МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет водного господарства та природокористування

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:

Завідувач кафедри

_____д. е. н., проф. П. М. Грицюк

«_____» _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «бакалавр»

за освітньо-професійною програмою «Інформаційні системи і технології» спеціальності 126 «Інформаційні системи та технології»

на тему: «**Розробка та впровадження CRM-систем для підвищення ефективності діяльності рекрутингового агентства**»

Виконав:

здобувач вищої освіти 4 курсу, групи ІСТ-41

Кравчук Вікторія Олександрівна

Керівник: к. т. н. доцент Барановський Сергій Віталійович

Рецензент: к. т. н. доцент Гладка Олена Миколаївна

Рівне – 2022

Анотація

Тема: «Розробка та впровадження CRM-систем для підвищення ефективності діяльності рекрутингового агентства»

Студентка: Кравчук Вікторія Олександрівна

Науковий керівник: Барановський Сергій Віталійович

Місто: Рівне

Рік захисту: 2022

Дипломна робота присвячена актуальній проблемі розробки та впровадження сучасних CRM-систем в діяльності рекрутингового агентства.

Діджиталізація – незворотній процес в 21 столітті. За статистикою, 94% молоді в Європі мають доступ до інтернету та більшість інформації проходить саме через інтернет. За допомогою інтернет-технологій робота, на яку потрібно було витрачати чималу кількість годин, за допомогою інтернету може бути зроблена кількома кліками миші.

У першому розділі дипломної роботи описано технічну сторону проєкту. Тут наведено характеристику використаних технологій та обґрунтовано вибір тих інструментів, які використані для реалізації проєкту. Плюси вибраних технологій та те, як ми будемо їх використовувати на проєкті.

Другий розділ присвячений опису процесу розробки проєкту. Цей розділ багатий на ілюстрації з кодом, який був використаний при створенні проєкту.

У третьому розділі наведено опис функціоналу створеного проєкту, які супроводжуються ілюстраціями з результатом роботи. Також у дипломній роботі зроблено висновок щодо основних результатів.

ЗМІСТ

ВСТУП.....4
РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ РОЗРОБКИ ТА ВПРОВАДЖЕННЯ CRM-СИСТЕМ.....6
1.1. Застосування CRM-систем для автоматизації інформаційних функцій рекрутингових агентств.....6
1.2. Проєктування та розробка ІТ-додатків. Життєвий цикл розробки.....7
1.3. Сучасні вимоги до ІС та ІТ-додатків.....10
1.4. Характеристика технологій розробки ІТ-проєктів.....11
РОЗДІЛ 2. ЗАСТОСУВАННЯ LARAVEL ДЛЯ РОЗРОБКИ ІТ-ДОДАТКІВ.....16
2.1. Кібербезпека.....16
2.2. Робота з базою даних.....18
2.3. Eloquent ORM.....20
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІТ-ПРОЄКТУ.....25
3.1. Загальна характеристика функціоналу ІТ-проєкту.....25
3.2. Внесення, зберігання та використання інформації про робітника.....26
3.3. Внесення, зберігання та використання документів.....30
3.4. Внесення та використання інформації про працевлаштування.....36
ВИСНОВОК.....39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:.....40

ВСТУП

Актуальність. Діджиталізація – незворотній процес в 21 столітті. За статистикою, 94% молоді в Європі мають доступ до інтернету, та більшість інформації проходить саме через інтернет. За допомогою інтернет-технологій, робота, на яку потрібно було витрачати чималу кількість годин, за допомогою інтернету може бути зроблена кількома кліками миші.

Результатом імплементацій інтернет-рішень у робочу галузь є не тільки шляхом в майбутнє, а й покращенням продуктивності та показників компанії за рахунок автоматизації процесів. Автоматизація процесів дозволяє зменшити ресурси, та пришвидшити час здійснення тієї чи іншої операції.

Імплементуванню інтернет-рішень у компанії немає лімітів. Можна розпочинати зі створення звичайних інтернет-сторінок та закінчувати більш складними CRM-системами та створенням роботів (інтернет-речей).

CRM-системи дозволяють автоматизувати та оптимізувати левову частину роботи покращуючи продуктивність компаній. Також використання CRM-систем допомагає зменшити вірогідність помилок людського фактору та зменшити необхідність використання паперу (знайоме як paperless).

Оскільки технології швидко еволюціонують, при побудуванні CRM-систем потрібно робити правильний вибір технології, на якій буде побудована система. Потрібно бути впевненим, що вибрана технологія зможе забезпечувати надійність, безпеку та сучасність наступних мінімум 10 років.

Метою роботи є побудова CRM-системи для рекрутингового агентства на базі сучасного фреймворку, бізнес вимоги якої були б сформовані аналізом сучасного ринку таких агентств та їх бізнес потреб.

Для досягнення цієї мети визначені такі **завдання роботи:**

- аналіз ринку рекрутингових агентств;
- дослідження бізнес потреб та вимог рекрутингових агентств;

- вибір технології для побудування проєкту відповідно до бізнес вимог;
- будування проєкту відповідно до бізнес вимог;
- описання висновку роботи.

Об’єктом дослідження є ринок послуг рекрутингових агентств.

Предметом дослідження є функції рекрутингових агентств, які можуть бути автоматизовані за допомогою CRM-системи.

Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел.

РОЗДІЛ 1. СУЧАСНІ ПІДХОДИ РОЗРОБКИ ТА ВПРОВАДЖЕННЯ CRM-СИСТЕМ

1.1. Застосування CRM-систем для автоматизації інформаційних функцій рекрутингових агентств

Ринок праці з кожним роком росте. Компанії намагаються більше приділяти час саме для внутрішніх процесів, тому змушені шукати рішення для найму працівників на ринку. Рішенням цієї проблеми виступають рекрутингові агентства, які допомагають в пошуку та найманні працівників.

Ринок послуг рекрутингових агенств зі свого боку також росте, що створює в собі потребу в інтернет-рішенні для легкого старту нових агентств.

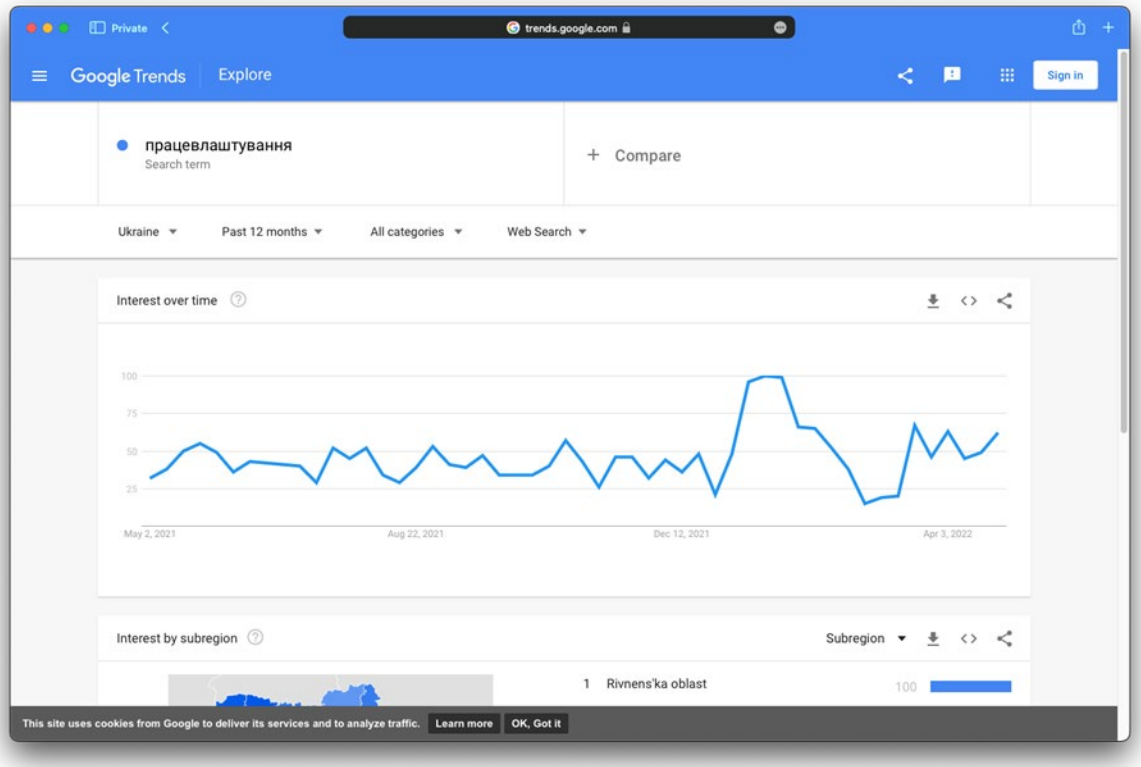

Рис. 1.1. Аналіз ринку праці

Однією та, мабуть, найважливішою потребою кожного рекрутингового агентства є потреба в зручному менеджменті компаній та охочих влаштуватись на

роботу. Саме це і створює попит для зручних CRM-систем, які могли б взяти вирішення цих проблем на себе.

Впроваджуючи CRM-систему, рекрутингові агентства в кінцевому рахунку матимуть змогу для легкого розширення, яке залежатиме лише від кількості вакансій. При великому напливі не потрібно буде мати величезні шафи з документами та створювати живі черги в офісах. Все це може бути вирішено за допомогою CRM-системи.

Ця CRM-система допоможе автоматизувати рішення, які використовуються більшістю рекрутинговими агентствами, як-от: база даних працівників, компаній, їх працевлаштувань та документів, які пов’язані з тим чи іншим суб’єктом.

Список функцій, які можуть бути автоматизовані шляхом впровадження CRM-додатку:

- перегляд та пошук робітників;
- створення, редагування та видалення компанії;
- створення, редагування та видалення бухгалтерів;
- створення, редагування та видалення робітників;
- створення, редагування та видалення документів робітників;
- створення робочих контрактів робітників з компаніями;
- відправка даних про робітників, та їх документів до бухгалтерії;
- завантаження документів на захищений S3-сервер;
- електронні підписи робочих контрактів компаній з робітниками;
- збереження унікальної інформації для кожного типу документів.

1.2. Проєктування та розробка IT-додатків. Життєвий цикл розробки

При розробці IT-проєктів досить часто використовується технологія MVC. MVC – архітектурний шаблон, який використовується під час проєктування та розробки різноманітних додатків. Перевагою цієї технології є те, що модифікація

кожного компоненту може відбуватись незалежно одне від іншого. MVC розшифровується як модель-вигляд-контроллер.

- *Модель* являє собою дані, та реагує на команти контролеру, змінюючи її стан.
- *Вигляд* відповідає за відображення даних моделі.
- *Контроллер* відповідає за зв'язок між моделлю та виглядом. Код контроллеру визначає, як сайт реагує на дії користувача.

Шаблон для імплементації MVC-архітектури надає Laravel. Laravel безкоштовний веб-фреймворк з відкритим кодом. Ми можемо створювати власні вигляди, де можемо описувати логіку відображення, моделі, де можемо працювати з даними, які отримуємо з бази даних, та можемо керувати цим всім за допомогою контроллеру.

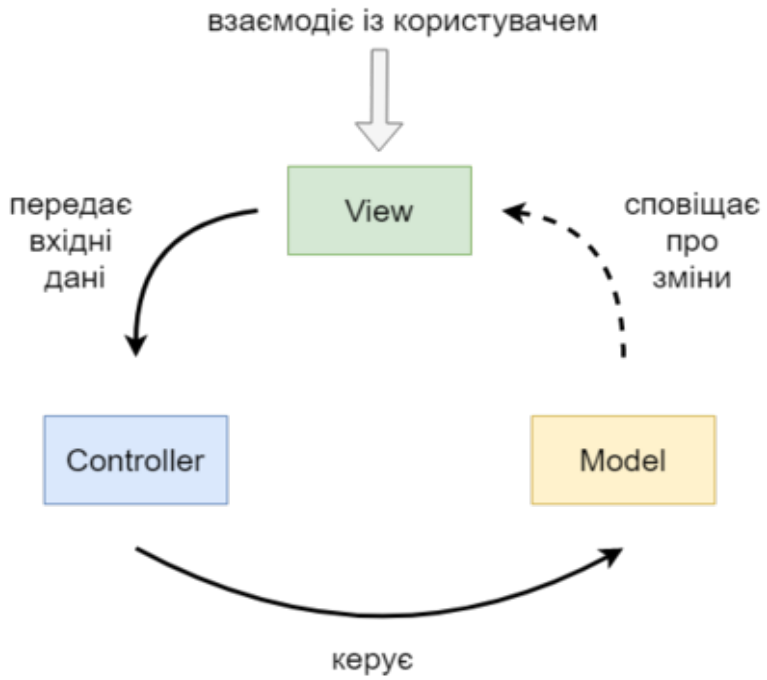

Рис.1.2. Огляд MVC-архітектури

Розробка, тестування та розгортання – три стадії, які реалізуються при розробці проєкту.

Написання та редагування коду можна здійснити в кросплатформовому текстомову редакторі Sublime, його перевагою є те що він підримує плагіни розроблені за допомогою досить широкого спектру сучасних мов програмувань. Laravel – може бути основним фреймворком при створенні додатку.

Як система контролю версій можуть бути використані Git та GitHub. Найбільш описова характеристика Git є його дистрибутивна система, у якій кожен користувач має доступ до локального репозиторію, де можуть бути зроблені локальні зміни до коду проєкту. При досягненні поставленої цілі локальні зміни можуть бути синхронізовані з віддаленим репозиторієм, тим самим роблячи ці зміни доступними для всієї команди. Віддалений репозиторій розміщується на GitHub.

Для розгортання проєкту на сервері зручно скористатися технологією Continuous Deployment. Встановлення та налаштування Continious Deployment можна зробити безпосередньо в GitHub репозиторію проєкту за допомогою GitHub Actions. Цей процес дозволяє скомпілювати проєкт, запустити тести та розгорнути його на сервері. Перевагами Continuous Deployment є запускання тестів, де при виявленні багів, розгортання проєкту на сервері буде зупинене до часу, коли будуть виправленні всі тести.

При успішній компіляції та запуску всіх тестів додаток буде розгорнуто на вказаному сервері. Якщо ж за деяких причин тести не були успішно пройдені, розгортання проєкту на сервер буде зупинено й ми отримаємо репорт від GitHub Actions про список тестів, які не пройшли перевірку.

За допомогою цього циклу розробки ми можемо мінімізувати ризик виникнення багів при розгортанні проєкту на головний сервер.

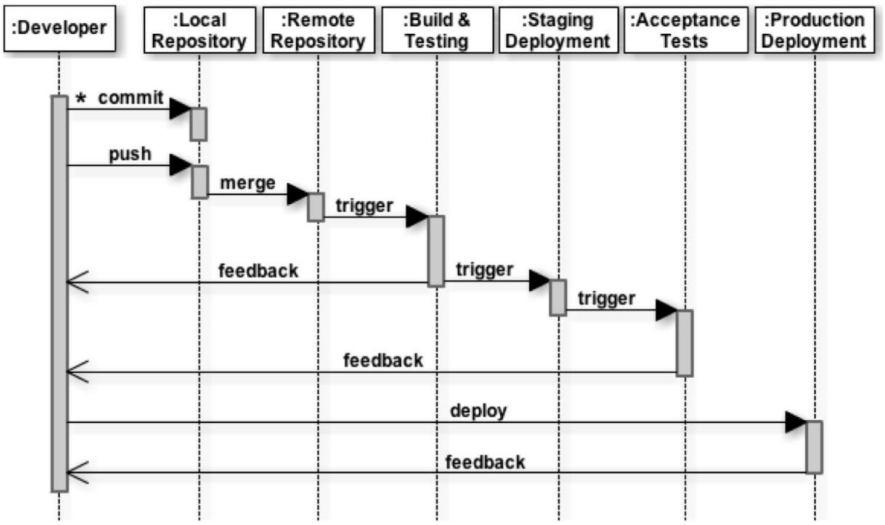

Рис.1.3. Огляд процесу Continuous deployment

1.3. Сучасні вимоги до ІС та ІТ-додатків

Вимоги щодо продуктивності:

- система має бути швидкодіючою та пристосованою до високих навантажень;
- Cross-platform. Додаток має працювати на будь-якому пристрої;
- DigitalOcean як хостинг провайдер;
- NGINGS як система балансування навантаження;
- MySQL — база даних.

Вимоги щодо кібербезпеки:

- система має використовувати захищене з’єднання з базою даних;
- документи, які зберігаються в хмарі. Мають бути надійно захищені від публічного доступу;
- робота з працівниками може бути виконана лише адміністраторами компанії.

Вимоги щодо обробки помилок:

- при непередбачуваних сценаріях інформація не має зникати;

- інформація та код помилки мають бути записані в реєстрі для подальшого вирішення проблеми.

1.4. Характеристика технологій розробки IT-проєктів

Laravel. Основою архітектурної частини є вибір веб-фреймворку для побудови проєкту. Основні вимоги при пошуку веб-фреймворку були: open-source, надійність (захист), сучасність (синтаксис за документація) та екосистема.

При пошуку веб-фреймворку за вище вказаними параметрами, вибір випав на Laravel, веб-фреймворк з MVC-архітектурою. Laravel успішно використовують такі компанії як: Twitch, The New York Times, Disney, Warner Brothers та інші.

Основною ідеєю Laravel фреймворку є зручність використання, зручний синтаксис.

Laravel використовує MVC-архітектуру за замовчуванням та дозволяє без проблем створювати CRUD системи не вигадуючи велосипедів, що дає змогу розробникам піклуватись тільки про будування бізнес логіки.

Плюси використання Laravel:

1). Велика екосистема. Екосистема в Laravel за допомогою її ком'юніті є величезна. Починаючи від звичайних open-source компонентів, які можна знайти на GitHub, є компоненти, які виділяються своєю унікальністю:

- Cashier – компонент для роботи з оплатами та оформленням підписок.
- Telescope – компонент для зручності дебагу проєкту.
- Horizon – компонент для виконання фонових завдань через Redis.
- Scout – компонент для пошуку в базі даних за алгоритмом full-text.

2). Легкий старт

Робота з Laravel починається досить легко: достатньо тільки створити новий екземпляр використовуючи команду laravel new project-name (де project-name – назва проєкту) та налаштування базових параметрів у .env файлі.

3). Легка інтеграція з front-end

Laravel має власний шаблонізатор Blade з чималою кількістю директив, які спрощують роботу на front-end частині проєкту.

Також Laravel іде з компонентом mix, який відповідає за компілювання CSS та JavaScript файлів використовуючи Webpack.

4). Кібербезпека

Laravel не потребує додаткових налаштувань безпеки. За замовчуванням Laravel містить в собі захист від найпоширеніших видів атак, таких як: SQL ін’єкція, XSS та CSRF атаки.

HTML (Hypertext Markup Language) – стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML розмітку від сервера за протоколами HTTP/HTTPS, далі інтерпретують отриманий код в інтерфейс, який відображатиметься в браузері.

HTML не є мовою програмування. Це мова розмітки, яка відповідає за відображення елементів у браузері.

Кожен HTML елемент має свою унікальну назву та параметри з визначеним синтаксисом, який записується латинськими літерами і не чутливий до регістру.

Приклад HTML розмітки:

```
<div class=”panel”>
  <p class=”panel-header”>Назва таблиці</p>
  <div class=”panel-body”>
    …вміст
  </div>
</div>
```

CSS каскадні таблиці стилів. Це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду.

Основна ціль CSS полягає в тому, щоб описати дизайн того чи іншого елементу. CSS відповідає за зовнішній вигляд та оформлення, де в свою чергу HTML відповідає за зміст та логічну структуру сторінки.

За допомогою CSS ми можемо, разом з іншими можливостями, налаштовувати колір фону, розмір шрифту, та чимало інших налаштувань.

Під'єднання CSS файлу можна здійснити в <head> тегові кожної html сторінки:

<link rel = "stylesheet" type = "text / css" href = "style.css" />.

Вміст файлу style.css:

```
#main {
margin-left: 220px;
margin-right: 200px; background-color: # 0A1646; font-size: 13px;
}
```

вміст HTML файлу:

```
<div id = "main">
  …
 </div>
```

Дизайн контенту всередині тегу з id "main" буде відповідно налаштувань в style.css файлу.

Javascript – мова програмування, що означає, що ми можемо використовувати всі її особливості: умови, цикли, порівняння тощо. У загальному, ми будемо працювати з даними на web сторінці, обробляти її для створення інтерактивності.

Javascript – мова програмування, яка використовується клієнтом. Це означає, що вона працює на стороні клієнту. Коли ми відкриваємо сайт в браузері, ми робимо це використовуючи HTTP-протокол. З нашого комп'ютеру, ми

відправляємо запит то серверу, на якому даний сайт знаходиться. Сервер, в свою чергу відправляє браузеру відповідь, яка зазвичай є HTML-розмітка. У сумі це працює як клієнт-серверна архітектура.

Клієнтом є наш локальний комп'ютер, з якого ми працюємо. Сервер – це віддалений комп'ютер, на якому цей сайт знаходиться. Javascript своєю чергою може виконатись лише на клієнті, у нашому випадку – у браузері.

Використовуючи Javascript, ми маємо змогу будувати різних типів слайдери, каруселі, анімації та інше.

Для підсумку, при створенні веб-сторінок, HTML дозволяє там описати сторінку елементами. Використовуючи CSS технологію, ми можемо описати дизайн для кожного елементу, описанного в HTML-файлі: колір, шрифт, позиція та інші налаштування. Та Javascript – логіка веб-сторінки та створення інтерактивних елементів на сторінці.

MySQL Вибір бази даних випав на MySQL. MySQL є однією з найпопулярніших баз даних, яка високо цінується розробниками, які працюють з базами даних. Вона характеризується високою продуктивністю, та дуже зручно інтегрована в Laravel. MySQL доступна за ліцензією GNU GLP, що дозволяє вільно користуватись нею на даному проєкті.

MySQL в порівнянні з іншими реляційними базами даних такі як PostgreSQL чи Firebird, є швидшою, проте інші бази даних, такі як PostgreSQL мають більше можливостей та функціоналу. Незважаючи на це MySQL продовжує бути однією з найбільш популярних типів бази даних.

Бази даних зазвичай використовують для зберігання різного типу інформації, розпочинаючи від інформації про користувачів, які користуються проектом, та додатковою інформацію, яка тим чи іншим чином пов'язана із зареєстрованими користувачами.

За допомогою бази даних ми можемо швидко отримати доступ до записаної інформації, сортувати її, та проводити пошуки за вказаними критеріями.

Таблиці в базі даних складаються з колонок та рядків з інформацією, де кожна колонка має свій особистий тип та розмір.

Використовуючи колонки як посилання, можна створити відношення рядків між різними таблицями. Даний тип з’єднання має назву зв’язок між таблицями (relationship). Такі зв’язки можуть бути: один-до-одного, один-до-багатьох, багато-до-багатьох тощо. За допомогою зв’язків ми можемо швидко отримувати інформацію, яка пов’язана між собою, але знаходиться в різних таблицях.

У MySQL існує декілька рушіїв, які по різному виконують роботу з інформацією. Найбільш популярними є MyISAM та InnoDB. Де перший тип є стандартним рушієм, який не підтримує транзакції, проте містить full-text пошук в порівнянні з іншими рушіями. InnoDB зі своєї сторони, підтримує транзакції.

РОЗДІЛ 2. ЗАСТОСУВАННЯ LARAVEL ДЛЯ РОЗРОБКИ IT-ДОДАТКІВ

2.1. Кібербезпека

Захист є критично важливим аспектом при побудові будь-якого додатку. Саме тому при виборі фреймворку для побудови даного додатку було критично важливо, аби фреймворк містив в собі захист від найбільш популярних кібер атак за замовчуванням.

Cross site scripting (XSS) захист. XSS – вид атак, які дозволяють користувачам записати код клієнт частини в базі даних, яка зрозуміла браузеру. Зазвичай проводячи такі атаки, користувачі намагаються передати JavaScript код у форму, який з рештою буде записаний в базі даних, та потім виведений користувачеві у браузері.

Laravel використовує шаблонізатор Blade, який за замовчуванням має вбудований захист від XSS-атак, та при генерації даних з бази даних, всі скрипти будуть виведені лише як текст та не будуть виконані браузером.

Cross site request forgery (CSRF) захист. При відсутності захисти від CSRF-атак, сервер не може здійснити перевірку чи користувач насправді здійснював той чи інший POST/PUT/PATCH/DELETE HTTP запит.

CSR-захист в Laravel працює за допомогою VerifyCsrfToken middleware, який звіряє значення поля _token, яке було передано в POST/PUT/PATCH/DELETE HTTP запиті. І при невідповідному _token коду, сервер має змогу дізнатись чи справді саме авторизований користувач здійснював той чи інший запит. При невідповідності токену, цей запит не буде виконаний.

Для захисту форм від CSRF-атак, використовується директива @csrf, яка генерує input поле зі значенням випадкового токену CSRF-типу. При відсутності поля _token в формі, Laravel за замовчуванням не дозволить виконання даної форми.

Захист від SQL-ін’єкцій. SQL-ін’єкція – один з поширених способів злому сайтів та програм, що працюють з базами даних, заснований на впровадженні в запит довільного SQL-коду. Відсутність захисту від SQL-ін’єкцій є суттєва загроза для будь-якої програми.

SQL-ін’єкції можливі при некоректній обробці вхідних даних, що використовуються в SQL-запитах, тим самим запити, які йдуть в до бази даних, можуть бути легко змінені за допомогою вхідних даних.

Робота з базою даних у Laravel здійснюється за допомогою патерну Active Record, який за замовчуванням використовує PDO для з’єднання з базою даних, що в свою чергу забезпечує захист від SQL-ін’єкцій за замовчуванням.

Захист від clickjacking. Clickjacking – тип атак, який дозволяє шкідливій сторінці виконати дію на сайті, на який іде clickjacking-атака, використовуючи вже авторизованого користувача.

Захист від даних тип атак здійснюється за допомогою ClickjackProtection middleware, який проводить перевірку хеадеру X-Frame-Options. При невдалій перевірці виконання форми не буде можливим.

SSL/HTTPS. В Laravel за допомогою Foge можна налаштувати деплой веб додатку з автоматичним налаштуванням та продовженням SSL-сертифікатів, даючи змогу користувачам використовувати з’єднання HTTPS. Без цього з’єднання, з’являється вікно для перехоплення інформації між клієнтом та сервером.

Referrer policy. Браузери використовують Referer header як шлях відправки інформації до сайту про те, як користувачі перейшли до тої чи іншої сторінки. У Laravel можна налаштовувати Referrer Policy для відправки його лише в певних умовах, тим самим забезпечуючи приватність користувачів.

2.2. Робота з базою даних

Робота з базою даних є невід'ємною частиною майже кожного проєкту. У Laravel робота з базою даних здійснюється за допомогою Active Record паттерну та має назву Eloquent.

Окрім великого набору функціоналу, в Eloquent окремо хочеться виділити наступні опції:

1). Query builder – дуже гнучкий API для побудови запитів до бази даних в легкому, зрозумілому синтаксисі.

2). Легка робота з зв'язками:

a. One-to-one

b. One-to-many

c. Many-to-many

d. Polymorphic зв'язки

3). Кешування результатів запитів

Початок роботи з базою даних у Laravel розпочинається з налаштування з'єднання з базою даних.

Laravel підтримує чимало тип баз даних, проте даний проєкт побудований за допомогою MySQL. За замовчуванням Laravel працює з MySQL, та для конфігурації з'єднання, нам достатньо просто вказати параметри з'єднання з базою даних в загальному файлі налаштувань проєкту .env, який знаходиться в головній директорії кожного Laravel проєкту.

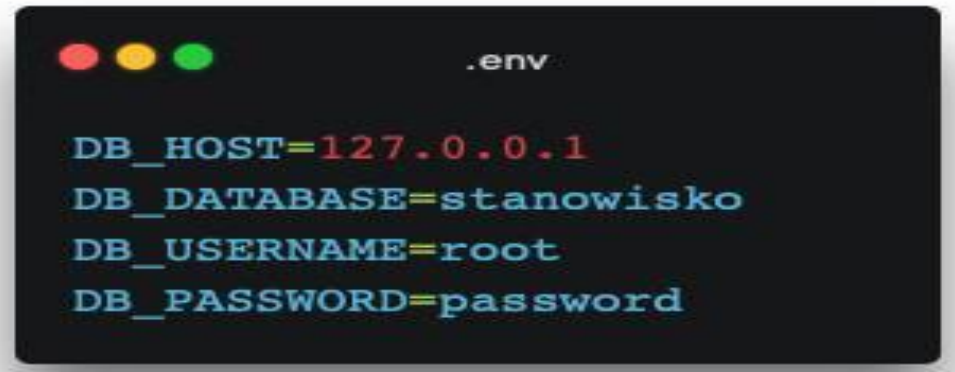

Рис.2.1. Параметри з’єднання з базою даних

Пояснення:

DB_HOST – хост адреса бази даних.

DB_DATABASE – ім’я бази даних для з’єднання.

DB_USERNAME – логін користувача в базі даних.

DB_PASSWORD – пароль користувача бази даних.

Після налаштувань параметрів з’єднання з базою даних ми маємо запевненитись у правильній конфігурації, для цього потрібно скористатись командою в терміналі:

laravel db

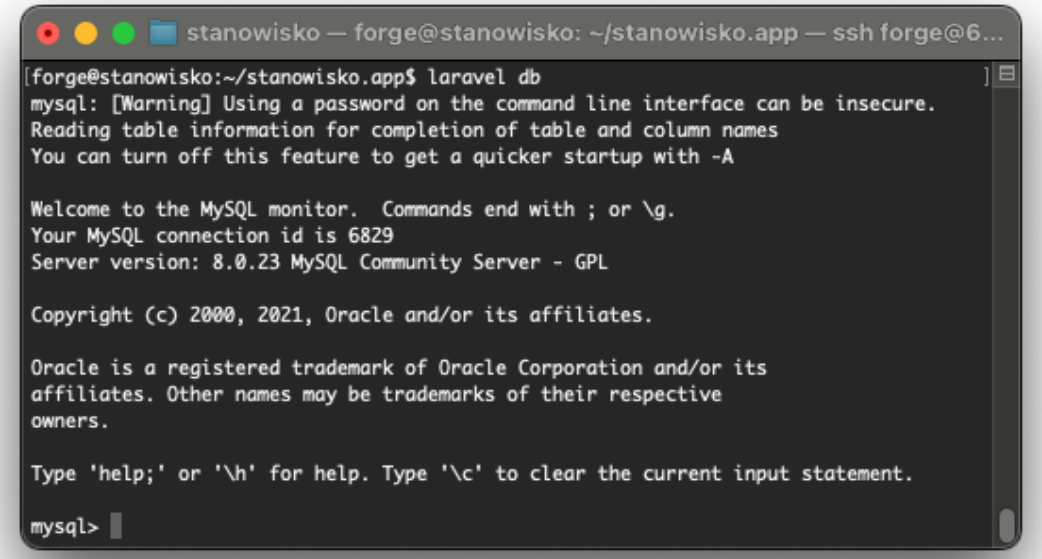

Рис.2.2. З’єднання з базою даних

При правильній конфігурації ми успішно будемо з’єднанні з базою даних.

При допущенні помилок у конфігурації ми не зможемо з’єднатись з базою даних, та відповідне повідомлення з помилкою буде виведено на екран в терміналі.

2.3. Eloquent ORM

У Laravel за допомогою Eloquent ORM ми будемо працювати з моделями. Модель – об’єкт, який репрезентує собою дані (та способи роботи з даними) з таблиці та з рядку таблиці бази даних.

Моделі в Eloquent – дуже гнучкі, що дозволяє з легкістю налаштувати.

Кожна модель має свої унікальні налаштування. У них ми можемо описати потрібні налаштування, як-от: відношення до інших моделей: one-to-one, one-to-many тощо, ацессори, мутатори, та інші хуки.

User

```php
class User extends Authenticatable
{
    use HasApiTokens;
    use HasTeams;
    use Notifiable;
    use TwoFactorAuthenticatable;

    /*
     * The table associated with the model.
     */
    protected string $table = 'users';

    /*
     * The attributes that are mass assignable.
     */
    protected array $fillable = [
        'name', 'email', 'password',
    ];

    /*
     * The attributes that should be hidden for arrays.
     */
    protected array $hidden = [
        'password',
        'remember_token',
        'two_factor_recovery_codes',
        'two_factor_secret',
    ];

    /*
     * The attributes that should be cast to native types.
     */
    protected array $casts = [
        'email_verified_at' => 'datetime',
    ];

    /*
     * The accessors to append to the model's array form.
     */
    protected array $appends = [
        'profile_photo_url',
    ];
}
```

Рис.2.3. Модель User

Модель **Worker** репрезентує собою робітника, де зберігаються загальні дані про нього, та має зв’язок з іншими моделями: **Employment** та **Document**.

Worker

```php
class Worker extends Model
{
    use HasFactory, SoftDeletes;

    protected $table = 'workers';

    protected $attributes = [
        'is_student' => false,
    ];

    protected $casts = [
        'is_student' => 'bool',
    ];

    public function documents()
    {
        return $this->morphMany(Document::class, 'subject');
    }

    public function employments()
    {
        return $this->hasMany(Employment::class, 'worker_id', 'id');
    }
    public function isStudent(): bool
    {
        return $this->is_student;
    }

    public function getDocument(string $type): ?Document
    {
        return $this->documents->withType($type);
    }

    public function fullName(): string
    {
        return str($this->first_name)
            ->append(' ')
            ->append($this->last_name);
    }
}
```

Рис.2.4. Модель Worker

Модель **Employment** пов’язана з таблицею працевлаштувань, яка зберігає дані про кожне працевлаштування, та зв’язок з компанією та працівником.

Employment

```php
class Employment extends Model
{
    use HasFactory, SoftDeletes;

    protected string $table = 'employments';

    protected array $fillable = [
        'worker_id',
        'company_id',
        'started_at',
        'finished_at',
    ];

    public function worker()
    {
        return $this->belongsTo(Worker::class, 'worker_id', 'id');
    }

    public function company()
    {
        return $this->belongsTo(Company::class, 'company_id', 'id');
    }

    public function isFinished(): bool
    {
        return now()->endOfDay()->gt($this->finished_at->endOfDay()) || !
is_null($this->closingAgreement());
    }

    public function digitalAgreement(): ?File
    {
        $digitalAgreement = $this->documents->digitalAgreement()->first();

        return optional($digitalAgreement, fn ($digitalAgreement) =>
$digitalAgreement->files->first());
    }

    public function isSigned(): bool
    {
        return (bool) optional($this->digitalAgreement())->isSigned();
    }
}
```

Рис.2.5. Модель Employment

Ця модель має додаткові методи для роботи з кожним з працевлаштувань:

- **isFinished(): bool** – перевіряє чи дане працевлаштування було завершене.
- **isSigned(): bool** – вказує на те, чи був підписаний даний контракт робітником.
- **digitalAgreement(): ?File** – дістає файл електронного підпису контракту.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІТ-ПРОЄКТУ

3.1. Загальна характеристика функціоналу ІТ-проєкту

Можливості працівника агентства в CRM системі:

- менеджмент робітників.
- енеджент контрактів з робітниками;
- робота з бухгалтерами.

Можливості адміністратора агентства в CRM системі:

- менеджмент працівників;
- менеджмент бухгалтерів;
- менеджмент компаній.

Дашборд – сторінка, на яку користувач потрапляє після авторизації на сайті. На дашборді знаходиться загальна інформація агенства працевлаштування. Оскільки будь-яке агентство нерухомості може мати велику кількість працівників, ця сторінка містить загальну інформацію, яка несе за собою статистичні дані та дані, які є найбільш важливими.

Аналізуючи інформацію, яка є важливою, та збір якої міг би зайняти деякий час, було створено два спеціальні віджети для швидкого доступу до загальної інформації:

1. У лівій колонці – список робітників, які мають активні контракти, час яких скоро завершується.
2. У правій колонці – список робітників, у яких контракт закінчилась.

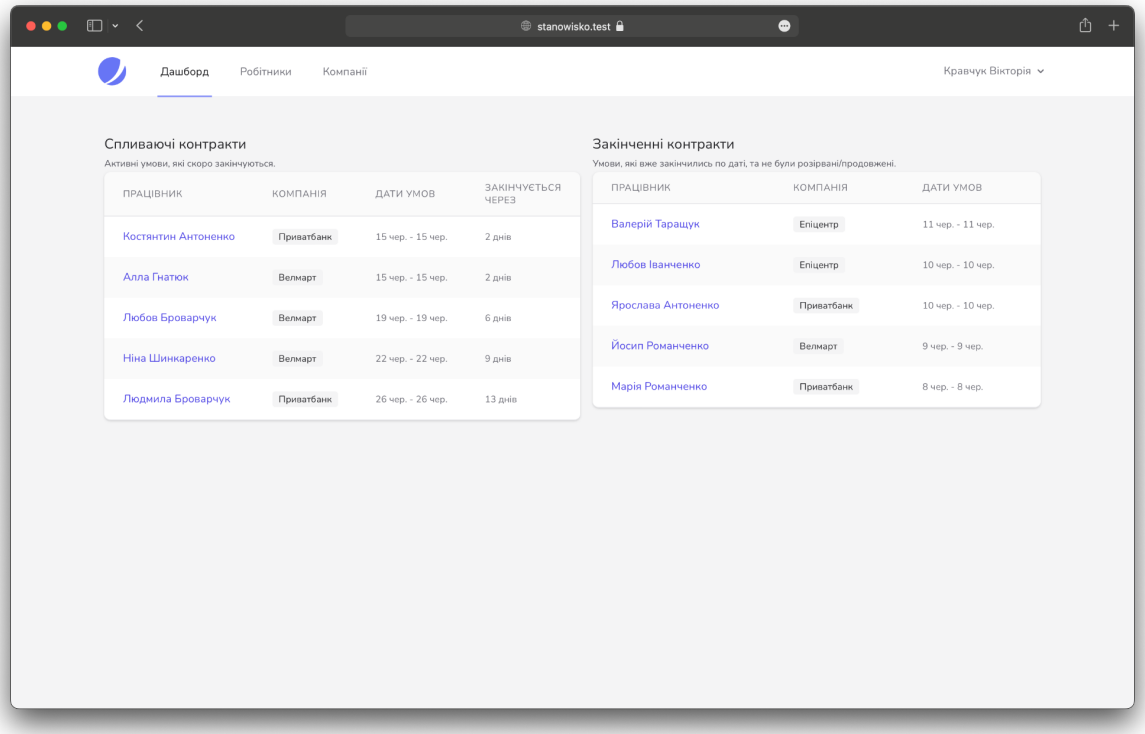

Рис.3.1. Сторінка Дашборд

3.2. Внесення, зберігання та використання інформації про робітника

Сторінка «Робітники» містить в собі таблицю, з пагінацією по 10 працівників на сторінку. У цій таблиці знаходиться список всіх працівників, які є в системі.

Для зручності використання таблиці був доданий пошук та додаткові атрибути до кожного рядка робітника в таблиці для покращення UX-дизайну.

Також на через дану таблицю можна видаляти робітників з бази даних. Та при помилковому видаленні можливо моментально відмінити дію видалення робітника для збереження інформації, яка могла б бути втрачена.

Список документів та бухгалетр, з яким пов’язаний даний працівник, також відображається в таблиці.

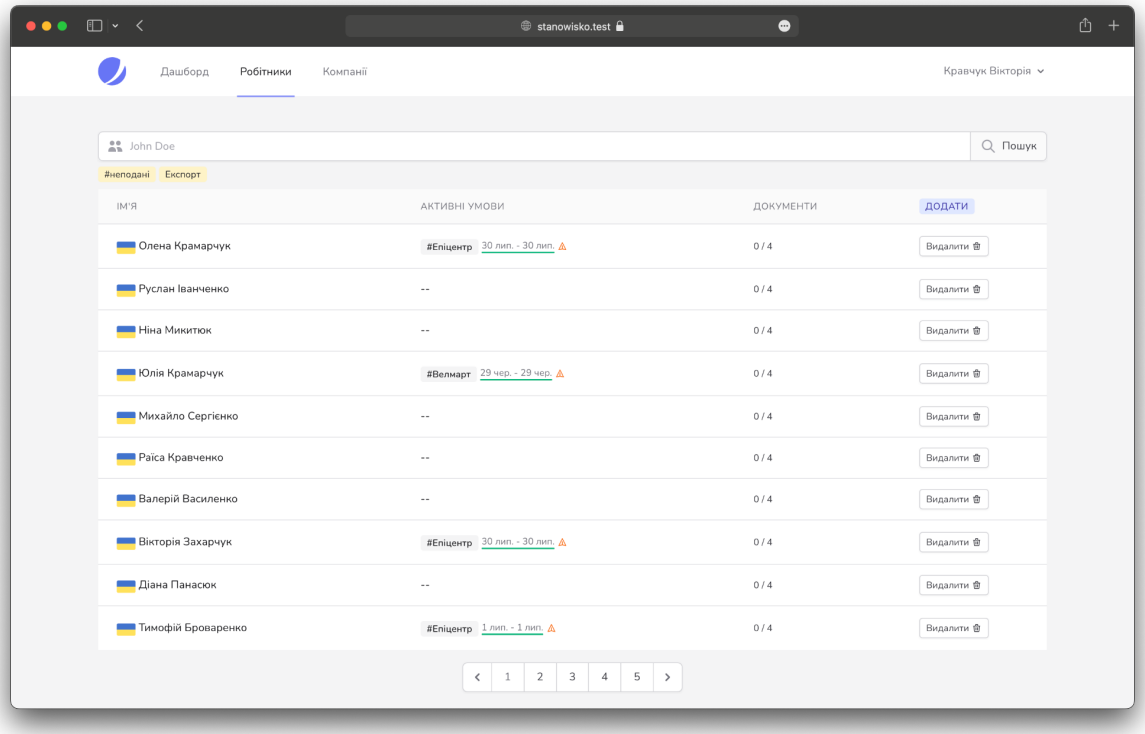

Рис.3.2. Сторінка Робітники

На цій сторінці адміністратори агенства працевлаштування розпочинають працевлаштування кожного з робітників. Спочатку за допомогою форми створюється робітник (модель User) з загальними даними про робітника: ім’я, прізвище, номер телефону, e-mail, номер рахунку в банку, стать та громадянство.

Після створення робітника він буде відображений на загальній таблиці всіх робітників, де можна буде вибрати його, та продовжити працевлаштування.

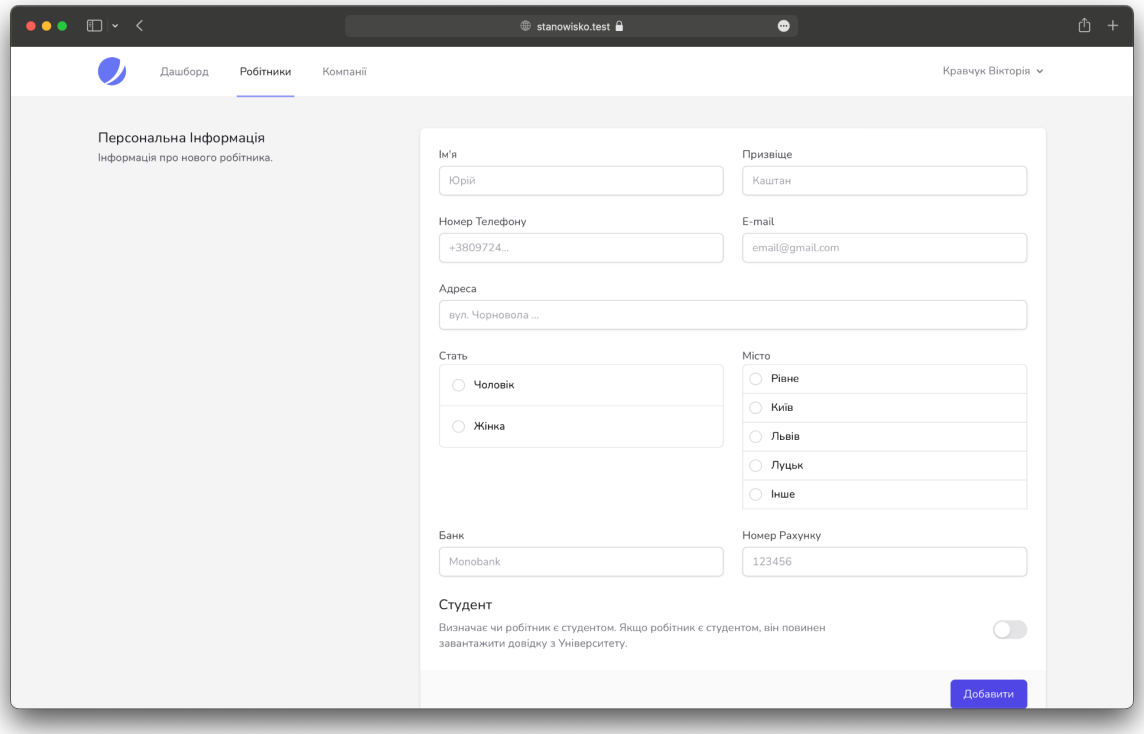

Рис.3.3. Сторінка створення робітників

Нище наведений приклад компоненту для створення робітника. У компоненті були описані три основні функції:

1) **render()** – функція, яка відповідає за рендер компоненту в браузері.

2) **getRules() –** функція, яка описує правила валідації при створенні робітника.

3) **save()** – функція, яка безпосередньо виконує валідацію введених даних, та при умові успішної валідації, створення робітника використовуючи модель User.

CreateWorker

```php
class CreateWorker extends Component
{
    public Worker $worker;

    public function render()
    {
        return view('livewire.workers.save-worker');
    }

    public function save()
    {
        $this->validate($this->getRules());

        $this->worker->save();

        return redirect(route('workers.show', [
            'worker' => $this->worker,
        ]));
    }

    protected function getRules(): array
    {
        return [
            'worker.first_name' => ['required', 'string', 'min:1', 'max:100'],
            'worker.last_name' => ['required', 'string', 'min:1', 'max:100'],
            'worker.email' => ['required', 'string', 'min:1', 'max:100', 'email'],
            'worker.gender' => ['required', 'string', 'min:1', 'max:100',
Rule::in('male', 'female')],
            'worker.country_code' => ['required', 'string', 'min:1', 'max:100',
Rule::in([
                'UA', 'PL', 'EU', 'GE', 'MD', 'KZ', 'UZ', 'ZA', 'IN',
            ])],
            'worker.is_student' => ['required', 'boolean'],
            'worker.pesel' => ['nullable', 'string'],

            'phoneNumber' => ['required', 'string', 'min:1', 'max:100'],

            'bankName' => ['nullable', 'required_with:accountNumber', 'string',
'max:255'],
            'accountNumber' => ['nullable', 'required_with:bankName', 'string'],
        ];
    }
}
```

Рис.3.4. Компонент створення працівника

Після створення робітника буде відкрита сторінка перегляду робітника, де буде знаходитись подальша інформація, яка пов’язана з робітником, а саме: умови працевлаштувань, документи, персональна інформація та інше.

Сторінка працівника містить чимало функціоналу. Є компоненти для редагування даних, створення/перегляду/видалення документів, та оновлення інформації про них, створення умов працевлаштувань для робітників та відправки даних робітника на бухгалтерію.

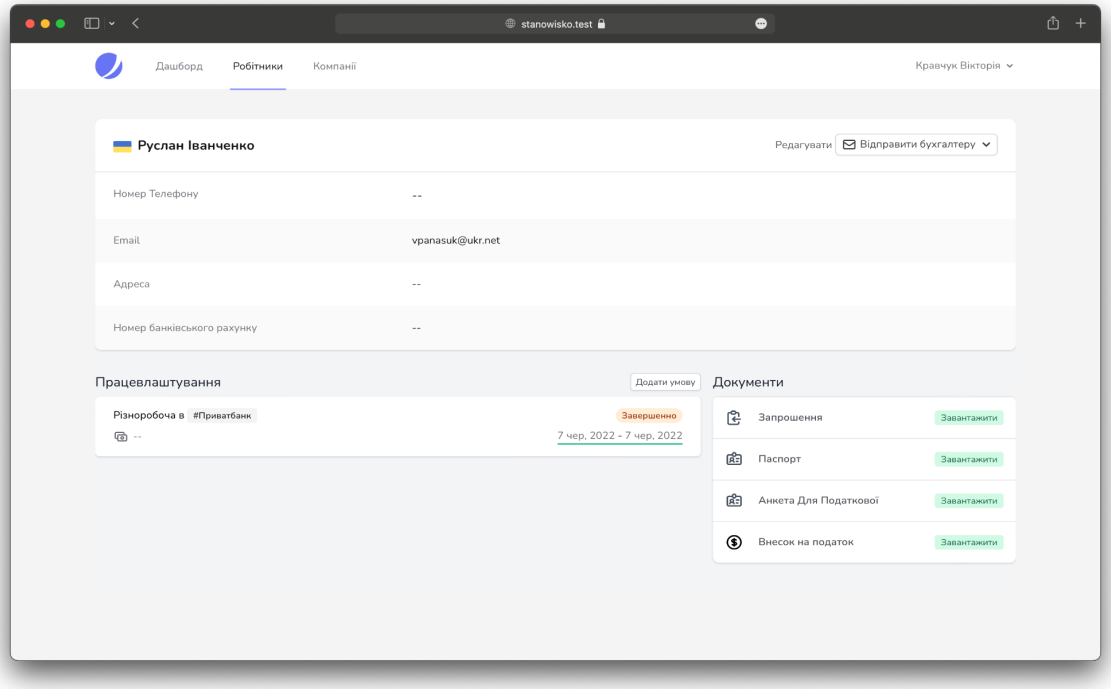

Рис.3.5. Сторінка робітника

3.3. Внесення, зберігання та використання документів

Цей компонент містить список з документів, які рекомендовано завантажити для робітника. Кожен документ має унікальні поля та може складатись з двох і більше ксерокопій. Кожен з документів може містити унікальні атрибути: паспорт може бути або звичайним, або біометричним та вказані унікальні поля будуть відображені в компоненті.

При відсутності документу, буде запропоновано завантажити документ через кнопку «Завантажити». Та при наявності документу – можливість переглянути його, завантажити і переглянути додаткову інформацію, яка була збережена при створенні документу.

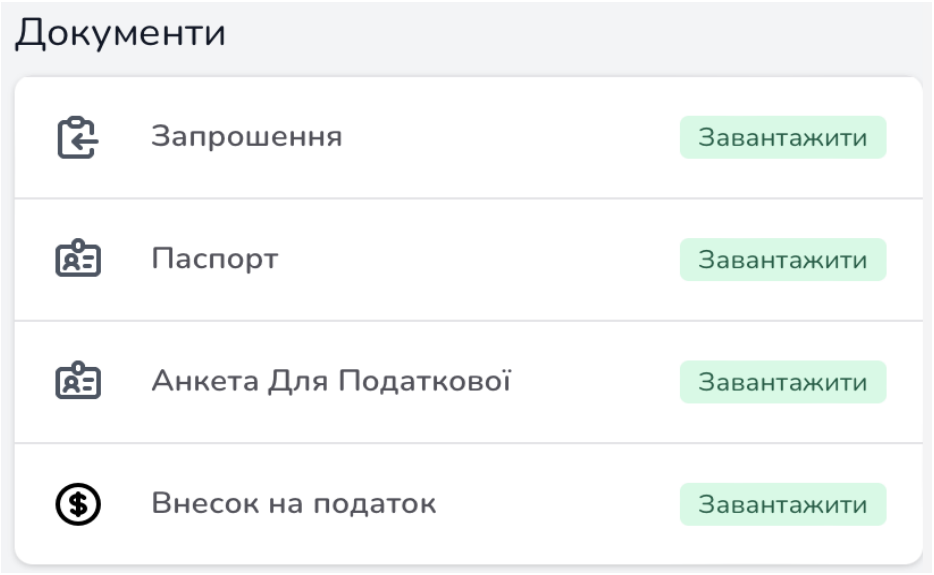

Рис.3.6. Компонент документів

Компонент спочатку отримує список всіх необхідних документів, які потрібні робітнику та для кожного з них шукає і приєднує потрібний компонент кожного з обов’язкових документів.

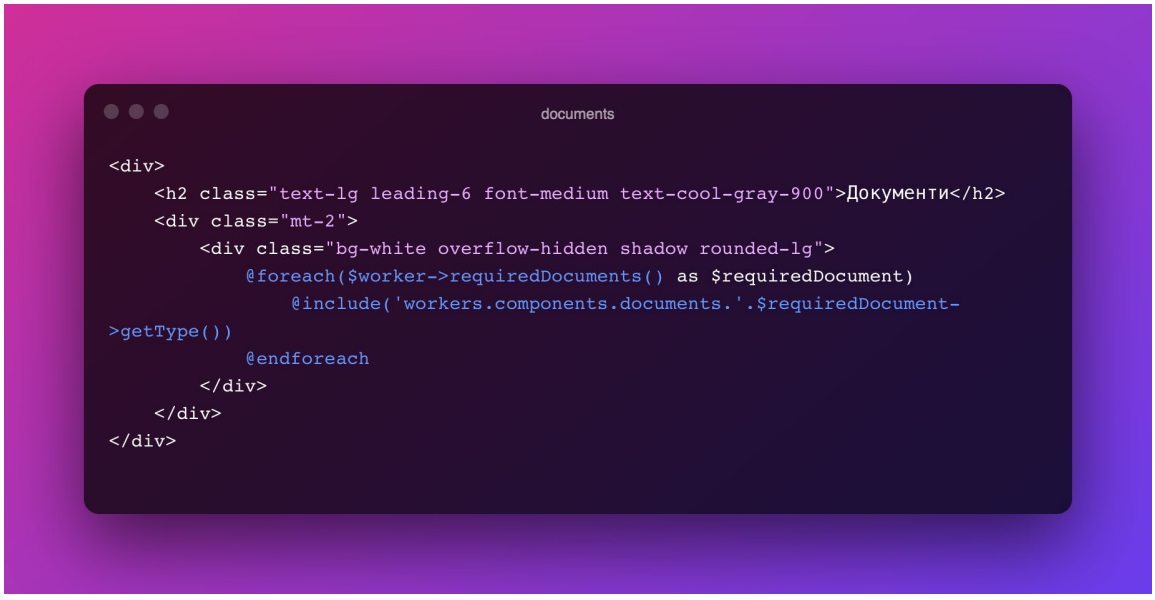

Рис.3.7. Компонент документів (код)

Компоненти кожного з документів є унікальні, щоби при створенні окремого рівня абстракції описувати унікальні деталі, які пов’язані саме з вибраним типом документа. До прикладу, в компоненті паспорту ми можемо перевіряти чи паспорт є біометричним чи звичайним.

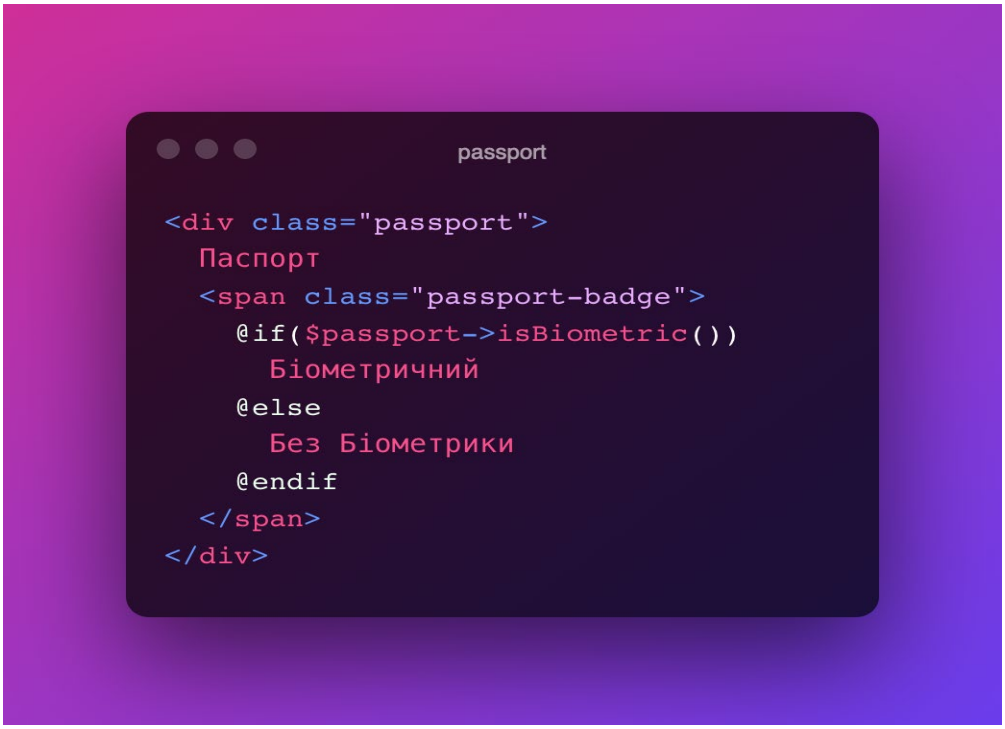

Рис.3.8. Компонент паспорту

Для зручності роботи з працівниками документи робітників зберігаються в хмарному сховищі.

При збереженні документів на хмарному сховищі критично важливим є забезпечення безпеки самих документів, щоб доступу до них не було в публічному просторі та при можливому завантаженні документів, щоб вони були зашифровані та були доступні для розшифрування лише використовуючи CRM.

Дані цілі ми можемо досягти використовуючи Storage компонент від Laravel та AWS S3 як хмарне сховище.

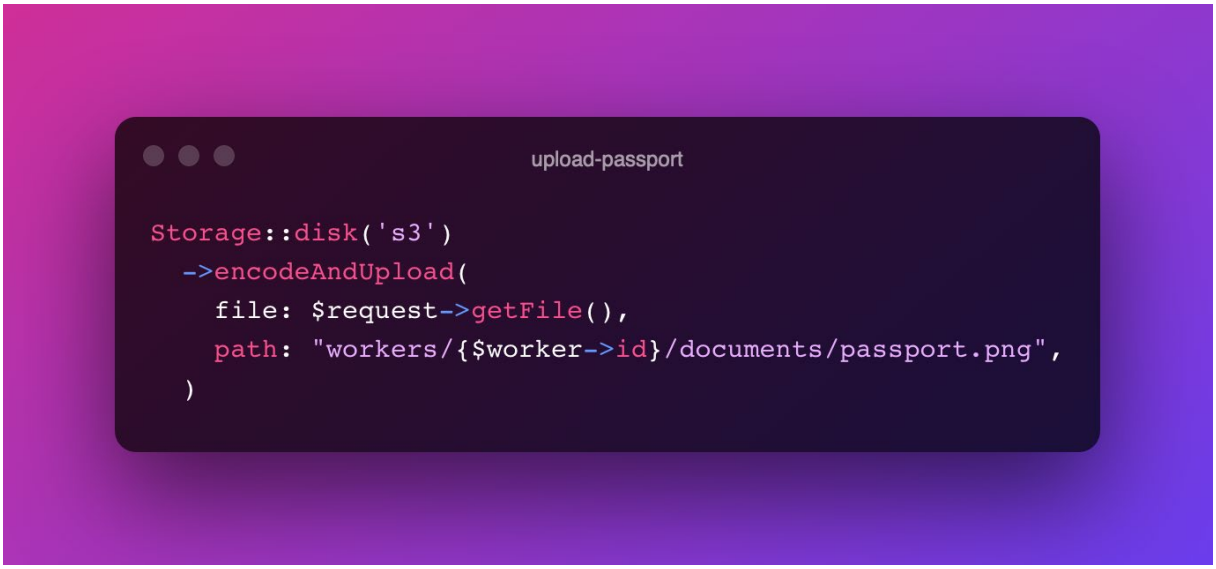

Рис.3.9. Приклад завантаження файлів

Для завантаження паспорту потрібно викликати вікно завантаження паспорту, натиснувши на кнопку «Завантажити» в рядку паспорту, вказати унікальні для даного документу дані: номер та серія паспорту, дата видачі, дійсний до та тип паспорту.

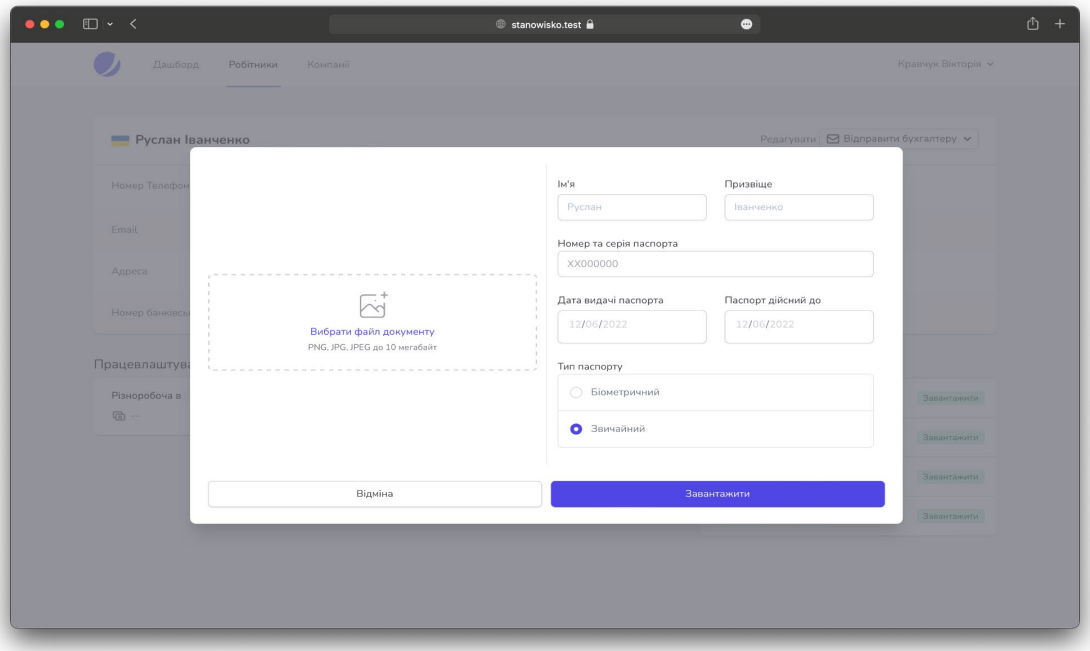

Рис.3.10. Завантаження паспорту №1

При введені даних та виборі ксерокопії паспорту можна звірити дані, та при впевненості введенні правильних даних завантажити паспорт використовуючи кнопку «Завантажити».

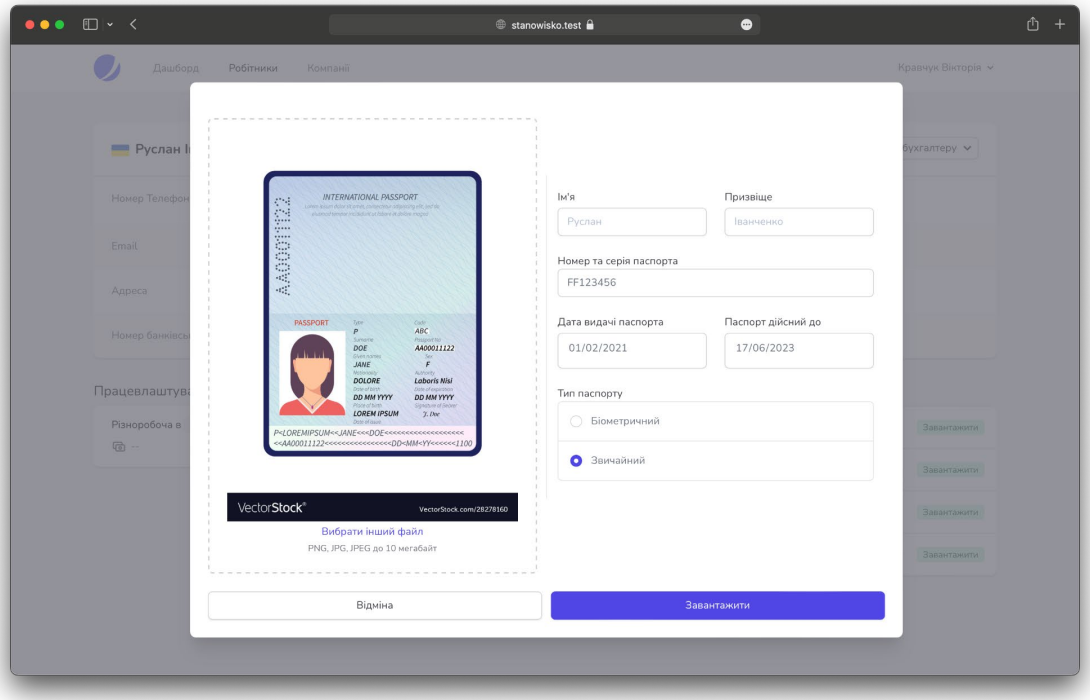

Рис.3.11. Завантаження паспорту №2

Завантаження фото та збереження введеної інформації виконується за допомогою компоненту **UploadPassport**:

upload-passport

```
class UploadPassport
{
  public function upload(Worker $worker, Request $request)
  {
     $worker->addDocument(
       file: $request->getFile(),
       type: 'passport',
       data: $request->getFields(),
     );

    return $this->redirect("workers/{$worker->id}");
  }
}
```

Рис.3.12. Завантаження паспорту (код)

Після завантаження паспорту в компоненті документів можна знайти паспорт, який було завантажено.

При натисканні на документ буде виведено додаткову інформацію про документ та допоміжний функціонал для завантаження або видалення документу.

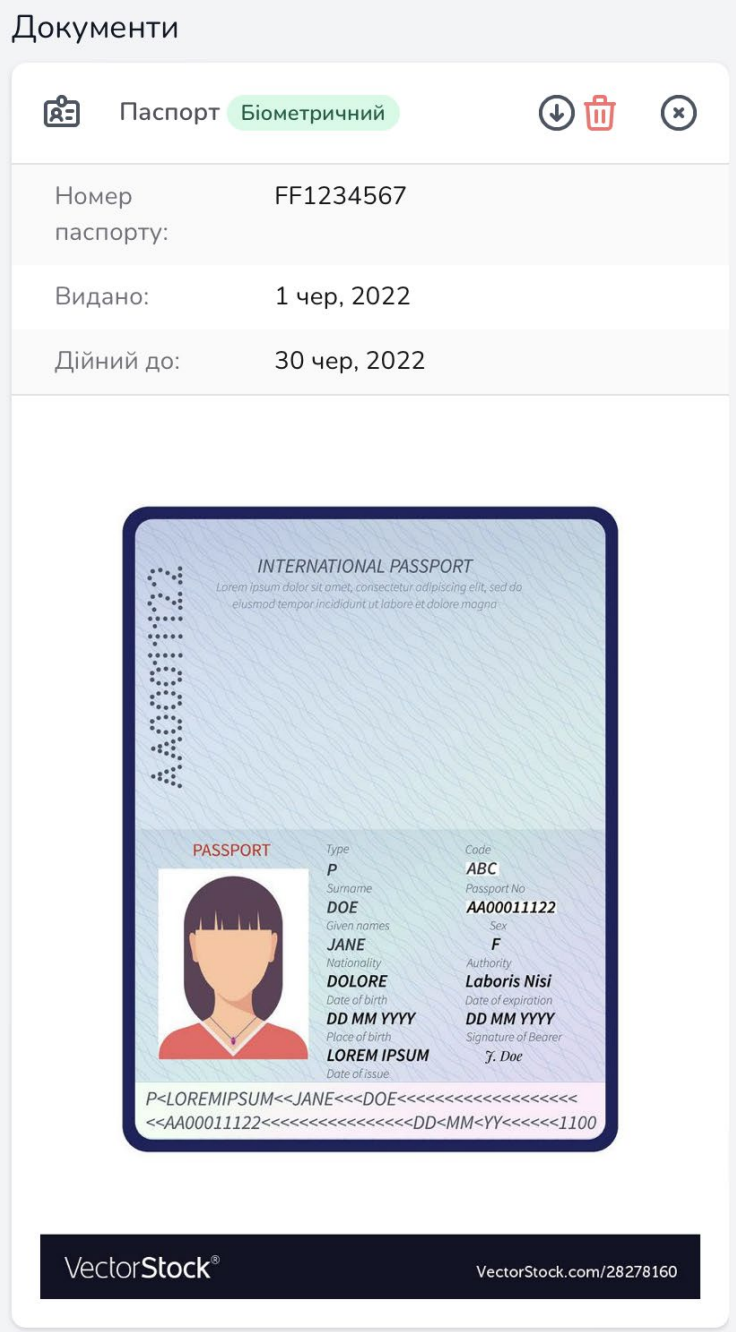

Рис.3.13. Перегляд паспорту

3.4. Внесення та використання інформації про працевлаштування

Створивши робітника, завантаживши всі потрібні документи, ми можемо розпочати працевлаштування робітника.

Працевлаштування розпочинається зі створення та підпису контракту між робітником та компанією. Створити працевлаштування можна на сторінці робітника, в компоненті «Працевлаштування» натиснувши кнопку «Додати умову».

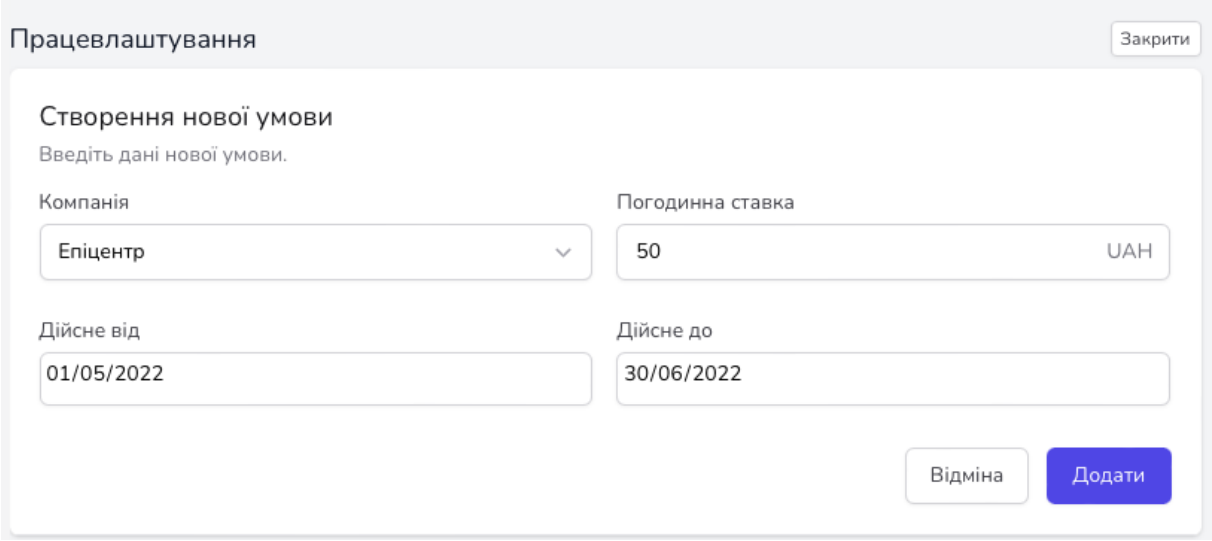

Рис.3.14. Створення працевлаштування

Для створення працевлаштування потрібно ввести потрібну інформацію про контракт та натиснути кнопку «Додати». У результаті буде створене працевлаштування, яке буде відображене на сторінці працівника.

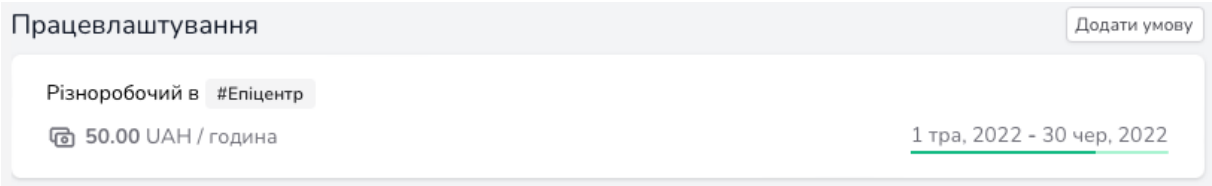

Рис.3.15. Перегляд працевлаштування №1

Умова буде автоматично визначена по даті, яка була вказана при створенні контракту.

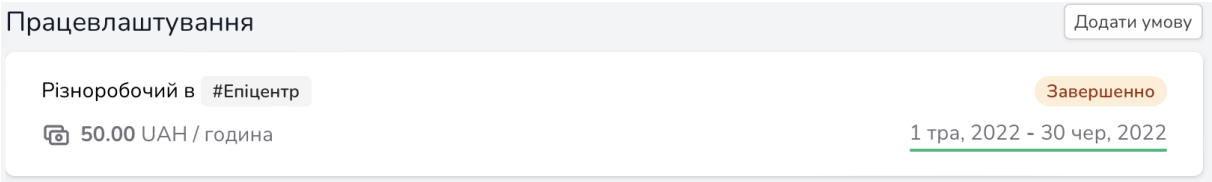

Рис.3.16. Перегляд працевлаштування №2

Перевірка чи завершена умова здійснюється за допомогою вхідних параметрів початку та кінця дії контракту.

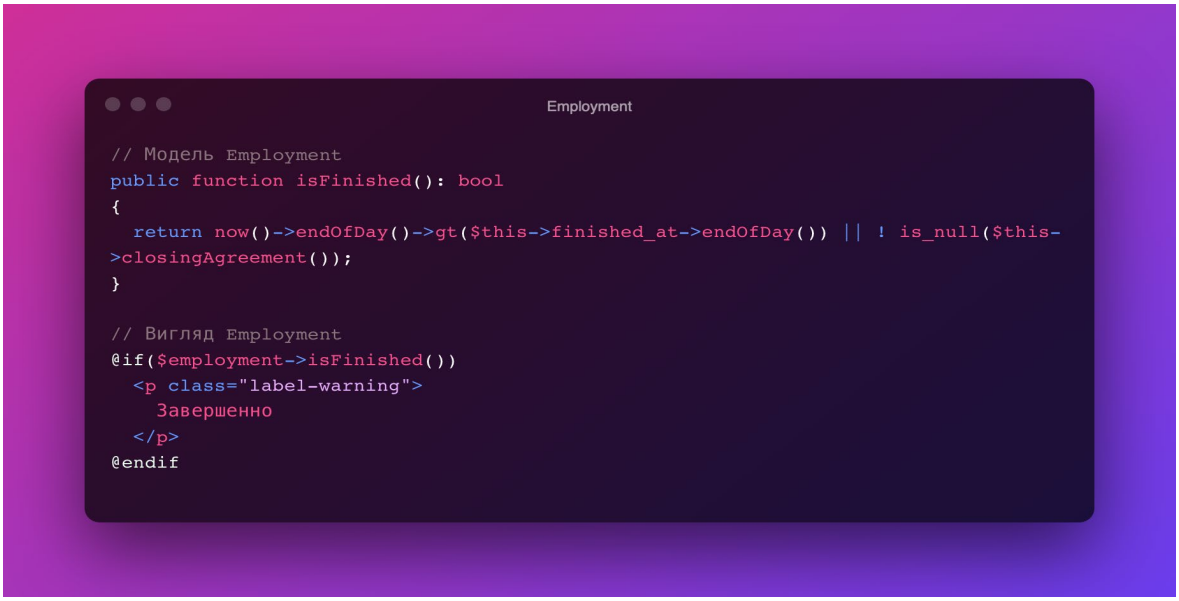

Рис.3.17. Деталі працевлаштування (код)

Також є можливість переглянути більше деталей вибраної умови, натиснувши на потрібну умову.

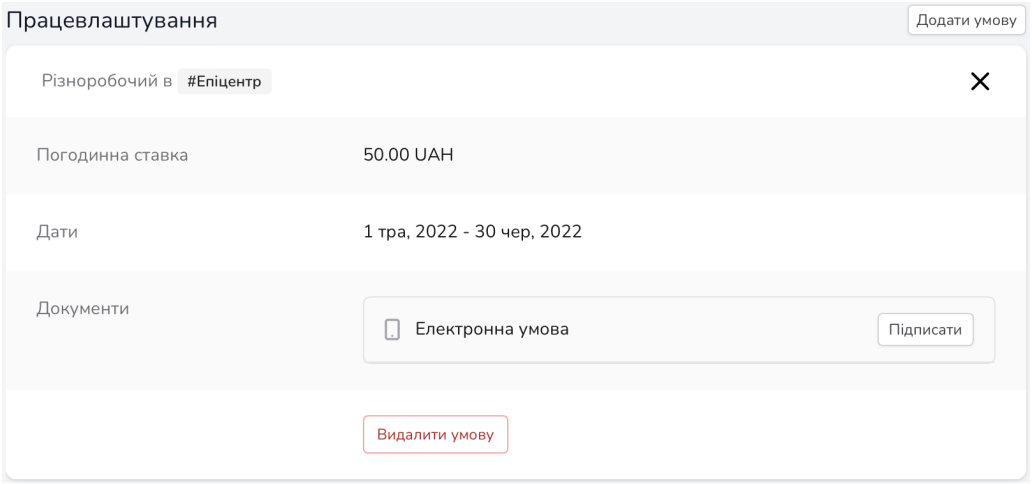

Рис.3.18. Деталі працевлаштування

За допомогою працевлаштувань компанії мають змогу вести облік їхніх робітників та зберігати/підписувати умови з компаніями.

ВИСНОВОК

Головною метою цієї роботи було застосування веб-фреймворку Laravel для побудови на його базі CRM-системи, яка забезпечить покращення роботи рекрутингових агентств, надаючи їм комфортні умови для роботи з компаніями та робітниками, з якими вони співпрацюють.

Було проаналізовано ринок рекрутингових агентств та складено список бізнес вимог, вирішення яких могла б вирішити CRM. Список основних бізнес вимог містить:

1) менеджмент працівників;
2) менеджмент компаній;
3) менеджмент умов контрактів (працевлаштувань).

Зручність використання Laravel фреймворку допомогла у створенні сучасної CRM-системи за невеликий проміжок часу відразу з автоматичним урахуванням вимог кібербезпеки та сучасності використаних технологій. Екосистема фреймворку приємно вражає надаючи все необхідне для успішної розробки проєкту та подальшому розгортання його на сервері.

Отже, у результаті виконання дипломної роботи було побудовано CRM-систему на базі Laravel, яка задовольняє основні вимоги ринку рекрутингових агентств.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Чернявська О. Ринок праці : навч. посіб. 2-ге вид. перероб та доп. Київ : «Центр учбової літератури», 2019. 522 с.
2. Рефакторинг.Гуру. URL: https://refactoring.guru/uk
3. Що таке CRM. URL: https://www.bitrix24.ua/articles/crm_what_is.php
4. Що таке CRM-система? URL: https://business.diia.gov.ua/handbook/prodazi/so-take-crm-sistema-ak-obrati-j-pracuvati-z-crm-so-vazlivo-zamiruvati
5. Beer B., Bell P. Introducing GitHub: A Non-Technical Guide. 2014. 172 p.
6. Chandrasekara C. Hands-on GitHub Actions: Implement CI/CD with GitHub Action Workflows for Your Applications 1st ed. 2021. 180 p.
7. Correa D. Practical Laravel: Develop clean MVC web applications. 2022. 211 p.
8. Correa D. Practical Laravel: Develop clean MVC web applications – Introduction to Laravel and Installation. 2022. 18 p.
9. Correa D. Practical Laravel: Develop clean MVC web applications – Introduction to MVC applications. 2022. 21 p.
10. Correa D. Practical Laravel: Develop clean MVC web applications – Configuration of MySQL Database. 2022. 30 p.
11. Correa D. Practical Laravel: Develop clean MVC web applications – Product Model. 2022. 36 p.
12. Correa D. Practical Laravel: Develop clean MVC web applications – Deploying to the Cloud – Heroku – Laravel Application. 2022. 180 p.
13. McGrath M. HTML, CSS & JavaScript in easy steps. 2020. 480 p.
14. Winand M. SQL Performance Explained Everything Developers Need to Know about SQL Performance. 2012. 193 p.
15. Winand M. SQL Performance Explained Everything Developers Need to Know about SQL Performance – Anatomy of an Index. 2012. 1 p.

16. Winand M. SQL Performance Explained Everything Developers Need to Know about SQL Performance – The Where Clause. 2012. 9 p.
17. Winand M. SQL Performance Explained Everything Developers Need to Know about SQL Performance – Performance and Scalability. 2012. 79 p.
18. Laracasts. URL: http://laracasts.com
19. Laravel Eloquent. URL: https://laravel.com/docs/9.x/eloquent
20. Laravel Database. URL: https://laravel.com/docs/9.x/database