МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

САЙТ ДЛЯ ПОШУКУ РОБОТИ ДЛЯ РОЗРОБНИКІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Здобувач освіти гр. ІН – 83-9 Данило ВОРОШИЛОВ

Науковий керівник Сергій ПЕТРОВ

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-83-9 спеціальності «122 – Комп'ютерні науки» денної форми навчання Ворошилова Данила Олександровича.

Тема: «САЙТ ДЛЯ ПОШУКУ РОБОТИ ДЛЯ РОЗРОБНИКІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) вибір інструментів для розробки веб сайту; 3) практична реалізація; 4) висновки.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____Сергій ПЕТРОВ
Завдання прийняв до виконання _____ Данило ВОРОШИЛОВ

РЕФЕРАТ

Записка: 26 стор., 12 рис., 4 табл., 1 додаток, 7 джерел.

Об’єкт дослідження — сайт для пошуку роботи для розробників програмного забезпечення.

Мета роботи — розробка сайту для пошуку роботи, за допомогою сучасних технологій таких як React та Strapi.

Результати — в результаті роботи ми маємо, веб додаток для пошуку роботи створений за допомогою найсучасніших веб технологій.

ВЕБ ДОДАТОК, REACT, STRAPI

ЗМІСТ

ВСТУП 4
1 ІНФОРМАЦІЙНИЙ ОГЛЯД 5
2 ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ВЕБ САЙТУ 6
2.1 React 6
2.2 Strapi 6
2.3 Styled-components 7
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ 8
3.1 Реалізація бекенд частини. Налаштування Strapi. 8
3.2 Реалізація фронтенд частини. 12
ВИСНОВКИ 22
СПИСОК ЛІТЕРАТУРИ 23
ДОДАТКИ 24

ВСТУП

Пошук роботи є постійною турботою багатьох людей у всьому світі. Тому багато хто звертається до інтернету зі своїми запитами. Пошук роботи через інтернет дуже спрощує життя людям.

Людям зручно переглядати інформацію в одному місці, маючи величезний вибір та різноманітність при виборі місця роботи, при цьому отримуючи можливість шукати роботу в будь-якому місці.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

Існує багато різноманітних сайтів з пошуку роботи, всі вони маю певні подібності.

Work.ua відомий в Україні сайт, послугами якого користуються велика кількість людей. Для спрощення пошуку передбачена функція розміщення резюме для того, щоб потенційні роботодавці могли знайти вас самі.

Rabota.ua альтернатива попереднього сайту, яка має схожу аудиторію. Доступні всі стандартні функції – заповнення анкети, сортування вакансій по категоріях, даті розміщення і вибору регіону.

OLX є не тільки популярним сайтом з оголошеннями, де можна купити/продати але і ресурсом з актуальними вакансіями. Зручність досягається шляхом сортування результатів за передбачуваною заробітною платою. На зв’язок з потенційним роботодавцем можна вийти відразу по телефону, електронній пошті.

Тому сайт з пошуку роботи, повинен відповідати на базові потреби користувачів.

При реєстрації користувачу на вибір буде надана можловись створити на вибір роботодавця (employer) або людину, яка шукає роботу (job-seeker).

Роботодавець має можливість створити компанію, кожен роботодавець може мати лише одну компанію, та створювати декілька вакансій для однієї компанії.

Людина яка шукає роботу, після реєстрації, має можливість переглянути список компаній. У списку компаній в пошуку, будуть лиш ті компанії, які мають хоч одну відкриту вакансію.

2 ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ВЕБ САЙТУ

Сайт складається з двох основних частин таких таких, як: фронтенд та бекенд.

Фронтенд - все, що браузер може читати, виводити на екран і/або запускати. Тобто це HTML, CSS і JavaScript.

Бекенд - все, що працює на сервері, підключеному до мережі, який відповідає на повідомлення від інших комп'ютерів.

На даний момент існує багато фреймворків та бібліотек для швидкого створення сайтів, як для фронтенду так і бекенду, серед них я вирішив використовувати найпопулярніші та найсучасніші технології, а саме: React - для фронтенд та Strapi - дя бекенду.

2.1 React.

React — це JavaScript-бібліотека від Facebook для зручної розробки інтерфейсів, тобто зовнішньої частини сайтів та програм, з якою взаємодіє користувач. React дозволяє розробникам створювати великі веб застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки.

Компоненти React.js пишуть особливою мовою - JSX, який виглядає як суміш JavaScript та HTML, який потім компілюется у JavaScript, який розуміє більшість браузерів.

2.2 Strapi.

Логіка традиційних CMS об’єднує бекенд та фронтенд частини однієї системи

Headless CMS - принципово інша система управління. Як правило, вона відповідає тільки за універсальне вміст, який може використовуватися на будь-яких платформах. Бекенд («тіло») при такому підході не пов’язаний з фронтенда («головою»).

Strapi - це фреймворк для керування контентом на основі Node.js. Він дозволяє швидко розробляти API для роботи з даними і займає проміжне положення між фреймворком для Node.js і CMS без інтерфейсу користувача. Strapi дозволяє розробляти API дуже швидко, що заощаджує час.

Strapi опенсорсний проект, що вигідно відрізняє його від інших CMS. Зокрема, це означає, що, по-перше, він повністю безкоштовний, а по-друге - те, що компанія, що обрала Strapi, розгортає CMS на власних серверах, тобто дані компанії залишаються під повним контролем. Крім того, Strapi можна налаштовувати та розширювати завдяки системі плагінів.

2.3 Styled-components.

Styled Components — загальновідома бібліотека для стилізації програм React. Вона дозволяє створювати власні компоненти за допомогою написання самого CSS в JavaScript.

Переваги Styled Components

1. Автоматична генерація критичного CSS. Бібліотека відстежує компоненти, що відображаються на сторінці, і додає лише їх стилі.
2. Проста динамічна стилізація. Стилі приєднуються на основі властивостей чи теми.
3. Зручне обслуговування. Додавання стилів відбувається не в різних файлах, а в одному місці.
4. Автоматичне застосування вендорних префіксів. Styled Components бере все у свої руки.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Реалізація бекенд частини. Налаштування Strapi.

Генерації всіх файлів та папок вводимо комаду до терміналу:

```
npx create-strapi-app@latest my-project --quickstart
```

Після завершення встановлення браузер автоматично відкриває нову вкладку (localhost:1337/admin/). Та просить заповнити форму для створення адміністратора. Після чого ми можемо приступити до налаштування (рис. 1.1).

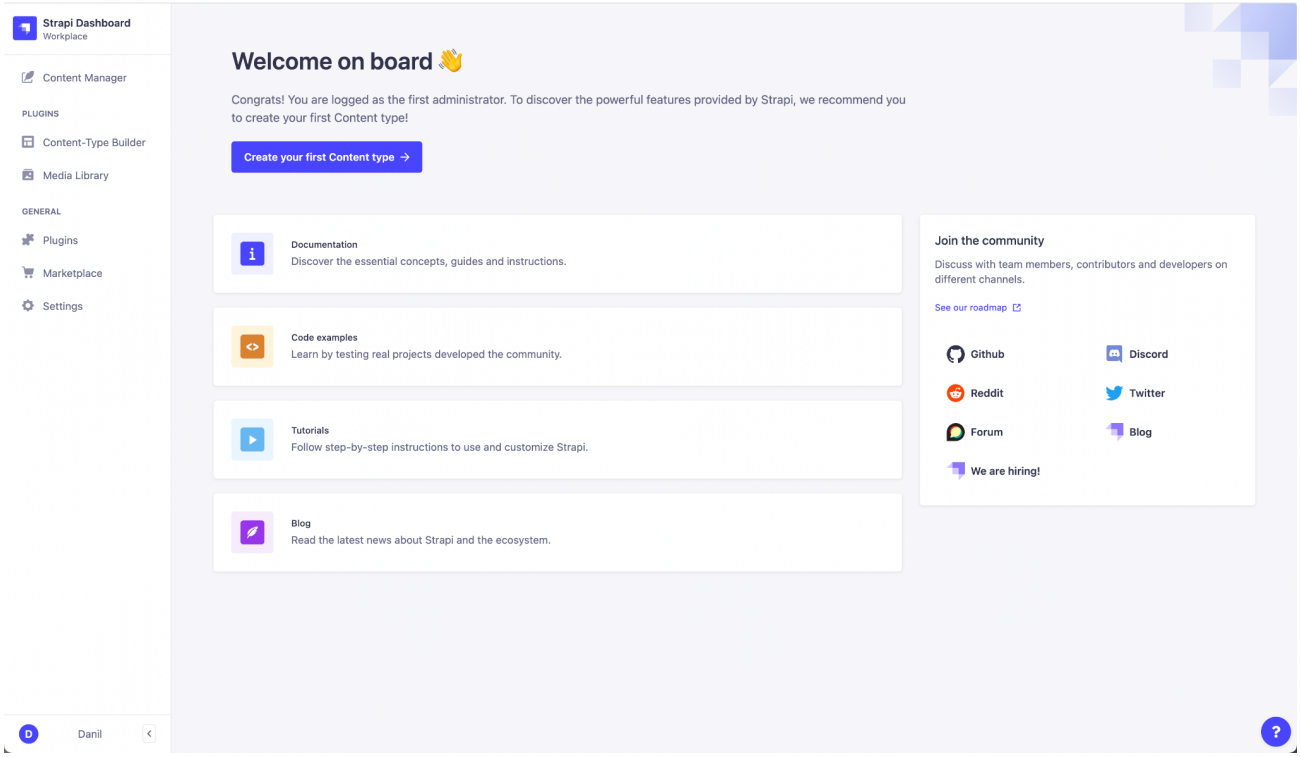
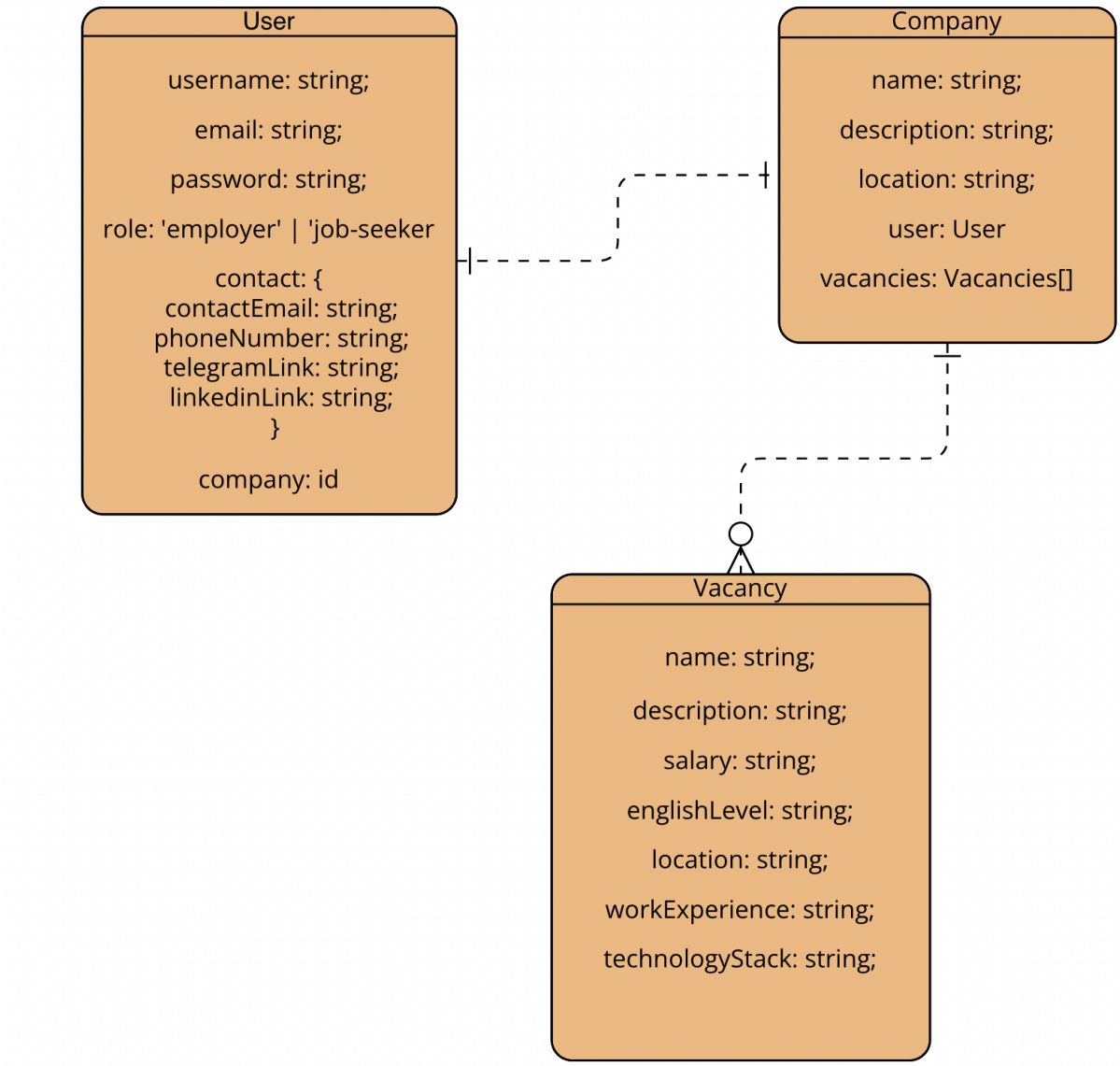


Рисунок 1.1 — Головная сторінка адміністра Strapi

Для зберігання інформації для роботи веб сайту, створюємо таблиці в базі даних за допомогою інструментів Strapi.

Створюємо схематичну модель бази даних (рис 1.2):



На основі діаграми за допомогою інструменту Content-Type Builder, створюємо Collection type для збереження даних.

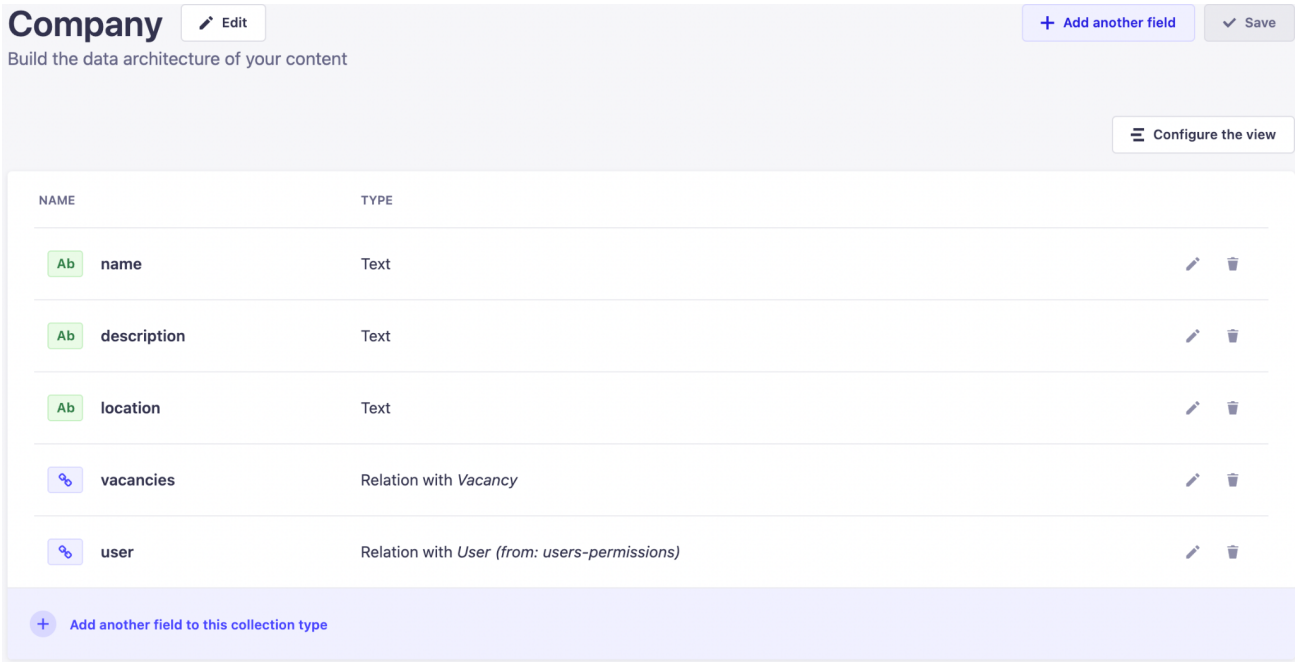


Рисунок 1.3 — Таблиця для зберігання Company

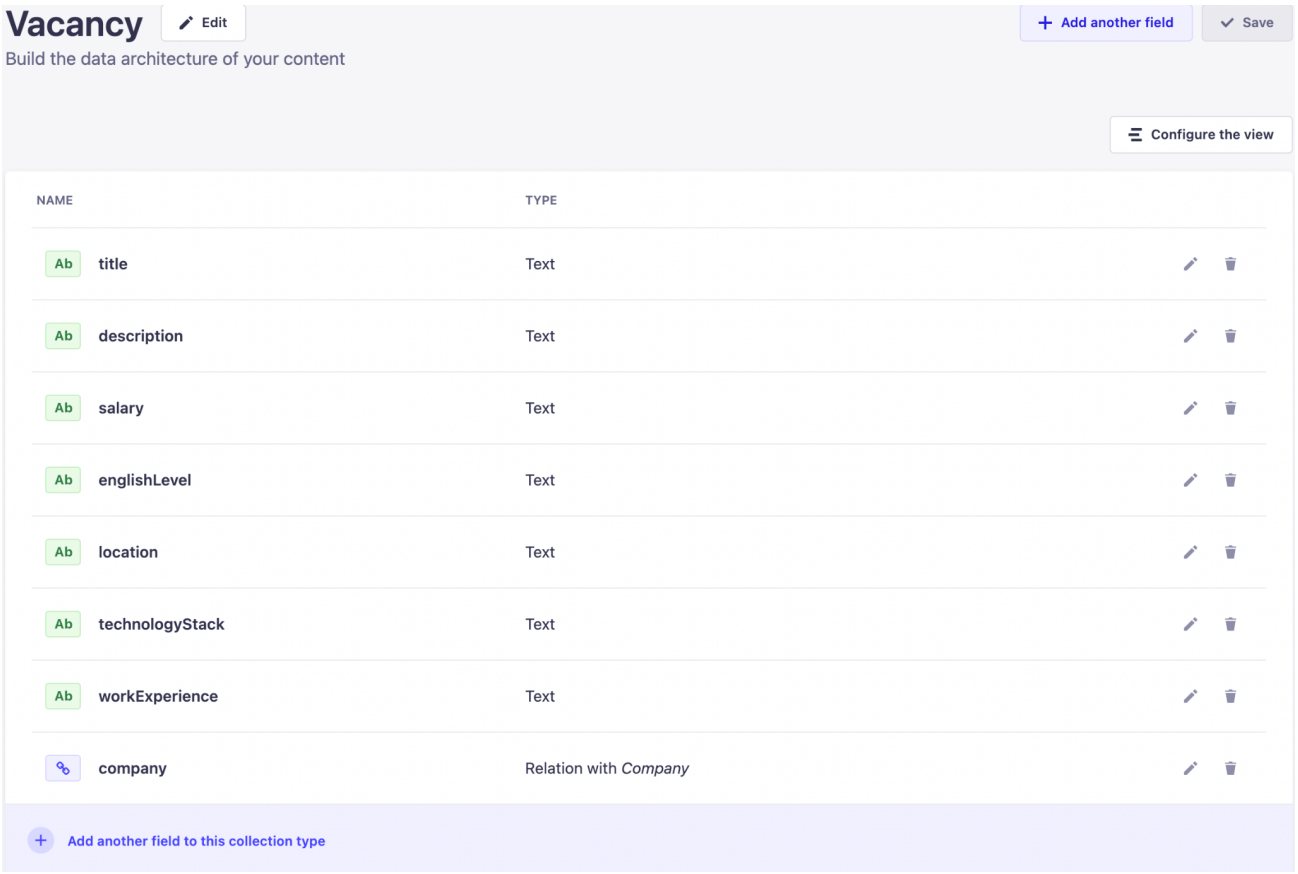


Рисунок 1.4 — Таблиця для зберігання Vacancy

Додаємо в автоматично згенеровану таблицю User потрібні поля.

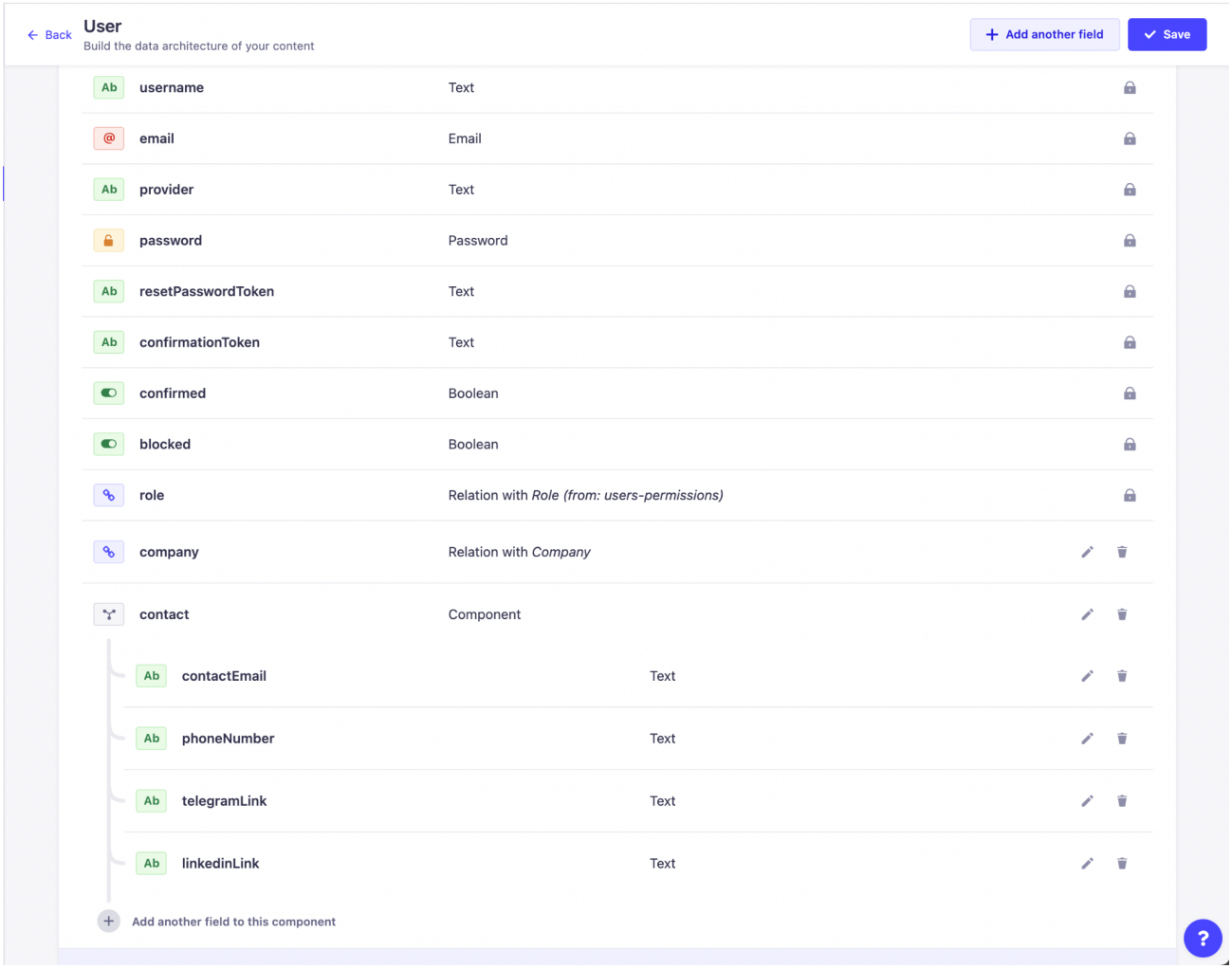


Рисунок 1.5 — Таблиця для зберігання User

Після створення таблиць, Strapi автоматично генерує контроллери для створення, змінення та видалення даних в таблицях (рис 1.6).

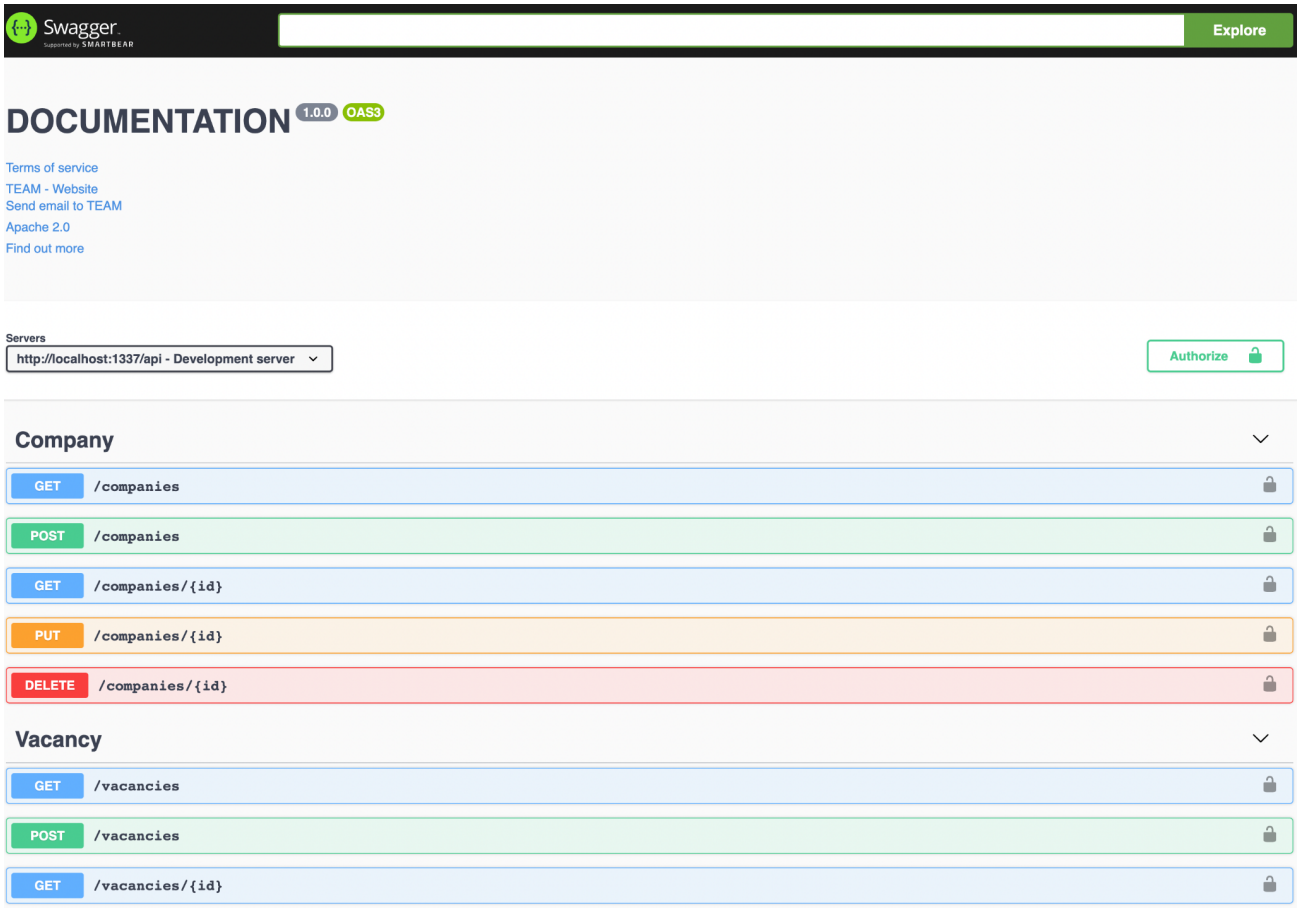


Рисунок 1.6 — Список ендпоентів для управління данним

3.2 Реалізація фронтенд частини.

Для генерації React проекту використовуємо команду в терміналі:

```
npx create-react-app itrun-frontend
```

Структура файлів та папок (рис 2.1):

1. components – папка з компонентами сторінок (наприклад header)
2. pages – папка з сторінками
3. services – все що стосується з'єднання з бекендом
4. index.jsx – основний файл з якого починається старт сайту
5. App.jsx – налаштування сторінок та їх url

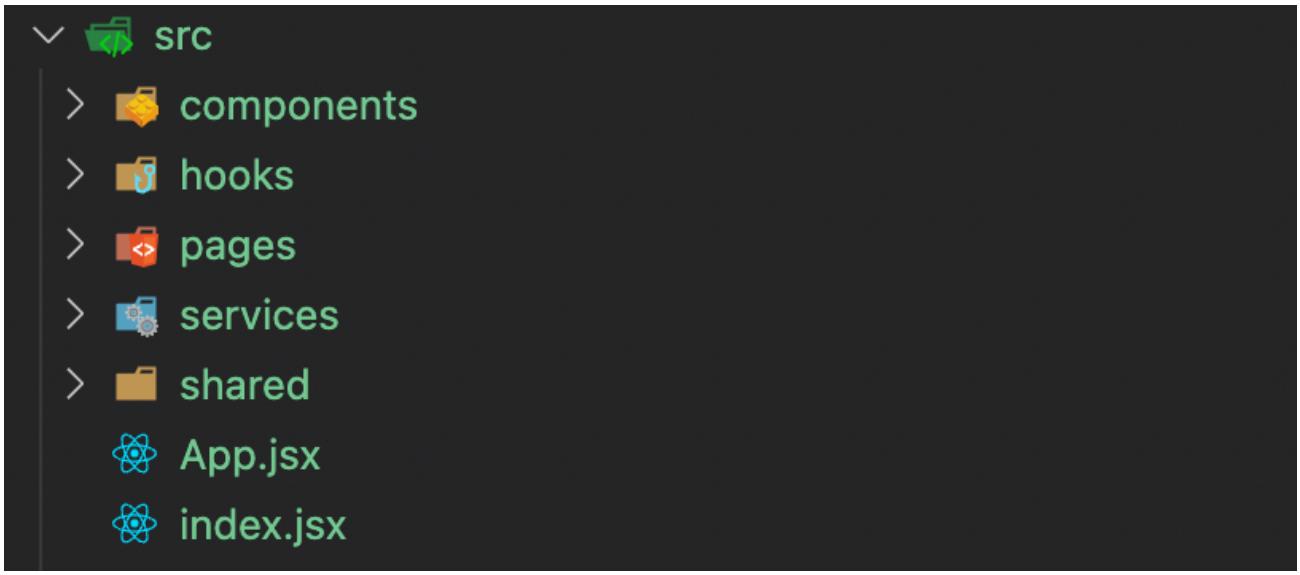


Рисунок 2.1 — Структура файлів та папок

Після входу на сайт пропонуємо користувачу створити аккаунт на вибір:

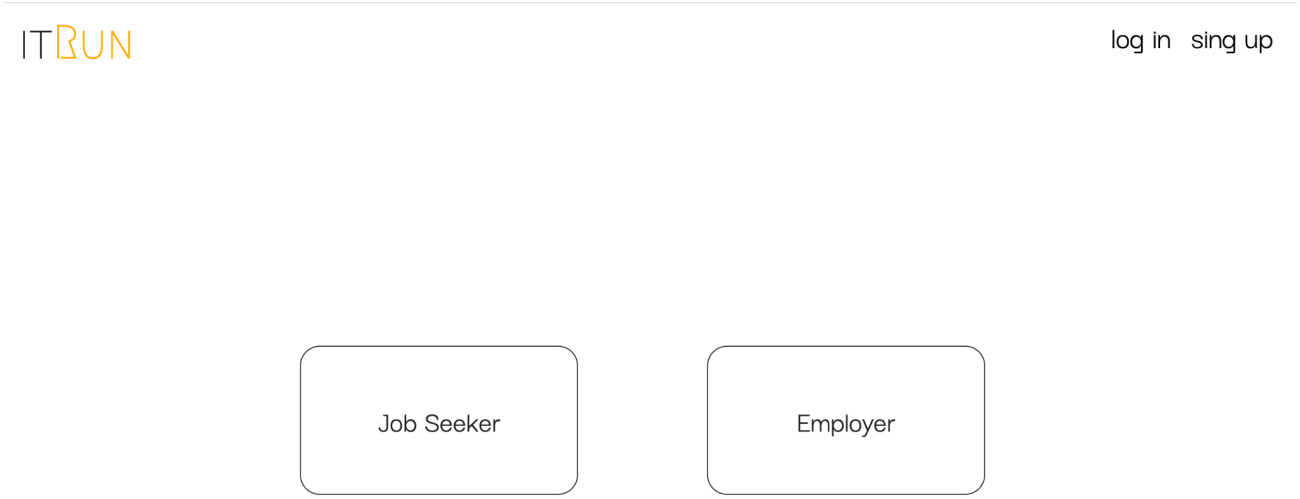


Рисунок 2.2 — Sing up сторінка

Для реєстрації користувача використовуємо бібліотеку react-hook-form. React Hook Form — одна з найпопулярніших бібліотек обробки елементів введення форми в екосистемі React. Додаток А.1.

При виборі employer, переходимо на сторінку, для заповнення інформації про роботодавця та його компанію. Після заповнення даних та створення аккаунту переходимо на сторінку профілю, де можна переглянути, відредагувати та створити вакансію (рис 2.3). Додаток А.2.

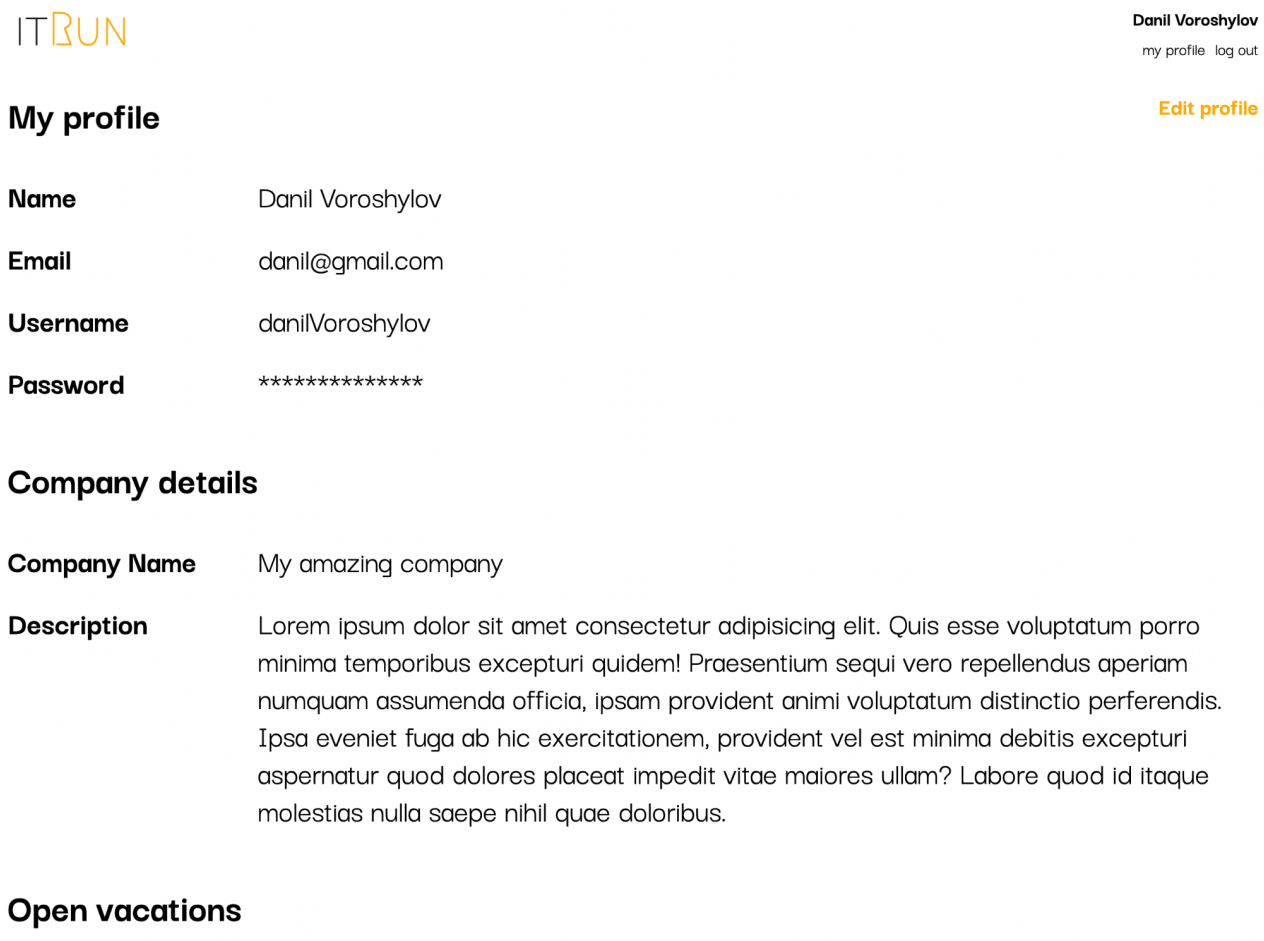


Рисунок 2.3 — Профіль employer

Створюємо вакансію для нашої компанії (рис 2.4) Додаток А.3:

1. Назву вакансії
2. Опис
3. Заробітну плату
4. Володіння англійською мовою

5. Місцезнаходження
6. Бажаний досвід роботи.

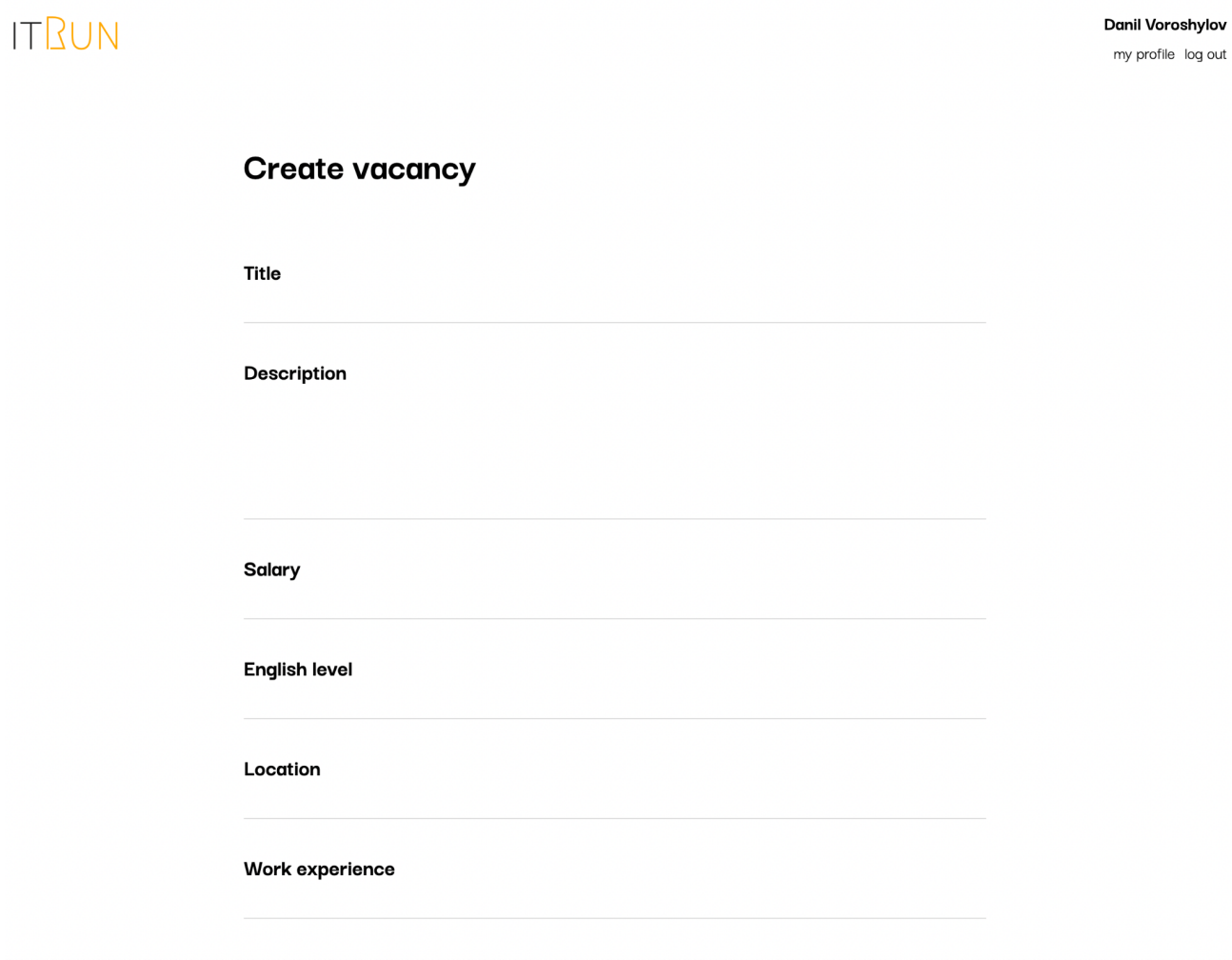


Рисунок 2.4 — Створення вакансії

Після створення вакансій, їх можно перглянути/редагувати/видалити в профілі. Та тепер вони будуть відображатися в пошуку для людей які шукають роботу. Додаток А.4.

Після створення аккаунту як job seeker. Ми можемо перейти на сторінку пошуку де відображається список компаній у яких є відкриті вакансії (рис 2.5).

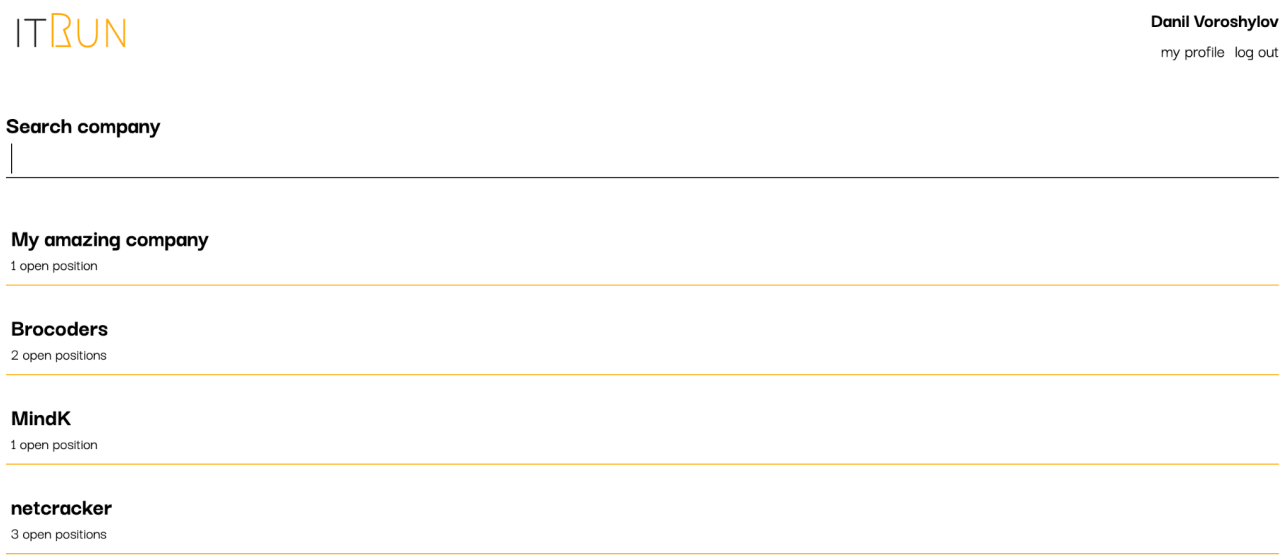


Рисунок 2.4 — Пошук компанії

По кліку на компанію, відкривається детальна інформація о компанії та її вакансіях, та інформація про контактні дані (рис. 2.5). Додаток А.5.

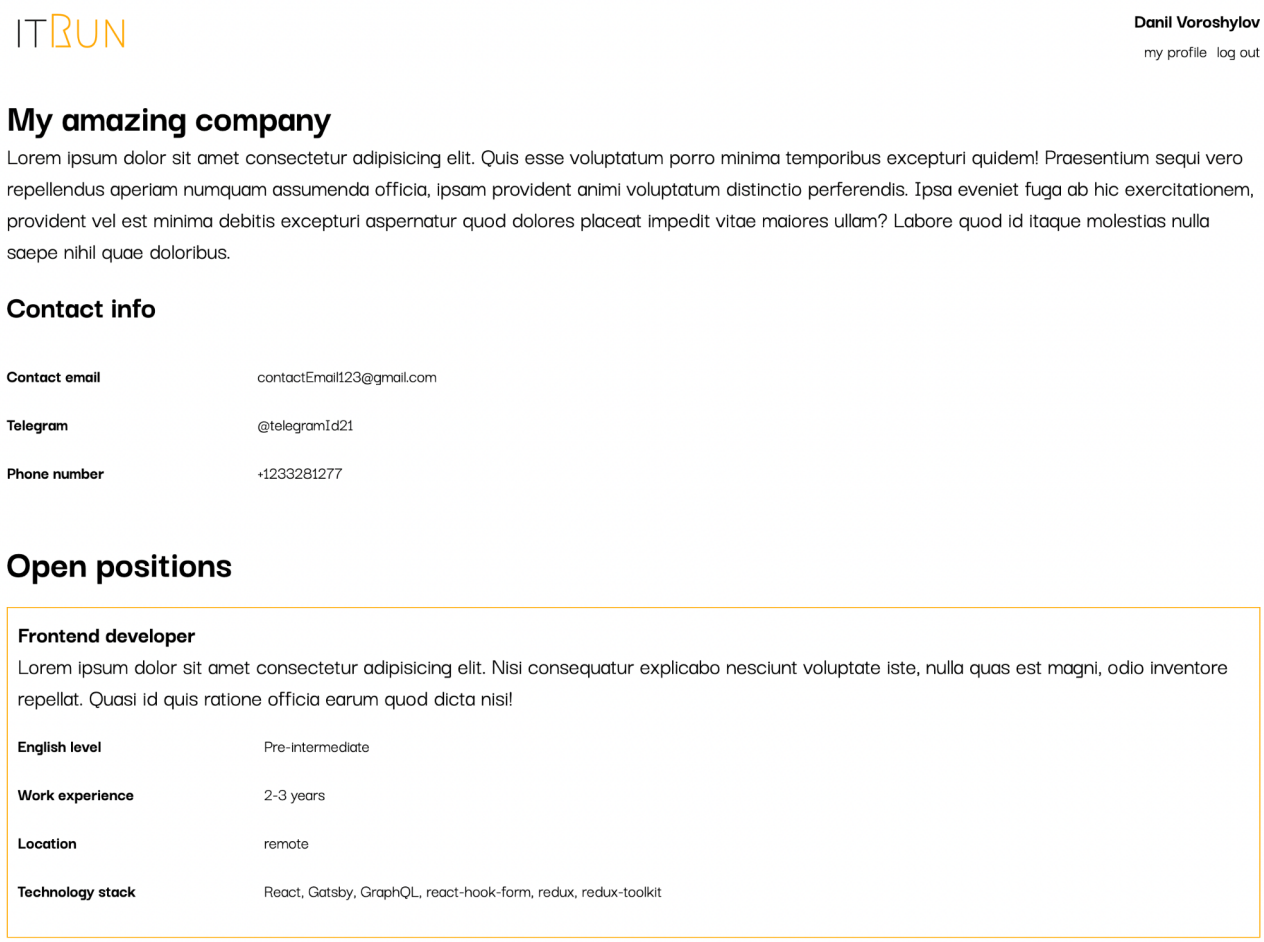


Рисунок 2.5 — Перегляд компанії та вакансій

Створення та реєстарція роутів (маршрутизації) для сторінок виконуеться за допомогую библиотеки react-router-dom.

React Router — це стандартна бібліотека маршрутизації (маршрутизації) в React. Він зберігає інтерфейс додатків синхронізованим з URL-адресою в браузері. React Router дозволяє вам маршрутизувати "поток даних" (потік даних) у вашому додатку понятним способом. Це таке підтвердження, якщо у вас є дана URL-адреса, це буде такий маршрут (маршруту), і інтерфейс буде таким

Нижче можно переглянути код реалізації на проекті:

```
<Routes>
  <Route element={<Layout />}>
    <Route path="/" element={<Home />} />
    <Route path="/sing-up" element={<SingUp />} />
    <Route path="/log-in" element={<LogIn />} />
    <Route path="/sing-up/job-seeker" element={<JobSeeker />} />
    <Route path="/sing-up/employer" element={<Employer />} />
    <Route path="/profile" element={<Profile />} />
    <Route path="/vacancy/:id" element={<Vacancy />} />
    <Route path="/search" element={<Search />} />
    <Route path="/company/:id" element={<CompanyInfo />} />
    <Route path="*" element={<Navigate to="/" replace />} />
  </Route>
</Routes>
```

Також для доступу к серверу був створений допоміжній файл (services/api.js) в якому описані функції до отримання та відправлення даних.

Нижче можно переглянути код реалізації на проекті :

```
const commonAxiosInstance = axios.create({
  baseURL: 'http://localhost:1337/api'
});

const authenticatedAxiosInstance = axios.create({
  baseURL: 'http://localhost:1337/api',
  headers: {
    Authorization: `Bearer ${localStorage.getItem('token')}`
  }
});

export const registerUser = async (data) => {
  try {
    return await commonAxiosInstance.post('/auth/local/register', data);
  } catch (err) {
    console.log(err);
  }
};

export const authUser = async (data) => {
  try {
    return await commonAxiosInstance.post('/auth/local', data);
  } catch (err) {
    console.log(err);
  }
};

export const getUser = async () => {
  try {
    return await authenticatedAxiosInstance.get(`/users/me`);
  } catch (err) {
    console.log(err);
  }
};
```

На прикладі реєстрації, авторизації та отримання юзеру с серверу, розглянемо детальніше, як саме ми отримуємо інформацію з серверу за допомогою axios.

Для початку розберемося, що таке axios. Axios це один із найпопулярніших HTTP клієнтів для браузерів та node.js, заснований на промісах.

В Axios є підтримка запитів, отримання відповідей від сервера, їх трансформація та автоматична конвертація в JSON.

Для початку розберемо, цей кусок коду:

```
const commonAxiosInstance = axios.create({
  baseURL: 'http://localhost:1337/api'
});
```

Для початку розберемо, цей кусок коду. При виклику функції create, в яку ми передаємо об'єкти з налаштуваннями (в нашому випадку baseURL: ‘http://localhost:1337/api’), вона повертає нам інстанс axios з параметрами які ми передали до функції.

Створений інстанс ми використовуємо для відправки post запиту, для реєстрації юзера.

```
export const registerUser = async (data) => {
  try {
    return await commonAxiosInstance.post('/auth/local/register', data);
  } catch (err) {
    console.log(err);
  }
};
```

Функція registerUser приймає data та повертає response, де:

```
data = {
  username: string;
  email: string;
  password: string;
}

response = {
  jwt: token string;
  user: User{}
}
```

jwt токен ми будемо використовувати для доступу до private endpoint таких як ‘/users/me’

```
const authenticatedAxiosInstance = axios.create({
  baseURL: 'http://localhost:1337/api',
  headers: {
    Authorization: `Bearer ${localStorage.getItem('token')}`
  }
});
```

Для доступу до private endpoint, щоб отримати дані про юзера, створюємо інстанс в який передаємо токен для авторизації, який заздалегідь зберегли в localStorage. Тепер цей інстанс має змогу використовувати дані з private endpoint.

Тепер для отримання даних про юзера, за допомогою метода get нашого авторизованого інстансу.

```
export const getUser = async () => {
  try {
    return await authenticatedAxiosInstance.get(`/users/me`);
  } catch (err) {
    console.log(err);
  }
};
```

ВИСНОВКИ

В процесі створення веб сайту ми ознайомилися та використали, найсучасніші і найпопулярніші інструменти для створення веб додатків, такі як: React.js, Strapi, styled-components, react-hook-form, react-router-dom та інші.

В результаті роботи ми маємо, веб додаток для пошуку роботи створений за допомогою найсучасніших веб технологій.

Розроблений ресурс має базовий функціонал та задовольняє всім вимогам, які ставилися на етапі постановки завдання.

СПИСОК ЛІТЕРАТУРИ

1. React: Up & Running. 2-ге вид. Stoyan Stefanov. 2021.
2. JavaScript повний посібник. 7-ме вид. Фленаган Д. 2020.
3. Learning React: Functional Web Development with React and Redux. 2-ге вид. Eve Porcello, Alex Banks. 2017.
4. Using SQLite. Jay A. Kreibich. 2010.
5. React in Action. 2-ге вид. Mark Tielens Thomas. 2018.
6. https://reactjs.org/docs/getting-started.html

ДОДАТКИ

Додаток А.1:

```
const Employer = () => {
  const { logIn, setUser } = useStore();
  const navigate = useNavigate();

  const form = useForm({
    defaultValues: {
      fullName: '',
      email: '',
      password: ''
    }
  });

  const onSubmit = async (data) => {
    const response = await registerUser(data);
    if (response.data) {
      logIn(response.data.jwt);
      setUser(response.data.user);
      navigate('/', { replace: true });
    }
  };

  return (
    <Container>
      <StyledTypography variant="h1" bold>
        Register new employer!
      </StyledTypography>
      <FormProvider {...form}>
        <form autoComplete="off" noValidate>
          <Input disableIndents label="Full name" name="fullName" />
          <Input label="Username" name="username" />
          <Input label="Email" name="email" />
          <Input label="Password" name="password" />

          <StyledTypography variant="h2" bold>
            Company details
          </StyledTypography>

          <Input disableIndents label="Company name" name="company.name" />
          <Textarea label="Description" name="company.description" />

          <StyledTypography variant="h2" bold>
            Contact information
          </StyledTypography>

          <Input disableIndents label="Additional email" name="contact.email"
/>
          <Input label="Phone number" name="contact.phoneNumber" />
          <Input label="Telegram" name="contact.telegram" />
          <Input label="LinkedIn" name="contact.LinkedIn" />

          <Button variant="secondary">cancel</Button>
          <Button variant="primary" onClick={form.handleSubmit(onSubmit)}>
            submit
```

```
            </Button>
          </form>
        </FormProvider>
      </Container>
    );
  };
```

Додаток А.2:

```
const Profile = () => {
  const { user } = useStore();
  const navigate = useNavigate();

  useEffect(() => {
    if (!user) {
      navigate('/');
    }
  }, [user]);

  switch (user.role) {
    case 'job-seeker': {
      return <JobSeekerProfile />;
    }
    case 'employer': {
      return <EmployerProfile />;
    }
  }
};

const JobSeekerProfile = ({ data }) => {
  return (
    <Main>
      <Typography variant="h1" bold>
        My profile
      </Typography>
      <Container>
        {data.map((item, i) => (
          <ItemContainer key={i}>
            <ItemName>
              <Typography variant="h2" bold>
                {item.label}
              </Typography>
            </ItemName>
            <ItemValue>
              <Typography variant="h2">{item.value}</Typography>
            </ItemValue>
          </ItemContainer>
        ))}
      </Container>
    </Main>
  );
};

const EmployerProfile = () => {
  return (
    <Main>
      <HeaderContainer>
```

```
  <Typography variant="h1" bold>
    My profile
  </Typography>

  <StyledEdit variant="h3" bold>
    Edit profile
  </StyledEdit>
</HeaderContainer>
<Container>
  {data.map((item, i) => (
    <ItemContainer key={i}>
      <ItemName>
        <Typography variant="h2" bold>
          {item.label}
        </Typography>
      </ItemName>
      <ItemValue>
        <Typography variant="h2">{item.value}</Typography>
      </ItemValue>
    </ItemContainer>
  ))}
</Container>
<Typography variant="h1" bold>
  Company details
</Typography>
<Container>
  <ItemContainer>
    <ItemName>
      <Typography variant="h2" bold>
        Company Name
      </Typography>
    </ItemName>
    <ItemValue>
      <Typography variant="h2">{data.company.name}</Typography>
    </ItemValue>
  </ItemContainer>
  <ItemContainer>
    <ItemName>
      <Typography variant="h2" bold>
        Description
      </Typography>
    </ItemName>
    <ItemValue>
      <Typography variant="h2">{data.company.details}</Typography>
    </ItemValue>
  </ItemContainer>
</Container>
<Typography variant="h1" bold>
  Open vacations
</Typography>
<Container>
  {data.vacations.map((item) => (
    <OpenPositionCard>
      <Typography variant="h3" bold>
        {item.title}
      </Typography>
      <Typography variant="h3">{item.description}</Typography>
      <br />
```

```
            <ItemContainer>
              <ItemName>
                <Typography variant="p" bold>
                  English level
                </Typography>
              </ItemName>
              <ItemValue>
                <Typography variant="p">{item.englishLevel}</Typography>
              </ItemValue>
            </ItemContainer>

            <ItemContainer>
              <ItemName>
                <Typography variant="p" bold>
                  Work experience
                </Typography>
              </ItemName>
              <ItemValue>
                <Typography variant="p">{item.workExperience}</Typography>
              </ItemValue>
            </ItemContainer>

            <ItemContainer>
              <ItemName>
                <Typography variant="p" bold>
                  Location
                </Typography>
              </ItemName>
              <ItemValue>
                <Typography variant="p">{item.location}</Typography>
              </ItemValue>
            </ItemContainer>

            <ItemContainer>
              <ItemName>
                <Typography variant="p" bold>
                  Technology stack
                </Typography>
              </ItemName>
              <ItemValue>
                <Typography variant="p">
                  {item.technologyStack}
                </Typography>
              </ItemValue>
            </ItemContainer>
          </OpenPositionCard>
        ))}
      </Container>
    </Main>
  );
};
```

Додаток А.3:

```
const CreateVacancy = () => {
  const navigate = useNavigate();

  const form = useForm({
```

```
    defaultValues: {
      title: '',
      description: '',
      salary: '',
      englishLevel: '',
      location: '',
      workExperience: '',
      technologyStack: ''
    }
  });

  const onSubmit = async (data) => {
    const response = await createVacancy(data);
    if (response.data) {
      navigate('/company', { replace: true });
    }
  };

  return (
    <Container>
      <StyledTypography variant="h1" bold>
        Create vacancy
      </StyledTypography>
      <FormProvider {...form}>
        <form autoComplete="off" noValidate>
          <Input disableIndents label="Title" name="title" />
          <Textarea label="Description" name="description" />
          <Input label="Salary" name="salary" />
          <Input label="English level" name="englishLevel" />
          <Input label="Location" name="location" />
          <Input label="Work experience" name="workExperience" />
          <Input label="Technology stack" name="technologyStack" />

          <Button variant="secondary">cancel</Button>
          <Button variant="primary" onClick={form.handleSubmit(onSubmit)}>
            submit
          </Button>
        </form>
      </FormProvider>
    </Container>
  );
};
```

Додаток А.4:

```
const Search = () => {
  const form = useForm({});

  const [data, setData] = useState([]);

  const handleChange = (e) => {
    const response = getCompanies({ search: e.value });
    setData(response.data);
  };

  useEffect(() => {
    const response = getCompanies();
```

```
    setData(response.data);
  }, []);

  return (
    <Main>
      <FormProvider {...form}>
        <Input label="Search company" name="search" onChange={handleChange} />
      </FormProvider>
      {data.map((item) => (
        <CompanyItem>
          <Typography variant="h3" bold>
            {item.companyName}
          </Typography>

          <Typography>{item.vacancies.length}</Typography>
        </CompanyItem>
      ))}
    </Main>
  );
};

export default Search;
```

Додаток А.5:

```
const CompanyInfo = () => {
  const { id } = useParams();
  const [data, setData] = useState();

  useEffect(() => {
    const response = getCompany(id);
    setData(response.data);
  }, []);

  return (
    <Main>
      <Typography variant="h1" bold>
        {data.name}
      </Typography>
      <Typography variant="h3">{data.description}</Typography>
      <br />
      <Typography variant="h2" bold>
        Contact info
      </Typography>
      <Container>
        <ItemContainer>
          <ItemName>
            <Typography variant="p" bold>
              Contact email
            </Typography>
          </ItemName>
          <ItemValue>
            <Typography variant="p">{data.contact.contactEmail}</Typography>
          </ItemValue>
        </ItemContainer>
        <ItemContainer>
```

```
    <ItemName>
      <Typography variant="p" bold>
        Telegram
      </Typography>
    </ItemName>
    <ItemValue>
      <Typography variant="p">{data.contact.telegram}</Typography>
    </ItemValue>
  </ItemContainer>
  <ItemContainer>
    <ItemName>
      <Typography variant="p" bold>
        Phone number
      </Typography>
    </ItemName>
    <ItemValue>
      <Typography variant="p">{data.contact.phoneNumber}</Typography>
    </ItemValue>
  </ItemContainer>
</Container>
<Typography variant="h1" bold>
  Open positions
</Typography>
<br />
<OpenPositionCard>
  {data.vacations.map((item) => (
    <OpenPositionCard>
      <Typography variant="h3" bold>
        {item.title}
      </Typography>
      <Typography variant="h3">{item.description}</Typography>
      <br />
      <ItemContainer>
        <ItemName>
          <Typography variant="p" bold>
            English level
          </Typography>
        </ItemName>
        <ItemValue>
          <Typography variant="p">{item.englishLevel}</Typography>
        </ItemValue>
      </ItemContainer>

      <ItemContainer>
        <ItemName>
          <Typography variant="p" bold>
            Work experience
          </Typography>
        </ItemName>
        <ItemValue>
          <Typography variant="p">{item.workExperience}</Typography>
        </ItemValue>
      </ItemContainer>

      <ItemContainer>
        <ItemName>
          <Typography variant="p" bold>
            Location
```

```
                </Typography>
              </ItemName>
              <ItemValue>
                <Typography variant="p">{item.location}</Typography>
              </ItemValue>
            </ItemContainer>

            <ItemContainer>
              <ItemName>
                <Typography variant="p" bold>
                  Technology stack
                </Typography>
              </ItemName>
              <ItemValue>
                <Typography variant="p">{item.technologyStack}</Typography>
              </ItemValue>
            </ItemContainer>
          </OpenPositionCard>
        ))}
      </OpenPositionCard>
    </Main>
  );
};
```