.

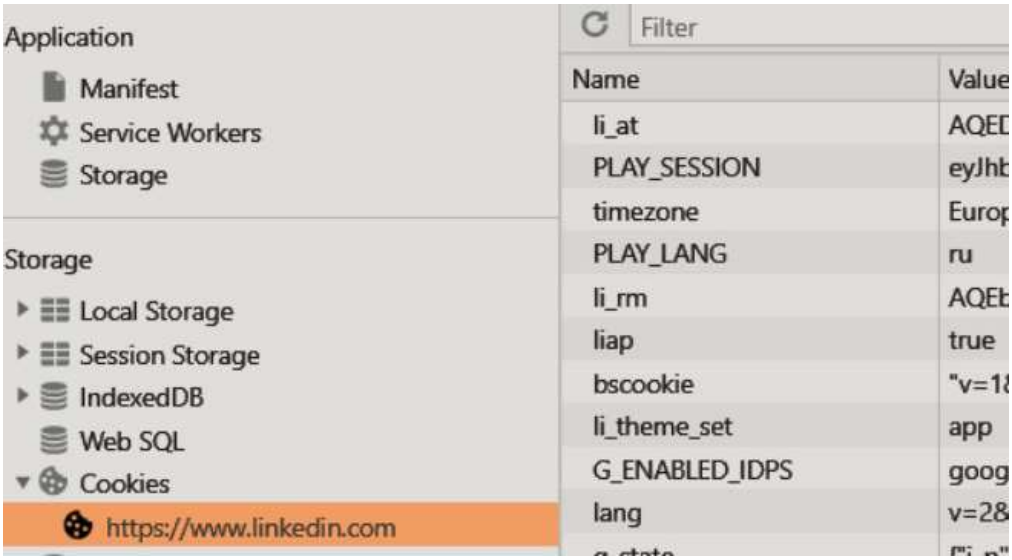

Рисунок 7 – Розміщення поля li_at у браузері

Підготовчі кроки виконано, тому ми вже можемо реалізувати першу частину коду (рисунок 8), яка буде відкривати вікно браузера, авторизуватися від імені системи та переходити до потрібної сторінки профіля.

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
driver.Navigate().GoToUrl(uri);
driver.Manage().Cookies.AddCookie(new OpenQA.Selenium.Cookie("li_at", clientBody.li_at));
driver.Navigate().Refresh();
wait.Until(d => ((IJavaScriptExecutor)d).ExecuteScript("return document.readyState").Equals("complete"));
```

Рисунок 8 – Перша частина алгоритму

Переходимо безпосередньо до витягування тексту зі сторінки.

Якщо ми спробуємо витягнути весь текст з тегу html, то побачимо що чомусь весь тест повторюється 2 рази (рисунок 9). Спершу ми вирішили, що це проблема реалізації, але потім стало зрозуміло що це не так.

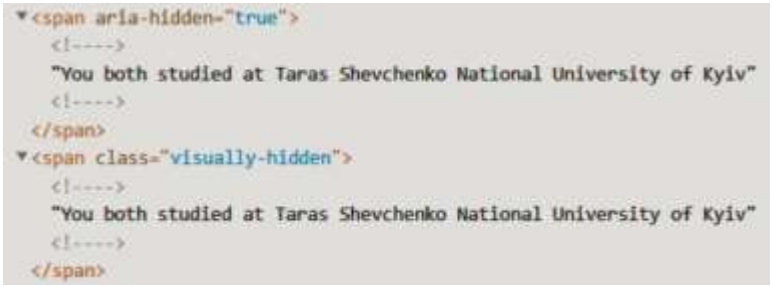

Рисунок 9 – Частина структури сторінки LinkedIn

Така проблема виникає не на всіх сторінках, також чому саме два рази не вдалося з’ясувати. Тому було прийняте рішення вносити всі стрічки до структури HashSet, яка не допускає повторів.

Наш додаток проводить певну обробку інформації. Тому від простого зчитування всього тексту сторінки користі досить мало. Нам потрібно брати інформацію з конкретних блоків.

Є декілька способів знайти блок у DOM структурі сторінки (рисунок 10).

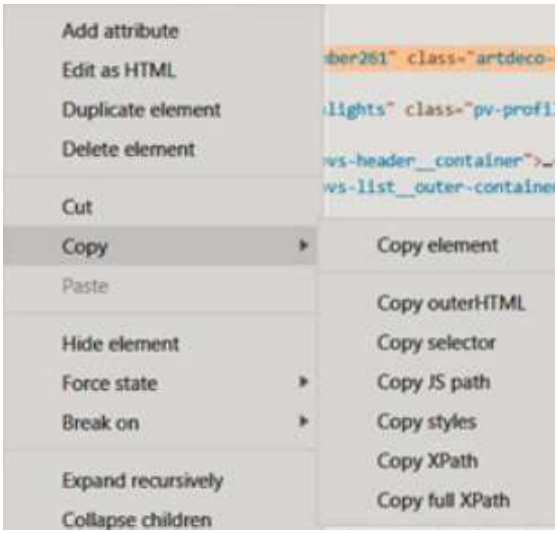

Рисунок 10 – Способи отримати шлях до елемента у структурі DOM

Майже усі ці способи використовують класи та ідентифікатори батьківських елементів. Це дуже добре та зручно, якщо вони залишаються незмінними. У нашому випадку це не так.

Ми визначили, що кількість параграфів, доступних для створення на сторінці користувачів LinkedIn не безмежна. І нам дуже пощастило, що кожен з них можна ідентифікувати на будь-якій сторінці користувача.

Тому було реалізовано зчитування кожного такого блоку за логікою зображеною на рисунку 11. Обробка інформації в ньому та внесення її до бази даних.

```
ReadOnlyCollection<IWebElement> experience = driver.FindElements(By.Id("experience"));
string experienceParent;
if (experience.Count() != 0)
{
    experienceParent = experience[0].FindElement(By.XPath("./..")).GetAttribute("id");
    ReadOnlyCollection<IWebElement> experience_li = driver.FindElements(By.XPath($"//*[@id='{experienceParent}']/div[3]/ul/li"));
    int i = 0;
    foreach (var li in experience_li)
    {
        ++i;
        HashSet<string> a = li.Text.Split("\r\n").ToHashSet();
        foreach (var lll in a)
        {
            var additionalParam = new CandidatesAdditionalParam();
            additionalParam.CandidateId = candidate.Id;
            additionalParam.ParamName = "experience_" + i;
            additionalParam.ParamValue = lll;

            await _context.CandidatesAdditionalParam.AddAsync(additionalParam);
            await _context.SaveChangesAsync();
        }
    }
}
```

Рисунок 11 – Реалізація витягування інформації з блоку Experience

Окрім LinkedIn наша система ще працює з іншими API. Для авторизації користувачів використовуються Google та Facebook. Також клієнт може створювати Zoom зустрічі. Для цього потрібно було створити акаунти розробника у цих додатках. Оскільки процеси ідентичні, то пробіжимося по алгоритму дій для одно х них.

Користувач кожної з систем має доступ до сторінки розробника. Створюємо додаток та заповнюємо обов’язкові поля. Звідси нам потрібні значення ключа та секрету додатку(рисунок 12).

Рисунок 12 – Ключ та секрет додатку

Ці значення ми маємо зберегти у файлах проєкту. Розберемо приклад створення конференцій у системі Zoom, використовуючи наш проєкт.

Перш за все, потрібно щоб клієнт надав нам дозвіл змінювати список його зустрічей (рисунок 13). Для цього і будуть використаніключ та секрет додатку.

```
[HttpGet]
0 references
public ActionResult SignIn() {
    return Redirect(
        "https://zoom.us/oauth/authorize?response_type=code&client_id=CLIENT_ID_VALUE&redirect_uri=https://localhost:7127/api/zoom/OAuthRedirect"
        );
}
```

Рисунок 13 – Приклад запиту дозволу

Після цього ми зможемо отримати ключ доступу за допомогою коду зображеному на рисунку 14.

```
[HttpGet]
public async Task<IActionResult> OAuthRedirect(string code) {
    var restClient = new RestClient($"https://zoom.us/oauth/token?grant_type=authorization_code&code={code}&redirect_uri=https://localhost:7127/api/zoom/OAuthRedirect");

    var request = new RestRequest($"https://zoom.us/oauth/token?grant_type=authorization_code&code={code}&redirect_uri=https://localhost:7127/api/zoom/OAuthRedirect", Method.Post);
    request.AddHeader("Authorization", string.Format(AuthorizationHeader));
    request.AddHeader("Content-Type", "application/x-www-form-urlencoded");

    RestResponse response = await restClient.ExecutePostAsync(request);

    Res resp = JsonSerializer.Deserialize<Res>(response.Content)!;
    ZoomAuth zoomAuth = new ZoomAuth();
    zoomAuth.AccessToken = resp.access_token;
    zoomAuth.RefreshToken = resp.refresh_token;

    await _context.ZoomAuth.AddAsync(zoomAuth);
    await _context.SaveChangesAsync();

    return Ok(zoomAuth.Id);
}
```

Рисунок 14 – Код отримання ключа доступу

Офіційна документація Zoom API пропустила ці кроки в описі процесу, тому ця частина роботи виконувалася за аналогією роботи з Google Drive API.

У фінальній версії нам довелось відмовитися від використання Drive API оскільки воно не надає вирішення проблеми відображення документів на сторінках сторонніх додатків. Були здійснені успішні спроби це обійти, але як виявилося потім, документи ми можемо отримати лише типу .docx та лише у html форматі.

В результаті наша система працює з Azure Blob storage. Клієнт може переглядати, додавати та видаляти елементи. З міркувань безпеки показувати налаштування цього кроку не доцільно. Наша система лише вносить зміни та зберігає посилання на ресурс. Також пов’язує кандидата й закріплені за ним файли (рисунок 15).

```
[HttpPut("{candidateId}")]
0 references
public async Task<IActionResult> PutCandidateFile(int? candidateId, [FromBody] CandidateFile file)
{
    if (candidateId == null)
        return NotFound();

    var _file = await _context.CandidateFile.FirstOrDefaultAsync(c => c.Id == file.Id);
    if (_file != null)
    {
        _file.FileName = file.FileName;
        _file.FileUrl = file.FileUrl;

        await _context.SaveChangesAsync();
    }

    var _fileGet = await _context.CandidateFile
        .Where(f => f.CandidateId == candidateId)
        .ToListAsync();

    return Ok(_fileGet);
}
```

Рисунок 15 – Встановлення зв’язку між файлом та кандидатом

Опис реалізація CRUD архітектури проєкту майже весь був пропущений, але це не означає що його пропустили в реалізації. Тому всі функції можна переглянути у додатку. Тут ми користалися допомогою специфікації OpenApi версії 3.0.3.

2.4.2 Angular

Перш за все необхідно було побудувати структуру сторінки. У нас вона складається з меню та контенту (рисунок 16).

```
<div class="container">
  <aside>
    <div class="top">
      <div class="logo">
        <img src="https://images.theconversation.com/files/399816/original/file-20210510-5797-xqoxsr.jpg?ixlib=rb-1.1.0&q=45&aut
        <h2>EGA<span class="danger">TOR</span></h2>
      </div>
      <div class="close" id="close-btn">
        <span class="material-icons-sharp">close</span>
      </div>
    </div>
    <div class="sidebar">
      <a [routerLink]="['/home']" routerLinkActive="active">
        <span class="material-icons-sharp">grid_view</span>
        <h3>Dashboard</h3>
      </a>
```

Рисунок 16 – Класи, стилі та перший елемент меню

Також необхідно було визначитися з кольорами, які будуть використані у проєкті. Вони винесені у окремий файл для зручного використання, а також підібрані їх аналоги для використання у реалізації темної теми (рисунок 17).

Рисунок 17 – Кольори, використані у проєкті

Можемо налаштувати реєстрацію за допомогою Azure AD B2C та інтегрувати її до проєкту.

Налаштовуємо поля та функції для користувача, та підключаємо їх у нашому додатку (рисунок 18).

```
export const b2cPolicies = {
    names: {
        signUpSignIn: "B2C_1_DipUserFlow",
        editProfile: "B2C_1_DipUserFlow_EDIT"
    },
    authorities: {
        signUpSignIn: {
            authority: "https://kaplenko.b2clogin.com/kaplenko.onmicrosoft.com/B2C_1_DipUserFlow",
        },
        editProfile: {
            authority: "https://kaplenko.b2clogin.com/kaplenko.onmicrosoft.com/B2C_1_DipUserFlow_EDIT"
        }
    },
    authorityDomain: "kaplenko.b2clogin.com"
};
```

Рисунок 18 – Підключення Azure AD B2C

Кінцевий користувач бачить елементи html сторінки. Функції використані у цьому блоці прив’язані до подій, які можуть виникнути у разі взаємодії користувача з інтерактивними елементами html сторінки.

Одними з перших були реалізовані функції входу та виходу зі сторінки користувача (рисунок 19).

```
login() {
  if (this.msalGuardConfig.authRequest){
    this.authService.loginRedirect({...this.msalGuardConfig.authRequest} as RedirectRequest);
  } else {
    this.authService.loginRedirect();
  }
}

logout() {
  this.authService.logoutRedirect({
    postLogoutRedirectUri: 'https://localhost:4200/home'
  });
}
```

Рисунок 19 – Функції login та logout

Розглянемо тривіальну функцію отримання інформації з серверної частини додатку та запису її до змінної клієнтської частини. Наприклад списку кандидатів. Це наглядно продемонструє взаємодію компоненти, моделі та сервісу створених для опрацювання кандидатів.

Перш за все нам потрібно визначити, які поля ми хочемо отримати із запиту та створити відповідний клас або інтерфейс моделі (рисунок 20).

```
export interface Candidate {
    id: string,
    name : string,
    surname : string,
    email : string,
    phone : string,
    linkedIn : string,
    processingStatus: string
}
```

Рисунок 20 – Інтерфейс моделі кандидатів

Далі нам необхідно зробити http запит на сервер (рисунок 21). Для цього необхідно знати куди відправляти запит, встановити параметри та передати їх безпосередньо до самого запиту. Ця функція повинна бути винесена до необхідного сервісу.

```
getTechnologyCandidate(candidateId: string): Observable<Technology[]>{
  let params = new HttpParams()
    .set("candidateId", candidateId!);
  return this.http.get<Technology[]>(this.baseUrl + "GetTechnologyCandidate", { params : params })
}
```

Рисунок 21 – Реалізація http запиту get за допомогою Angular

Саме до неї ми будемо звертатися у компоненті, якщо користувач захоче отримати відповідний список. Але нам потрібно не лише відправити інформацію, а й отримати відповідь. Для цього ми підписуємося на подію та коли її буде виконано, то інформацію буде записана до змінної response (рисунок 22).

```
getTechnologyCandidate(candidateId: string){
  this.candidateService.getTechnologyCandidate(candidateId)
  .subscribe(
    response => {
      this.technology = response;
      console.log(response);
    }
  );
}
```

Рисунок 22 – Функція запиту списку користувачів

Значна частина коду працює за аналогічної схемою. Може відрізнятися тип запиту, його тіло та заголовки, але алгоритм незмінний.

Клієнтська частина використовує готові рішення для відображення розкладу та попереднього перегляду документів.

PdfTron – додаток використаний для відображення документів [23]. Його функціонал було розширено щодо вимог нашої системи. Додано темну тему та перемикач до неї, а також відкриття файлів з пристрою користувача. Частина коду підключення зображена на рисунку 23.

```
  WebViewer({
    path: '../../assets/lib',
  }, this.viewer.nativeElement).then(instance => {
    const { Feature } = instance.UI;
    instance.UI.enableFeatures([Feature.FilePicker]);

    document.getElementById('file_upload')?.addEventListener('change', (e) => {
      instance.UI.loadDocument(this.currentFile!);
    });
  });
}
```

Рисунок 23 – Підключення PdfTron до нашої системи

Angular Schedule Component – рішення компанії Syncfusion [24]. Воно потребує спеціального токену. Тому після успішної реєстрації та підключення його до нашого додатку ми можемо використовувати його повноцінно.

Єдина проблема полягає в тому, що цей розклад використовує стилі, які мають досить високий пріоритет.

Налаштування рішення дуже гнучке і потребує деякого часу. Перш за все потрібно визначитися з вкладками розкладу та підключити їх у файл модулів. Тепер можемо створити компоненту, яка перевизначить Schedule Component за замовчуванням. Визначимо провайдери, стилі та шаблон відображення сторінки так як показано на рисунку 24.

```
@Component({
  selector: 'app-scheduler',
  providers: [DayService, WeekService, WorkWeekService, MonthService, AgendaService],
  templateUrl: './scheduler.component.html',
  styleUrls: ['./scheduler.component.css']
})
```

Рисунок 24 – Код підключення Schedule у нашу компоненту

Інформація, якою заповнюється розклад береться з серверної частини. Та вноситься у поле data (рисунок 25). Воно має JSON структуру, але не сталу. Розклад має багато налаштувань та параметрів. Тому прийнято зберігати інформацію у базі даних у вигляді рядку, а коли вона знадобиться конвертувати її знову до JSON об’єкту.

```
data: any[] = [];

public eventSettings: EventSettingsModel = {
    dataSource: this.data
};
```

Рисунок 25 – Змінна, що зберігає наповнення розкладу

У випадку скупчення значної інформації, наш додаток реалізує функцію пошуку потрібного елементу або вибірки, яка задовольняє значення деякого ключа. Ця функція реалізована за допомогою ріре елементів (рисунок 26).

```
transform(value: any, filteringString: string) {
  if(value.length === 0 || filteringString === '') {
    return value;
  }

  const candidates = [];
  for(const candidate of value) {
    if(candidate['id'].toString().toLowerCase().indexOf(filteringString.toLowerCase()) !== -1 ||
       candidate['name'].toLowerCase().indexOf(filteringString.toLowerCase()) !== -1 ||
       candidate['surname'].toLowerCase().indexOf(filteringString.toLowerCase()) !== -1 ||
       candidate['email'].toLowerCase().indexOf(filteringString.toLowerCase()) !== -1) {
         candidates.push(candidate);
       }
  }

  return candidates;
}
```

Рисунок 26 – Реалізація ріре для пошуку кандидата з вибірки

Адаптація системи під мобільні пристрої та планшети. Структура і вигляд блоків проєктувалися так, щоб потім її можна було підстроїти під менший розмір екрану. Деякі блоки доводилося повністю переписувати.

Загалом, щоб реалізувати це було використано наступний код (рисунок 27)

```
/* ================================ MEDIA QUERIES ================================ */
/* ============================ LAPTOPS/TABLETS ============================ */

@media screen and (max-width: 1200px) {
    .container {
        width: 94%;
        grid-template-columns: 7rem auto;
    }

    aside {
        display: block;
    }

    aside .top {
        justify-content: center;
    }

    aside .logo h2 {
        display: none;
    }
```

Рисунок 27 – Зображення логіки адаптації системи під планшети

Також фронтенд розробка передбачає стилізацію сторінки та її елементів. Цей процес звісно не вражає своєю складністю, на перший погляд, але дійсно вражає об’ємами.

Всього створено і використано 15 файлів стилів. Кожен з них створює або вдосконалює уже існуючий вигляд, положення або поведінку елемента.

Про красу сторінок можна говорити довго, але краще її показати.

РОЗДІЛ 3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Цей розділ допоможе Вам пройти шлях від реєстрації до повноцінної обробки першого кандидата.

Перш за все, Вам необхідно зареєструватися в нашій системі. Цей процес надзвичайно зручний, зрозумілий і не вимагає великої кількості інформації. Тож переглянемо сторінку реєстрації зображену на рисунку 28.

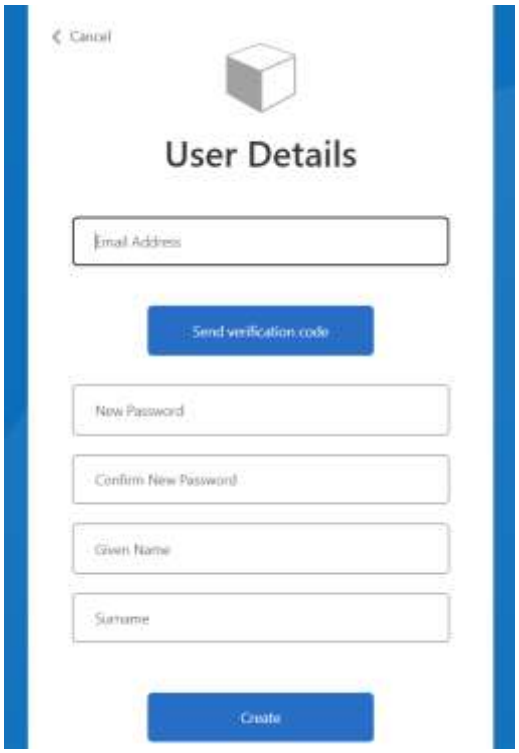

Рисунок 28 – сторінка реєстрації користувача

Заповнюємо поля. Щоб активувати акаунт необхідно ввести код верифікації. Його буде надіслано на вказану пошту. Копіюємо код, та вставляємо у спеціальне поле (рисунок 29).

Рисунок 29 – Використання кода верифікації

Якщо акаунт вже існує, то Ви можете авторизуватися, навіть з використанням Google та Facebook сервісів (рисунок 30).

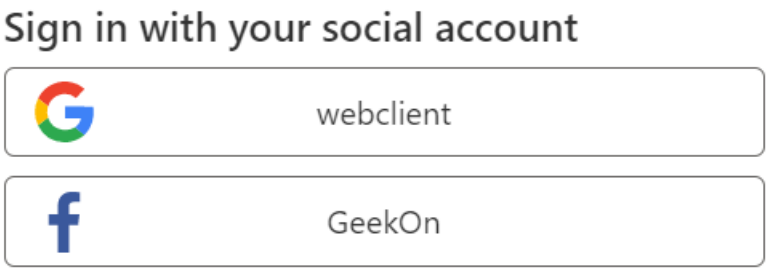

Рисунок 30 – Сервіси, доступні для авторизації

Тепер, коли ми увійшли до свого акаунту, можемо переглянути свій профіль. Тут зображено інформацію, котра доступна системі (рисунок 31).

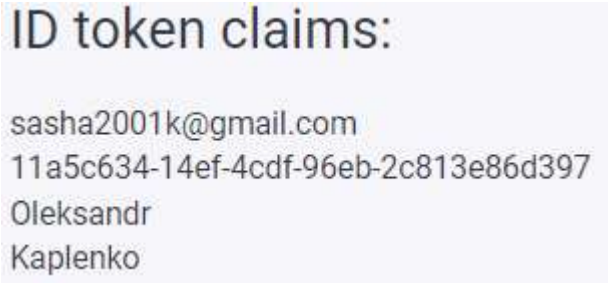

Рисунок 31 – Поля клієнта у системі

Тепер переходимо безпосередньо до демонстрації головної функції системи. Створюємо проєкт (рисунок 32). Та переходимо до нього.

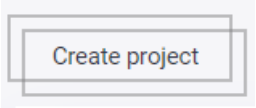

Рисунок 32 – Кнопка створення проєкту

Натискаємо, заповнюємо поля у формі, що з’явилася та переходимо до створеного проєкту (рисунок 33).

Рисунок 33 – Вигляд сторінки проєкту

Вона має досить широкий функціонал. Перш за все ми можемо переглянути інформацію про створений проєкт. Також тут розміщено перелік позицій. Саме сюди ми додаємо потенційних працівників нашої компанії.

Якщо їх буде досить велика кількість, то пошук може стати проблематичним. Для цього існує поле пошуку позицій (рисунок 34).

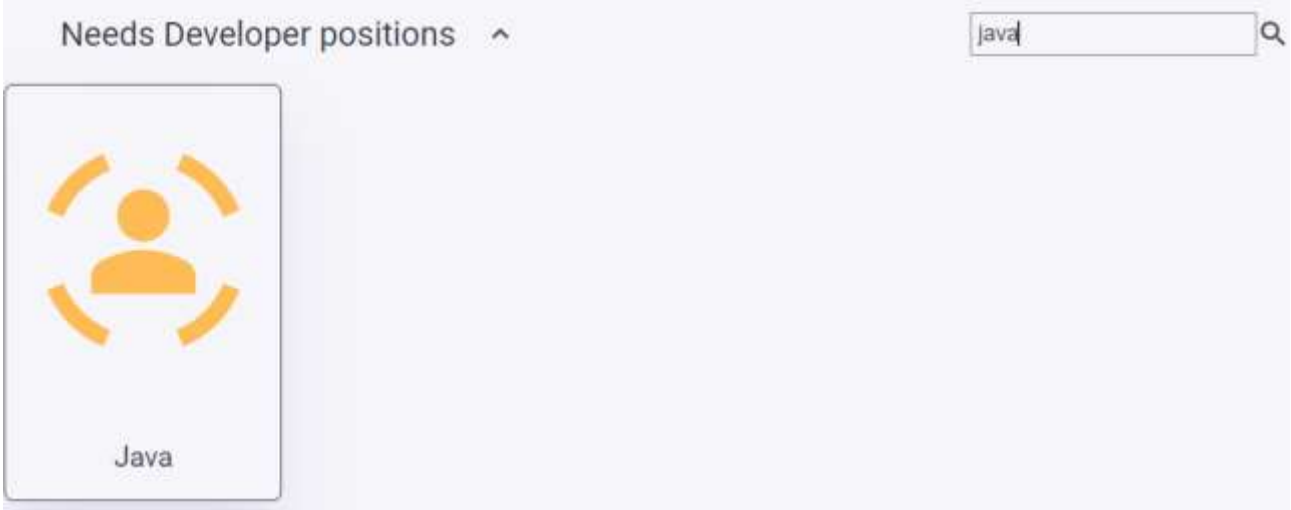

Рисунок 34 – Використання поля сортування технологій

Спробуємо додати першого кандидата до цієї категорії. Тицяємо на неї. Відкрилося віконце технології, зображене на рисунку 35.

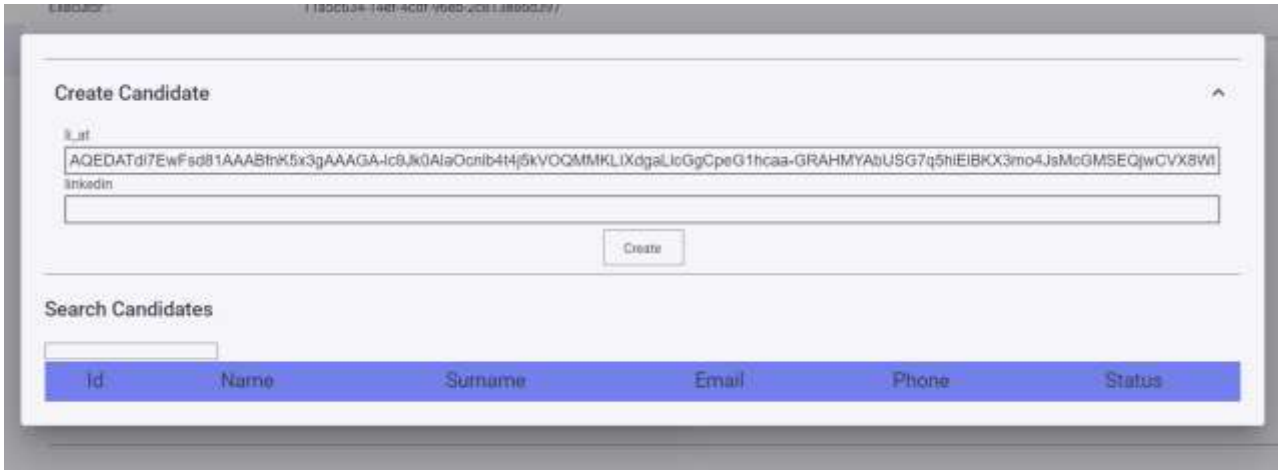

Рисунок 35 – Вікно обраної технології

В спеціальне поле необхідно ввести посилання на LinkedIn сторінку. Та натискаємо кнопку Create. Відкриється віконце, у якому відбудеться вся магія.

Тепер одразу після завершення ми потрапляємо до віконця редагування кандидата. Воно має досить багато функцій, які ми розглянемо далі.

Кандидат може бути розглянутий одразу для декількох позицій. Демонстрація цього явища зображена на рисунку 36.

Рисунок 36 – Секція редагування списку технологій кандидата

Синім виділено ті технології, що залишаться у списку після збереження, червоним – ті, що буде видалено, а також, у правому кутку зображено ті, що можуть стати потенційними елементами цього списку.

Також ця сторінка має секцію обов'язкових полів кандидата (рисунок 37).

Рисунок 37 – Список обов'язкових полів кандидата

Усі поля, що заповнено, були витягнуті з LinkedIn сторінки. Деяка кількість залишилася порожніми, це означає, що відповідну інформацію було не знайдено. Але, їх можна заповнити вручну.

Також існує додаткова інформація, що була знайдена на сторінці (рисунок 38).

Рисунок 38 – Список додаткової інформації кандидата

Цю інформацію можна гнучко редагувати, доповнювати та вилучати.

Вилучена інформація потрапляє до окремого списку, щоб випадкові зміни не були критичними.

Тут ми можемо змінити не лише значення полів, а ще й назву. Тобто керувати кандидатом досить приємно і легко. Сподіваємося, що Вам буде зручно використовувати представлений функціонал.

Також, ця сторінка не може бути закрита без підтвердження (рисунок 39), оскільки ми піклуємося про Ваші зусилля.

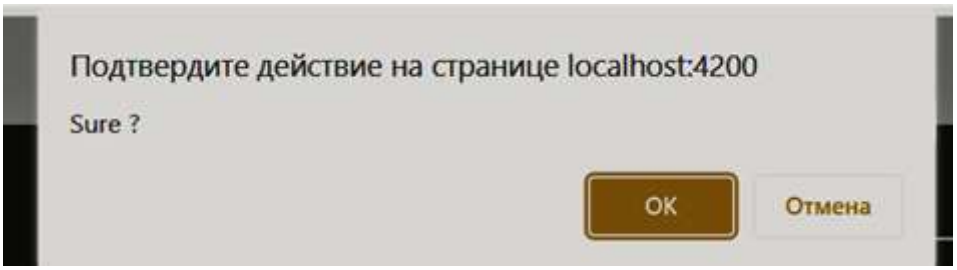

Рисунок 39 – Підтвердження виходу з форми

Також, може виявитися, що клієнту необхідно буде присвоїти файли. Наприклад його резюме, чи тестове завдання. Наша система дозволяє зберігати файли будь-якого формату, та надає до них доступ за посиланням, навіть поза системою. Тож перейдемо до огляду цієї функції. Перш за все необхідно обрати кандидата як показано на рисунку 40.

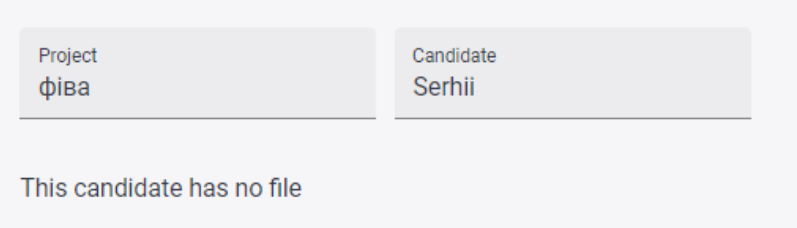

Рисунок 40 – Вибір потрібного кандидата

Бачимо, що даний кандидат не має жодного файлу. Тому додамо йому деяку картинку. Натискаємо Select File. Обираємо потрібний файл. Його буде відображено у віконці попереднього перегляду.

Тепер ми можемо його редагувати. Загалом, функцій досить багато, тому пропоную Вам самостійно їх дослідити.

Також, паралельно з цим, заповнилася форма, щоб швидко і зручно завантажити файл до нашої системи. Переглянемо вигляд сторінки до натискання кнопки Add (рисунок 41).

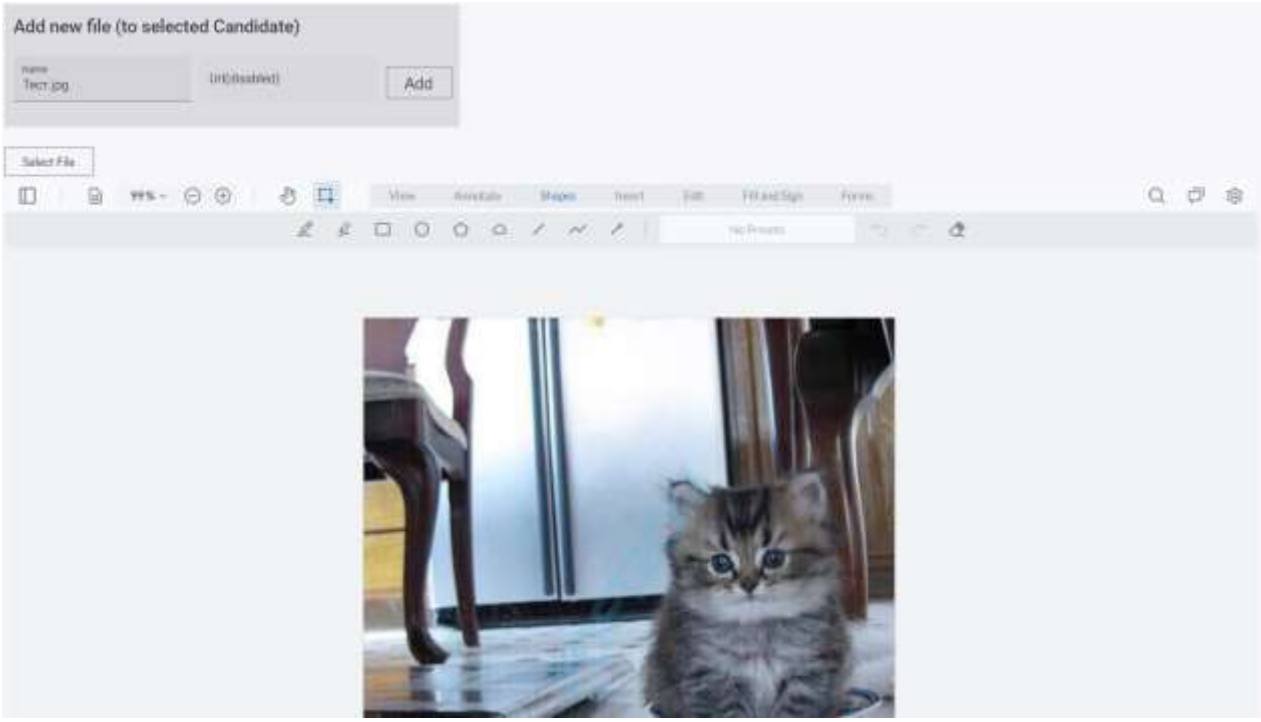

Рисунок 41 – Вигляд сторінки після обрання файлу

Крім обробки полів кандидата, професія HR менеджера передбачає спілкування з потенційним працівником. Наша система дозволяє планувати такі зустрічі. Кожен проєкт містить розклад (рисунок 42), у якому ми можемо легко додавати необхідні Вам події, а, також, редагувати їх за необхідності. Налаштування події досить гнучке. Можна обрати часовий пояс, локацію, час початку та завершення, періодичність повторення і багато чого ще.

Рисунок 42 – Приклад створеної події

Також, на цій же сторінці, можемо створити Zoom конференцію, для зустрічі з клієнтом.

Тож наша система дозволяє автоматизувати досить широкий спектр функцій HR спеціаліста, за допомогою описаних у цій інструкції кроків.

Блукайте системою, досліджуйте її, ми певні, що Ви зможете оцінити її сильні сторони та використовувати її на практиці.

Бажаю Успіхів!

ВИСНОВКИ

Отже, здійснивши розробку допоміжного додатку для HR-спеціалістів, ми можемо зробити загальні висновки, відповідно до сформульованих мети й завдань нашої кваліфікаційної роботи.

Насамперед було визначено план роботи над продуктом. Обрано сферу його функціонування.

HR-сфера як специфічне середовище функціонування реалізованого проєкту диктує ряд вимог до затребуваного технологічного додатку в умовах нинішньої ринкової конкуренції. Часто ефективність виконання своїх посадових обов'язків HR-спеціалістом безпосередньо залежить від володіння ним різними інструментами (зокрема, IT-технологіями) для побудови якісних процесів. Багатофункціональність ролі цього фахівця сьогодні вимагає від нього такої собі мультикомпетентності, якщо говорити про реальні бізнес-партнерські амбіції відповідного посадовця.

У процесі реалізації проєкту було поглиблено знання з технологій розробки додатків та визначено ті з них, які будуть застосовані у роботі. Так, в якості інструменту створення програмного засобу було обрано комбінацію технологій ASP.NET Core (виступає у ролі серверної частини) та Angular (за допомогою якої реалізована клієнтська частина проєкту).

Реалізація серверного блоку коду стала можливою завдяки використанню інтегрованого середовища розробки Visual Studio 2022 для автоматичного завершення коду, який розпізнає контекст; технології MS SQL Server для автоматичного пов'язування моделей нашої системи з таблицями у базі даних; пакета Selenium для зчитування інформації з сайту.

Реалізація клієнтської частини, що передбачає роботу з файлами, була здійснена шляхом використання платформи PdfTron для перегляду файлів; технології Blob сховища Azure для збереження даних; системи Syncfusion Schedule для календарного планування подій. Для забезпечення комфортної атмосфери та

естетизації процесів було додано сервіс MatDialog, а також застосовано Web Animations, Toastr, Azure Active Directory B2C.

Опис структурних частин нашої програми здійснено у другому розділі кваліфікаційної роботи. Систематизовано вимоги до функціонування відповідного додатку, презентовано його інтерфейс. Тут варто відзначити, що даний проєкт реалізовано максимально в інтересах користувача, а відтак, було застосовано цілий ряд зручних пропозицій переважно візуального спектру індикаторів – кольору, уніфікації стилю зображення, маркування елементів тощо. Також детально описано структуру та реалізацію програми.

Для забезпечення успішного функціонування подано інструкцію користувача у межах третього розділу кваліфікаційної роботи.

Програмний продукт – допоміжний технологічний додаток – створено безпосередньо для потреб застосування HR-спеціалістами. Таким чином, отримані результати доводять практичну доцільність реалізованого проєкту.

У процесі розробки веб-ресурсу та оформленні кваліфікаційної роботи на практиці були закріплені теоретичні знання, вдосконалені навички програмування, проведено оформлення технічної документації в текстових та графічних редакторах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Леонова О. 30+ додатків для рекрутерів та HR-ів. [Електронний ресурс] – Режим доступу до ресурсу: https://hurma.work/blog/30-dodatkiv-dlya-rekruteriv-ta-hr-iv/

2. HackerRank [Електронний ресурс] – Режим доступу до ресурсу: https://www.hackerrank.com/

3. Pymetrics [Електронний ресурс] – Режим доступу до ресурсу: https://www.pymetrics.ai/

4. Self Management Group [Електронний ресурс] – Режим доступу до ресурсу: https://www.selfmgmt.com/

5. Ideal [Електронний ресурс] – Режим доступу до ресурсу: https://www.ideal.nl/en/

6. Textio [Електронний ресурс] – Режим доступу до ресурсу: https://textio.com/

7. Bullhorn [Електронний ресурс] – Режим доступу до ресурсу: https://www.bullhorn.com/

8. Greenhouse [Електронний ресурс] – Режим доступу до ресурсу: https://www.greenhouse.io/

9. SAP SuccessFactors [Електронний ресурс] – Режим доступу до ресурсу: https://www.sap.com/products/human-resources-hcm/products.html

10. iCIMS [Електронний ресурс] – Режим доступу до ресурсу: https://www.icims.com/

11. Jobvite [Електронний ресурс] – Режим доступу до ресурсу: https://www.jobvite.com/

12. Workable [Електронний ресурс] – Режим доступу до ресурсу: https://www.workable.com/

13. ConveyIQ [Електронний ресурс] – Режим доступу до ресурсу: https://www.conveyiq.com/blog

14. Wepow [Електронний ресурс] – Режим доступу до ресурсу: https://support.wepow.com/hc/en-us

15. Zenefits [Електронний ресурс] – Режим доступу до ресурсу: https://www.zenefits.com/

16. OpenTute [Електронний ресурс] – Режим доступу до ресурсу: https://opentute.com/

17. Microsoft Visual Studio 2022 [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/ru/vs/

18. Wikipedia LinkedIn 2022 [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/LinkedIn#%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97

19. Syncfusion documentation Angular Schedule [Електронний ресурс] – Режим доступу до ресурсу: https://help.syncfusion.com/angular/schedule/overview

20. Microsoft documentation Огляд сховища Blob об’єктів [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/azure/storage/blobs/storage-blobs-overview

21. Microsoft documentation About Azure AD B2C [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/azure/active-directory-b2c/overview

22. Microsoft documentation Entity Framework 6 [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database

23. PdfTron platform [Електронний ресурс] – Режим доступу до ресурсу: https://www.pdftron.com/

24. Syncfusion documentation Getting started with Angular Schedule component [Електронний ресурс] – Режим доступу до ресурсу: https://ej2.syncfusion.com/angular/documentation/schedule/getting-started/