

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

кафедра автоматика та управління в технічних системах
(повна назва кафедри)

Курсова робота

з дисципліни «Основи програмування»

на тему: веб-сайт пошуку роботи

Виконав : студент 1 курсу, групи ІТ-03
(шифр групи)

Чабан Антон Євгенович (прізвище, ім'я, по батькові) (підпис)

Науковий керівник _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Члени комісії _____ (посада,
науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

_____ (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Студент
Чабан А.Є. _____ (підпис)

Київ – 2021

ЗМІСТ

| | |
|---|----|
| ВСТУП | 3 |
| 1 ВИМОГИ ДО СИСТЕМИ | 4 |
| 1.1 Функціональні вимоги до системи..... | 4 |
| 1.2 Нефункціональні вимоги до системи..... | 4 |
| 2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ..... | 5 |
| 2.1 Діаграма прецедентів..... | 5 |
| 2.2 Опис сценаріїв використання системи..... | 6 |
| 3 АРХІТЕКТУРА СИСТЕМИ..... | 18 |
| 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ | 20 |
| 4.1 Загальна структура проекту | 20 |
| 4.2 Компоненти рівня доступу до даних..... | 20 |
| 4.3 Компоненти рівня бізнес-логіки..... | 26 |
| 4.4 Компоненти рівня інтерфейсу користувача | 29 |
| ВИСНОВКИ..... | 31 |
| ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ..... | 32 |
| ДОДАТКИ..... | 33 |
| ДОДАТОК А..... | 33 |
| ДОДАТОК Б..... | 43 |
| ДОДАТОК В..... | 46 |

ВСТУП

Пошук роботи в сучасному світі, зазвичай, відбувається в дистанційному форматі. А епоха карантину, особливо закріпила цей формат і показав його ефективність. Тому, на сьогоднішній день – сайт пошуку вакансій є надзвичайно актуальною темою.

Одним з популярних сервісів пошуку роботи в Україні є rabota.ua , проте існує і багато аналогів.

Насамперед, постає питання пошуку конкретної вакансії – отже, невід'ємною частиною буде реалізація пошуку по ключовим словам необхідної вакансії серед запропонованих.

По-друге, для роботодавця постає питання створення, редагування та видалення вакансій, що теж є частиною функціоналу.

Усі хто шукають вакансії мають одразу бачити ключові вимоги, такі як: досвід роботи та вимоги до освіти.

Метою роботи є створення системи, що дозволяє роботодавцям розміщувати нові вакансії, редагувати та видаляти їх, користувачам – шукати та дізнаватися деталі. Для цього система повинна мати певний перелік властивостей та вирішувати такі задачі:

- Переглядати дошку вакансій та конкретні вакансії
- Створювати вакансії
- Редагувати вакансії
- Видаляти вакансії
- Шукати вакансії

1 ВИМОГИ ДО СИСТЕМИ

1.1 Функціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

- незареєстрований користувач повинен мати можливість переглядати дошку вакансій та інформацію про них
- зареєстрований користувач повинен мати усі можливості, що є у незареєстрованого користувача, а також він повинен мати можливість створювати нові вакансії, та редагувати власні

1.2 Нефункціональні вимоги до системи

Система має відповідати наступним нефункціональним вимогам:

- система повинна мати архітектуру MVC;
- система повинна мати веб-інтерфейс;
- інтерфейс користувача має бути зручним та інтуїтивно-зрозумілим;
- система повинна бути крос-платформною.

2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

2.1 Діаграма прецедентів

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами є користувачі системи: незареєстрований (гість) та зареєстрований (роботодавець).

Зареєстрованому користувачу доступна уся функціональність, що і незареєстрованому, а також можливість створення, редагування та видалення вакансії.

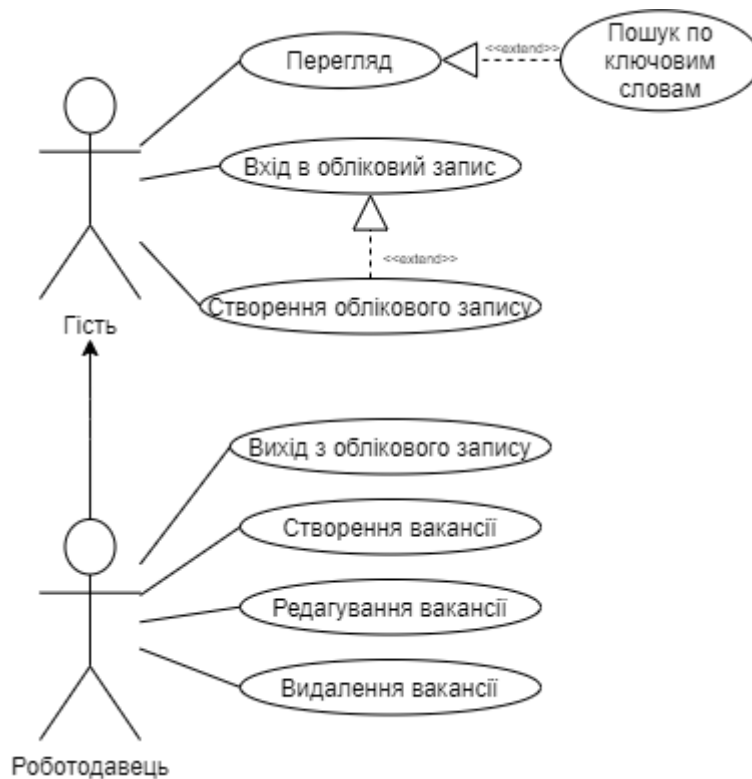


Рисунок 2.1 – Діаграма прецедентів

2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.7.

В таблиці 2.1 представлений сценарій використання «Перегляд дошки вакансій»

Таблиця 2.1 – Сценарій використання «Перегляд дошки вакансій»

| | |
|----------------------|---|
| Назва | Перегляд дошки вакансій |
| ID | 1 |
| Опис | Користувач переглядає вакансії на головній сторінці |
| Актори | Користувач(гість/роботодавець) |
| Вигоди компанії | Сервіс з пошуку вакансій без можливості перегляду дошки оголошень не буде користуватися попитом |
| Частота користування | Постійно |
| Тригери | Перехід на головну сторінку |
| Передумови | — |
| Постумови | Користувач бачить всі доступні вакансії |

| | |
|------------------------|--|
| Основний розвиток | Користувач переходить на головну сторінку і «скролить» сайт для перегляду вакансій |
| Альтернативні розвитку | - |
| Виняткові ситуації | - |

Таблиця 2.2 – Сценарій використання «Пошук по ключовим словам»

| | |
|------------------------|---|
| Назва | Пошук по ключовим словам |
| ID | 2 |
| Опис | Користувач, використовуючи поле пошуку шукає вакансії по ключовим словам |
| Актори | Користувач(гість/роботодавець) |
| Вигоди компанії | Якщо користувачі не можуть зменшити вибірку вакансій за потрібними ключовими словами, то пошук займе більше часу, отже клієнти буду шукати більш ефективні альтернативи |
| Частота користування | Часто |
| Тригери | Користувач вводить пошуковий запит |
| Передумови | Пошукове поле доступне у будь-якому вікні |
| Постумови | Користувач потрапляє на вікно з результатами пошуку |
| Основний розвиток | Користувач вводить запит у пошукову строку, натискає на кнопку пошуку чи Enter |
| Альтернативні розвитку | Поле пошуку не заповнено, тоді відобразяться всі доступні вакансії |
| Виняткові ситуації | — |

В таблиці 2.3 представлений сценарій використання «Перегляд обраної вакансії»

Таблиця 2.3 – Сценарій використання «Перегляд обраної вакансії»

| | |
|----------------------|--|
| Назва | Перегляд обраної вакансії |
| ID | 3 |
| Опис | Користувач обирає вакансію, яку хоче переглянути та потрапляє у вікно перегляду |
| Актори | Користувач(гість/роботодавець) |
| Вигоди компанії | Сервіс з пошуку вакансій без можливості перегляду конкретних оголошень не буде користуватися попитом |
| Частота користування | Постійно |
| Тригери | Користувач натискає на цікаву йому тему (“See more”) |
| Передумови | Користувач натиснув на вакансію на головній сторінці |
| Постумови | Користувач потрапляє на сторінку даної вакансії |
| Основний розвиток | Користувач знаходить цікаву йому вакансію на головній сторінці, або при пошуку та натискає на неї |

| | |
|------------------------|--|
| Альтернативні розвитки | Користувач переходить за посиланням даної вакансії |
| Виняткові ситуації | - |

В таблиці 2.4 представлений сценарій використання «Створення власної вакансії»

Таблиця 2.4 – Сценарій використання «Створення власної вакансії»

| | |
|----------------------|--|
| Назва | Створення власної вакансії |
| ID | 4 |
| Опис | Роботодавець створює нову вакансію |
| Актори | Роботодавець |
| Вигоди компанії | Легке створення власних вакансій утримує користувачів та спонукає до повторних дій |
| Частота користування | Часто |
| Тригери | Роботодавець натискає кнопку “Create New Vacancy” |
| Передумови | Користувач зайшов у обліковий запис |

| | |
|------------------------|---|
| Постумови | Роботодавець зберігає нове оголошення у базі оголошень |
| Основний розвиток | <p>Роботодавець натискає кнопку “Create New Vacancy” на головній сторінці. Відкривається вікно з доступними для редагування полями</p> <p>Роботодавець заповнює обов’язкові текстові та цифрові поля з параметрами: назва, опис, необхідний досвід.</p> <p>Роботодавець заповнює поля з прапорцями за бажанням: необхідна вища освіта.</p> <p>Хазяїн натискає кнопку “Save”</p> <p>Відбувається збереження нової вакансії та присвоєння йому ID у базі даних</p> <p>Нове оголошення додається до списку власних оголошень роботодавця</p> |
| Альтернативні розвитку | - |
| Виняткові ситуації | Роботодавець заповнює поля невірним типом даних, тоді збереження не можливе і користувачу буде запропоновано заповнити необхідні поля коректними даними |

В таблиці 2.5 представлений сценарій використання «Редагування власної вакансії»

Таблиця 2.5 – Сценарій використання «Редагування власної вакансії»

| | |
|-------|------------------------------|
| Назва | Редагування власної вакансії |
| ID | 5 |

| | |
|------------------------|--|
| Опис | Роботодавець редагує власне оголошення |
| Актори | Роботодавець |
| Вигоди компанії | Легке редагування власних вакансій утримує користувачів та спонукає до повторних дій |
| Частота користування | Часто |
| Тригери | Роботодавець натискає кнопку «Edit» на сторінці власної вакансії |
| Передумови | Роботодавець зайшов у власний обліковий запис та має власні вакансії |
| Постумови | Зміни зберігаються у базі даних |
| Основний розвиток | <p>Роботодавець натиснув на кнопку “Edit” на сторінці власної вакансії.</p> <p>Відбувається пошук вакансії по ID у базі даних</p> <p>Відкривається вікно з доступними для редагування полями, що заповнені даними об’єкту вакансії.</p> <p>Роботодавець редагує поля за бажанням.</p> <p>Натискає кнопку “Update”</p> <p>Відбувається збереження змін у БД</p> <p>Відкривається головна сторінка</p> |
| Альтернативні розвитку | — |

| | |
|--------------------|--|
| Виняткові ситуації | Хазяїн видаляє інформацію з обов'язкових полів або заповнює їх невірним типом даних, тоді оновлення не можливе і користувачу буде запропоновано заповнити необхідні поля коректними даними |
|--------------------|--|

В таблиці 2.6 представлений сценарій використання «Видалення власної вакансії»

Таблиця 2.6 – Сценарій використання «Видалення власної вакансії»

| | |
|----------------------|---|
| Назва | Видалення власної вакансії |
| ID | 6 |
| Опис | Роботодавець видаляє власну вакансію |
| Актори | Роботодавець |
| Вигоди компанії | Легке видалення власних оголошень утримує користувачів та спонукає до повторних дій |
| Частота користування | Часто |
| Тригери | Роботодавець натиснув кнопку Edit і обрав кнопку Delete |
| Передумови | Роботодавець увійшов до облікового запису, має власні вакансії які можна видалити |
| Постумови | Роботодавець видаляє власну вакансію з БД |

| | |
|------------------------|--|
| Основний розвиток | Роботодавець знаходиться у вікні редагування власної вакансії Роботодавець натискає кнопку “Delete” Відбувається видалення вакансії з бази даних Відбувається оновлення списку вакансій |
| Альтернативні розвитку | - |
| Виняткові ситуації | - |

В таблиці 2.7 представлений сценарій використання «Вхід в обліковий запис»

Таблиця 2.7 – Сценарій використання «Вхід в обліковий запис»

| | |
|----------------------|---|
| Назва | Вхід в обліковий запис |
| ID | 7 |
| Опис | Користувач входить у існуючий обліковий запис |
| Актори | Користувач |
| Вигоди компанії | Можливість входу у власний обліковий запис привертає нових користувачів, та надає їм змогу стати керувати власними оголошеннями |
| Частота користування | Регулярно |

| | |
|------------------------|---|
| Тригери | Користувач заповнює поля логіну, пароллю та натискає кнопку “Submit” |
| Передумови | Обліковий запис існує |
| Постумови | Зміна ролі з гостя на роботодавця |
| Основний розвиток | <p>Користувач заходить на сайт та натискає кнопку Login</p> <p>Користувач потрапляє на сторінку логіну</p> <p>Користувач заповнює обов’язкові поля: логін, пароль</p> <p>Користувач натискає кнопку “Submit”</p> <p>Відбувається пошук облікового запису у базі даних за логіном</p> <p>Користувач потрапляє на головну сторінку у ролі роботодавця вже у обліковому записі</p> |
| Альтернативні розвитку | - |
| Виняткові ситуації | <p>Користувач не заповнює усі поля або заповнює їх некоректним типом даних, тоді вхід неможливий і користувачу буде запропоновано заповнити усі необхідні поля коректними даними</p> <p>Користувач вводить неіснуючий логін, тоді відкривається вікно помилки</p> <p>Користувач вводить невірний пароль, тоді відкривається вікно помилки</p> |

В таблиці 2.8 представлений сценарій використання «Створення облікового запису»

Таблиця 2.8 – Сценарій використання «Вхід в обліковий запис»

| | |
|-------|------------------------|
| Назва | Вхід в обліковий запис |
|-------|------------------------|

| | |
|------------------------|---|
| ID | 8 |
| Опис | Користувач створює новий обліковий запис |
| Актори | Користувач |
| Вигоди компанії | Можливість створення нового облікового запису привертає нових користувачів, та надає їм змогу стати Хазяїнами оголошень |
| Частота користування | Регулярно |
| Тригери | Користувач заповнює поля логіну, паролю та натискає кнопку "Register" |
| Передумови | Немає |
| Постумови | Створює новий обліковий запис у БД Зміна ролі з гостя на роботодавця |
| Основний розвиток | Користувач заходить на сайт та натискає кнопку Sign-Up Користувач потрапляє на сторінку реєстрації Користувач заповнює обов'язкові поля: логін, пароль Користувач натискає кнопку "Register" Створюється новий обліковий запис у БД та йому присвоюється ID Користувач потрапляє на головну сторінку у ролі роботодавця вже у новому обліковому записі |
| Альтернативні розвитку | - |

| | |
|--------------------|--|
| Виняткові ситуації | <p>Користувач не заповнює усі поля або заповнює їх некоректним типом даних, тоді реєстрація неможлива і користувачу буде запропоновано заповнити усі необхідні поля коректними даними</p> <p>Користувач вводить вже існуючий логін, тоді відкривається вікно помилки</p> |
|--------------------|--|

3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1

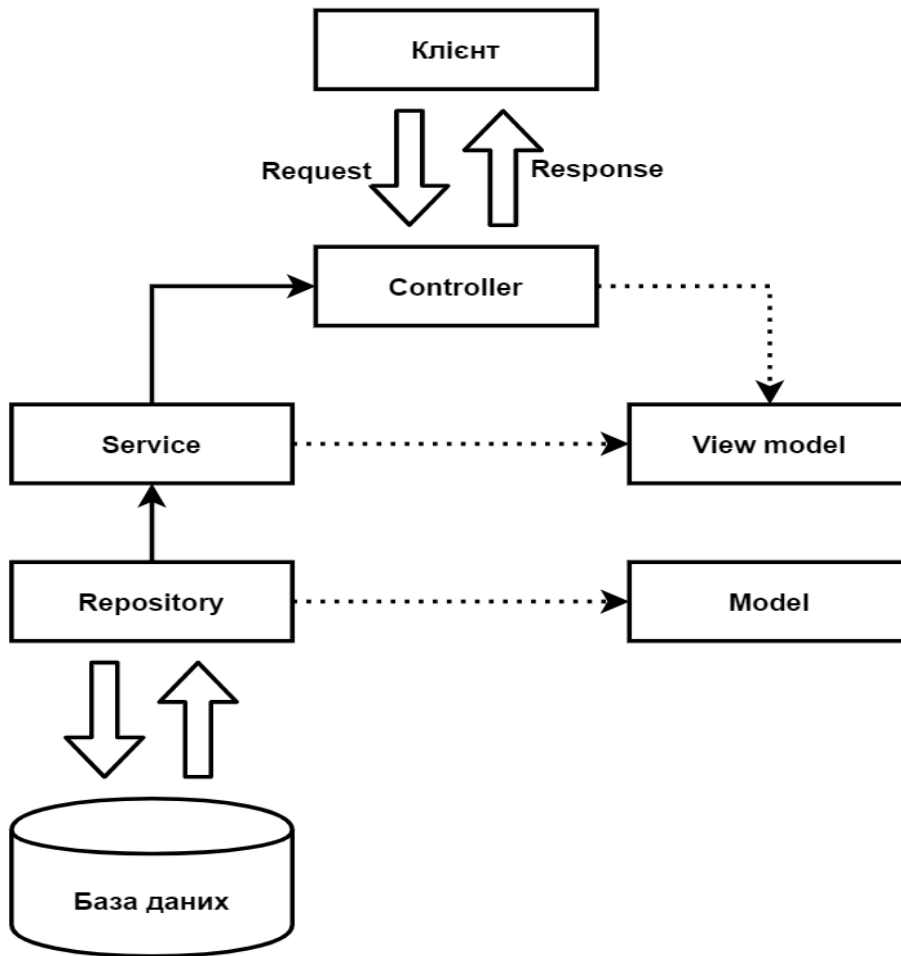


Рисунок 3.1 – Загальна архітектура системи

Система складається з наступних елементів:

- графічний інтерфейс;
- серверна частина;
- база даних (in memory implementation)

Графічний інтерфейс необхідний для взаємодії з користувачем. HTTP запит надходить до серверної частини, де оброблюється і повертається відповідь. На серверній частині виконується основна логіка системи. Дані, отриманні з

графічного інтерфейсу валідуються, конвертуються. Також, серверна частина формує запит до бази даних та оброблює відповідь і передає її до графічного інтерфейсу. База даних зберігає дані, які були сформовані на серверній частині та повертає їх у разі запиту.

До серверної частини належать наступні елементи:

- контролер;
- модель та вигляд;
- сервіс;
- репозиторій.

На контролер надходять дані з графічного інтерфейсу. З контролеру, дані формуються в сервісі для запиту в репозиторій. З репозиторію дані надсилаються до бази даних і зберігаються. Також в контролері формується вид, тобто об'єкт і його ім'я для відображення на графічному інтерфейсі.

4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

4.1 Загальна структура проекту

Загальна структура проекту представлена на рис.4.1

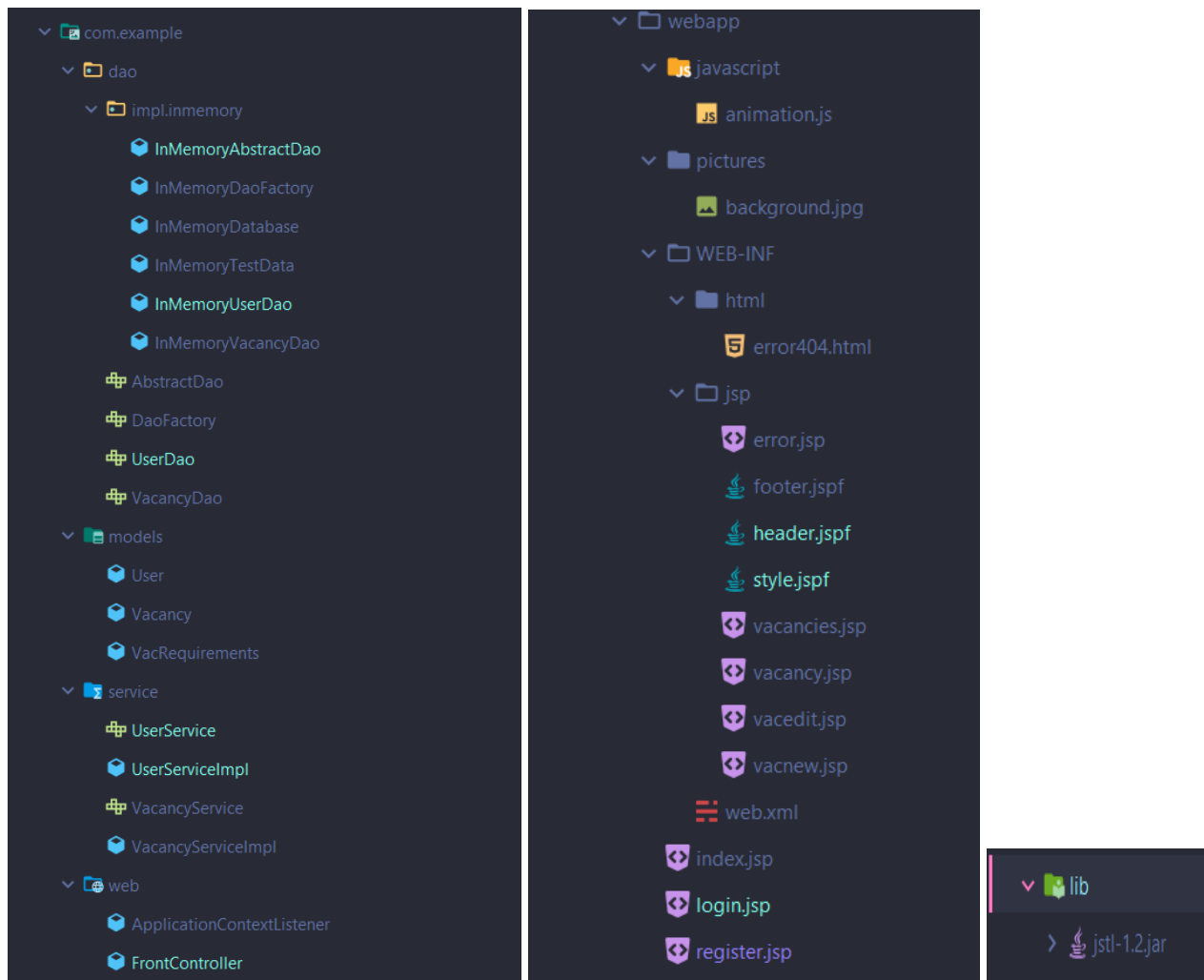


Рисунок 4.1 – Загальна структура проекту

Проект складається з веб-ресурсів, бібліотек, та вихідного коду, який в свою чергу можна поділити на компоненти рівня доступу до даних, компоненти бізнес логіки та веб-компоненти.

4.2 Компоненти рівня доступу до даних

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.2

AbstractDAO інтерфейс

Описує такі дії:

додавання нового об'єкту певного типу з вибірковою генерацією ID;
видалення об'єкту певного типу;
оновлення об'єкту певного типу;
отримання колекції з усіх об'єктів певного типу;
отримання об'єкту певного типу за його ID;

InMemoryAbstractDao реалізує AbstractDAO

Має такі сутності:

Мар об'єктів певного типу з їх ID в якості ключів;
Функції для отримання та встановлення ID об'єкту певного типу;
Базу даних.

Конструктор приймає та встановлює усі сутності protected рівня.

І реалізує наступні дії інтерфейсу:

Додавання об'єкту певного типу

Видалення об'єкту певного типу з Мар об'єктів за значенням ключа

Оновлення об'єкту певного типу з Мар об'єктів за значенням ключа

Отримання колекції об'єктів певного типу – повертається колекція зіставлена зі значень Мар об'єктів певного типу

Отримання об'єкту певного типу за ID – повертається знайдений за ключем ID об'єкт з Мар об'єктів

DaoFactory інтерфейс

Описує такі дії:

Отримання об'єкту типу UserDao

Отримання об'єкту типу VacancyDao

InMemoryDaoFactory реалізує інтерфейс DaoFactory

Має такі сутності:

Об'єкти інтерфейсів UserDao та VacancyDao

Об'єкт бази даних

Конструктор приймає базу даних та встановлює сутності класу ініційовані від неї.

Реалізує дії інтерфейсу:

Отримання об'єкту типу UserDao – повертає власну сутність типу UserDao;

Отримання об'єкту типу VacancyDao – повертає класну сутність типу VacancyDao.

UserDao інтерфейс розширяє AbstractDao від об'єкту типу User

Описує дії:

Отримання об'єкту типу User за його логіном

Додавання об'єкту типу Vacansy до власного списку об'єкту типу User

Видалення об'єкту типу Vacansy з власного списку об'єкту типу User

Додавання нового об'єкту типу User.

InMemoryDatabaseDao

Конструктор приймає базу даних та наслідує дії InMemoryAbstractDao (визначення сутностей protected рівня: функції отримання та встановлення ID з класу User, Map об'єктів типу User з бази даних та саму базу даних).

Реалізує дії інтерфейсу:

Отримання об'єкту типу User за його логіном – потік зі значень Map об'єктів типу User бази даних фільтруються за умовою еквівалентності логіна та повертає перший знайдений об'єкт типу User, інакше null

Додавання об'єкту типу Vacansy до власного списку об'єкту типу User – до власної HashMap об'єкту типу User додається об'єкт типу Vacansy за його ID як ключем

Видалення об'єкту типу Vacansy з власного списку об'єкту типу User - з власної HashMap об'єкту типу User видаляється об'єкт типу Vacansy за його ID як ключем

Додавання нового об'єкту типу User – викликається метод додавання InMemoryAbstractDao від об'єкту типу User з підтвердженням генерації для нього ID, а у об'єкту типу User встановлюється ID за отриманим з об'єкту типу User шуканим за його логіном.

VacancyDao інтерфейс розширює AbstractDAO від об'єкту типу Vacansy

Описує дії:

Отримання колекції об'єктів типу Vacansy за ключовими словами

Отримання HashMap об'єктів типу Vacansy за ID власника

Додавання нового об'єкту типу Vacansy за його власником

Видалення об'єкту типу Vacansy за його власником.

InMemoryVacancyDao реалізує інтерфейс VacancyDao та розширює

InMemoryAbstractDao від об'єкту типу Vacansy

Конструктор приймає базу даних та наслідує дії InMemoryAbstractDao (визначення сутностей protected рівня: функції отримання та встановлення ID з класу Vacansy, Map об'єктів типу Vacansy з бази даних та саму базу даних).

Реалізує дії інтерфейсу:

Отримання колекції об'єктів типу Vacancy за ключовими словами – повертає колекцію об'єктів типу Vacancy з фільтрації потоку

Отримання HashMap об'єктів типу Vacancy за ID роботодавця – повертає HashMap з об'єктів типу Vacancy та їх ID як ключів з фільтрації потоку значень Map об'єктів типу Vacancy з бази даних за умовою еквівалентності ID власника

Додавання нового об'єкту типу Vacancy за його за його власником - викликається метод додавання InMemoryAbstractDao від об'єкту типу Vacancy з підтвердженням генерації для нього ID, а у об'єкту типу Vacancy встановлюється ID з функції отримання ID за об'єктом типу Vacancy, окрім цього власнику присвоюється новий HashMap об'єктів типу Vacancy отриманих за пошуком ID об'єкту типу User
Видалення об'єкту типу Vacancy за його роботодавцем - викликається метод видалення InMemoryAbstractDao від об'єкту типу Vacancy, а власнику присвоюється новий HashMap об'єктів типу Vacancy отриманих за пошуком ID об'єкту типу User.

User

Має такі поля:

Integer userId – ID користувача

String name – Ім'я користувача

String login – логін користувача

String password – пароль користувача

HashMap<Integer, Vacancy> myVacs – мапа з об'єктами типу вакансії та їх ID

Конструктор (ID, Ім'я, логін, пароль)

Конструктор (ID, Ім'я, логін, пароль, мапа моїх вакансій)

Конструктор(Ім'я, логін, пароль) – для створення нового користувача

Реалізує такі дії:

Отримання та встановлення полів

Vacancy

Має такі поля:

Integer vacId – ID вакансії

Integer userId – ID роботодавця

String vacName – назва вакансії

String description – опис вакансії

VacRequirements requirements – об'єкт типу VacRequirements

Конструктор усіх полів

Конструктор усіх полів окрім ID вакансій

Реалізує такі дії:

Отримання та встановлення полів

VacRequirements

Integer experience – необхідний досвід

boolean highEducation – чи потрібна вища освіта

Конструктор полів

Реалізує такі дії:

Отримання та встановлення полів

4.3 Компоненти рівня бізнес-логіки

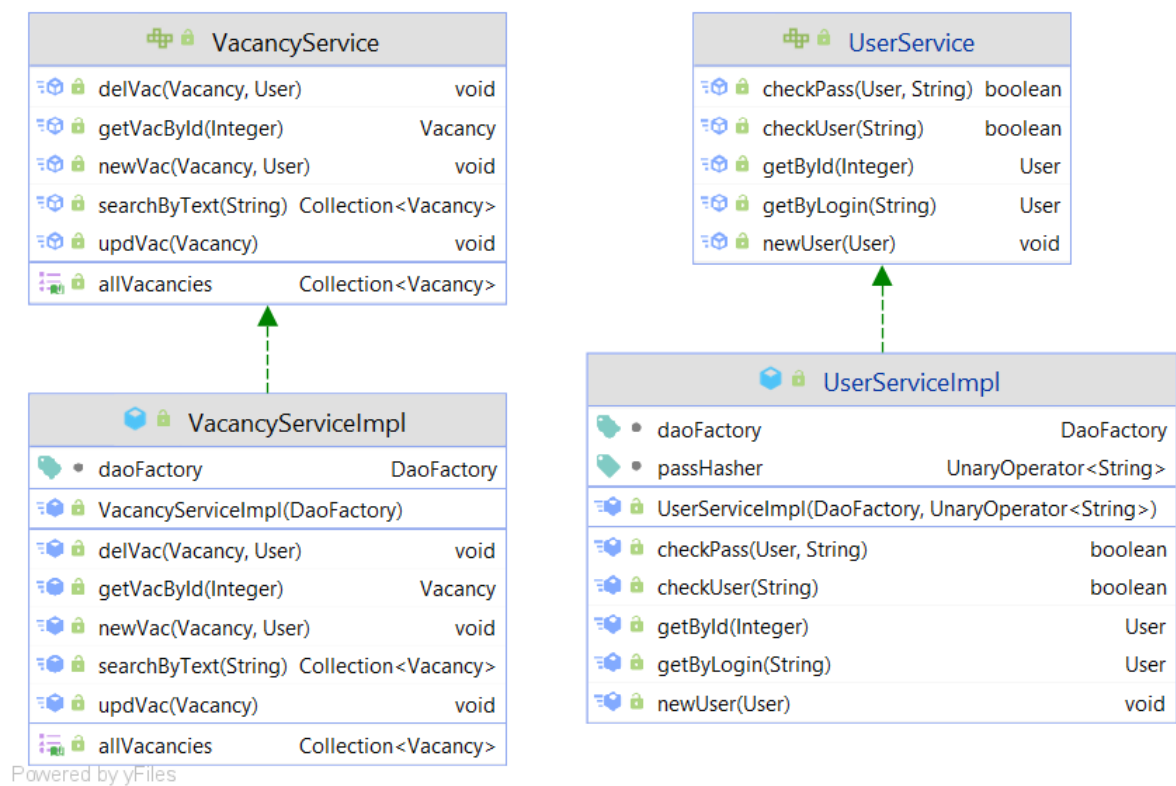


Рисунок 4.3 – Основні сутності рівня бізнес-логіки

VacancyService інтерфейс

Описує такі дії:

Отримання колекції об'єктів типу Vacancy

Отримання колекції об'єктів типу Vacancy за ключовими словами

Отримання об'єкту типу Vacancy за його ID

Видалення об'єкту типу Vacancy за власником

Оновлення об'єкту типу Vacancy

Додавання нового об'єкту типу Vacancy за власником

VacancyServiceImpl реалізує VacancyService

Має такі сутності:

Об'єкт типу DaoFactory

Конструктор приймає об'єкт типу DaoFactory та призначає його власній сутності цього типу

Реалізує такі дії:

Отримання колекції об'єктів типу Vacancy – повертає колекцію усіх об'єктів типу Vacancy зі звернення до об'єкту типу VacancyDao з DaoFactory

Отримання колекції об'єктів типу Vacancy за параметрами пошуку - повертає колекцію відповідних ключовим словам, або усіх, якщо параметри не задано, об'єктів типу Vacancy зі звернення до об'єкту типу VacancyDao з DaoFactory

Отримання об'єкту типу Vacancy за його ID - повертає об'єкт типу Vacancy зі звернення до об'єкту типу VacancyDao з DaoFactory за його ID

Видалення об'єкту типу Vacancy за власником – видаляє об'єкт типу Vacancy за зверненням до об'єкту типу VacancyDao з DaoFactory

Оновлення об'єкту типу Vacancy – оновлює об'єкт типу Vacancy за зверненням до об'єкту типу VacancyDao з DaoFactory

Додавання нового об'єкту типу Vacancy за власником – додає об'єкт типу Vacancy за зверненням до об'єкту типу VacancyDao з DaoFactory

UserService інтерфейс

Описує такі дії:

Отримання об'єкту типу User за логіном

Отримання об'єкту типу User за ID

Підтвердження паролю об'єкту типу User за паролем

Підтвердження існування об'єкту типу User за логіном

Створення нового об'єкту типу User

UserServiceImpl реалізує інтерфейс UserService

Має такі сутності:

Об'єкту типу DaoFactory

Унарний оператор String для паролю

Конструктор що приймає та призначає усі сутності класу

Реалізує такі дії:

Отримання об'єкту типу User за логіном – повертає об'єкт типу User зі звернення до об'єкту типу UserDao з DaoFactory за його логіном



















Отримання об'єкту типу User за ID - повертає об'єкт типу User зі звернення до об'єкту типу UserDao з DaoFactory за його ID

Підтвердження паролю об'єкту типу User за паролем – повертає підтвердження еквівалентності паролю обробленого унарним оператором паролю об'єкту типу User

Підтвердження існування об'єкту типу User за логіном – повертає підтвердження існування об'єкту типу User зі звернення до об'єкту типу UserDao з DaoFactory за його логіном

Створення нового об'єкту типу User – додає новий об'єкт типу User у базу даних за зверненням до об'єкту типу UserDao з DaoFactory

4.4 Компоненти рівня інтерфейсу користувача

| FrontController | | |
|---|---|----------------|
|  | vacService | VacancyService |
|  | userService | UserService |
|  | FrontController() | |
|  | doGet(HttpServletRequest, HttpServletResponse) | void |
|  | doPost(HttpServletRequest, HttpServletResponse) | void |
|  | error(HttpServletRequest, HttpServletResponse, String) | void |
|  | init(ServletConfig) | void |
|  | login(HttpServletRequest, HttpServletResponse) | void |
|  | logout(HttpServletRequest, HttpServletResponse) | void |
|  | main(HttpServletRequest, HttpServletResponse) | void |
|  | processRequest(HttpServletRequest, HttpServletResponse) | void |
|  | register(HttpServletRequest, HttpServletResponse) | void |
|  | vacancy(HttpServletRequest, HttpServletResponse) | void |
|  | vacdel(HttpServletRequest, HttpServletResponse) | void |
|  | vacedit(HttpServletRequest, HttpServletResponse) | void |
|  | vacnew(HttpServletRequest, HttpServletResponse) | void |
|  | vacsave(HttpServletRequest, HttpServletResponse) | void |
|  | vacupdate(HttpServletRequest, HttpServletResponse) | void |

| ApplicationContextListener | | |
|---|------|--|
| ApplicationContextListener() | | |
| contextDestroyed(ServletContextEvent) | void | |
| contextInitialized(ServletContextEvent) | void | |

Powered by yFiles

Рисунок 4.4 – Основні сутності рівня інтерфейсу користувача

Інтерфейс користувача складається з jsp/html сторінок та контролеру, який використовується для зв'язку та взаємодії з ними.

Ініціалізація контексту відбувалася за допомогою `ApplicationContextListener`, в якому створювалися об'єкти `InMemoryDatabase` та ініціалізація тестової БД.

При передачі запиту на контролер він задіює метод `processRequest(req, resp)`, який, в свою чергу, використовує один з наступних методів:

`Login` – для авторизації, вже зареєстрованого в системі користувача.

`Logout` – для виходу з облікового запису користувача в системі .

`Main` – головна сторінка яка зчитує усі параметри пошуку у `String`, та записує у `HashMap` для подальшого пошуку по ключовим словам.

`Vacancy` – метод для переходу на сторінку конкретної вакансії.

`VacEdit` – метод для редагування вже створеної вакансії.

`VacNew` – метод для створення нової вакансії.

`VacDel` – метод для видалення з бази вже існуючої вакансії.

`VacSave` – метод для збереження до бази нової вакансії.

`VacUpdate` – метод для оновлення інформації про відредаговану вакансію у базі

`Register` – метод для реєстрації нового користувача у системі сайту

`Error` – метод для відображення повідомлень про помилки

ВИСНОВКИ

Під час виконання роботи, було проведено аналіз ринку сервісів для пошуку вакансій, були враховані всі їх переваги та недоліки, для створення оптимальної системи. Було вирішено розробити систему з універсальним пошуком по ключовим словам, та можливістю перегляду, як загальної дошки вакансій, так і конкретної.

Першим кроком було формування схеми роботи проекту, функціональні вимоги до системи. Наступним кроком було обрання технології на яких буде написана система. В якості мови програмування була обрана Java, середою розробки послугувала IntelliJ IDEA, їх можливості та потужності в сукупності дали можливість системі мати необхідний функціонал. Для роботи з запитами були створені власні інтерфейси та класи. В якості графічного інтерфейсу користувача виступали HTML, JSP з використанням бібліотеки JSTL, вибір пав саме на них через зручність у використанні.

Наступним кроком були описані сценарії використання. Система підтримує функціонал перегляду, створення, оновлення та видалення вакансій. Також була впроваджена система автентифікації користувачів, яка надає їм можливість створення, входу та виходу з облікового запису.

Далі, необхідно було розробити архітектуру системи. Використовуючи шаблон проектування MVC – систему було розділено на логічні рівні.

Далі була спроектована модель для встановлення взаємозв'язку між БД та елементами коду Java. Було створено моделі, та необхідні поля та взаємозв'язки.

Отже, сумуючи наведене вище, можна зрозуміти що перевагами системи є її простота та інтуїтивна зрозумілість для користувача. Завдяки відкритій архітектурі системи – розширення та підтримка системи не є проблемою. Графічний інтерфейс є зручним та простим в налаштуванні. Система є конкурентноспроможною та має потенціал для оновлень та розширення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. rabota.ua [Електронний ресурс] – Режим доступу до ресурсу: <https://rabota.ua>
2. LinkedIn [Електронний ресурс] – Режим доступу до ресурсу: - <https://www.linkedin.com>
3. CodePly [Електронний ресурс] – Режим доступу до ресурсу: - <https://www.codeply.com/>
4. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: - <https://getbootstrap.com>
5. HTML Documentation [Електронний ресурс] – Режим доступу до ресурсу: - <https://developer.mozilla.org/en-US/docs/Web/HTML>
6. CSS Guides [Електронний ресурс] – Режим доступу до ресурсу: - <https://www.w3schools.com/css/>
7. Draw.io [Електронний ресурс] – Режим доступу до ресурсу: - <https://app.diagrams.net>

ДОДАТКИ

ДОДАТОК А

Рівень доступу до даних

User

```
package com.example.models;

import java.util.HashMap;
import java.util.Objects;

public class User {
    private Integer userId;
    private String name;
    private String login;
    private String password;
    private HashMap<Integer, Vacancy> myVacs;

    public User(Integer userId, String name, String login, String password) {
        this.userId = userId;
        this.name = name;
        this.login = login;
        this.password = password;
    }

    public User(Integer userId, String name, String login, String password,
HashMap<Integer, Vacancy> myVacs) {
        this.userId = userId;
        this.name = name;
        this.login = login;
        this.password = password;
        this.myVacs = myVacs;
    }

    public User(String name, String login, String password) {
        this.name = name;
        this.login = login;
        this.password = password;
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLogin() {
```

```

        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public HashMap<Integer, Vacancy> getMyVacs() {
        return myVacs;
    }

    public void setMyVacs(HashMap<Integer, Vacancy> myVacs) {
        this.myVacs = myVacs;
    }

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 37 * hash + Objects.hashCode(this.userId);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final User other = (User) obj;
        if (!Objects.equals(this.userId, other.userId)) {
            return false;
        }
        return true;
    }
}

```

Vacancy

```

package com.example.models;

public class Vacancy {
    private Integer vacId;
    private Integer userId;
    private String vacName;
    private String description;
    private VacRequirements requirements;
}

```

```

    public Vacancy(Integer vacId, Integer userId, String vacName, String description,
VacRequirements requirements) {
        this.vacId = vacId;
        this.userId = userId;
        this.vacName = vacName;
        this.description = description;
        this.requirements = requirements;
    }

    public String getVacName() {
        return vacName;
    }

    public void setVacName(String vacName) {
        this.vacName = vacName;
    }

    public Vacancy(Integer userId, String vacName, String description, VacRequirements
requirements) {
        this.userId = userId;
        this.vacName = vacName;
        this.description = description;
        this.requirements = requirements;
    }

    public Integer getVacId() {
        return vacId;
    }

    public void setVacId(Integer vacId) {
        this.vacId = vacId;
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public VacRequirements getRequirements() {
        return requirements;
    }

    public void setRequirements(VacRequirements requirements) {
        this.requirements = requirements;
    }
}

```

VacancyRequirements

```
package com.example.models;

public class VacRequirements {
    private Integer experience;
    private boolean highEducation;

    public Integer getExperience() {
        return experience;
    }

    public void setExperience(Integer experience) {
        this.experience = experience;
    }

    public boolean isHighEducation() {
        return highEducation;
    }

    public void setHighEducation(boolean highEducation) {
        this.highEducation = highEducation;
    }

    public VacRequirements(Integer experience, boolean highEducation) {
        this.experience = experience;
        this.highEducation = highEducation;
    }
}
```

AbstractDao

```
package com.example.dao;

import java.util.Collection;

public interface AbstractDao<T> {
    T get(Integer id);

    Collection<T> findAll();

    void insert(T entity, boolean generatedId);

    void delete(T entity);

    void update(T entity);
}
```

DaoFactory

```
package com.example.dao;

public interface DaoFactory {
    UserDao getUserDao();
    VacancyDao getVacancyDao();
}
```

UserDao

```

package com.example.dao;

import com.example.models.User;
import com.example.models.Vacancy;

public interface UserDao extends AbstractDao<User>{

    User getByLogin(String login);

    void addVac(User user, Vacancy vacancy);
    void delVac(User user, Vacancy vacancy);
    void newUser(User user);
}

```

VacancyDao

```

package com.example.dao;

import com.example.models.User;
import com.example.models.Vacancy;

import java.util.Collection;
import java.util.HashMap;

public interface VacancyDao extends AbstractDao<Vacancy>{
    Collection<Vacancy> findByText(String string);
    HashMap<Integer,Vacancy> findByUserID(Integer userId);
    void addVac(Vacancy vacancy, User user);
    void delVac(Vacancy vacancy, User user);
}

```

InMemoryAbstractDao

```

package com.example.dao.impl.inmemory;

import com.example.dao.AbstractDao;

import java.util.Collection;
import java.util.Map;
import java.util.function.BiConsumer;
import java.util.function.Function;

public class InMemoryAbstractDao<T> implements AbstractDao<T> {
    protected Map<Integer, T> entities;
    protected Function<T, Integer> idGetter;
    protected BiConsumer<T, Integer> idSetter;
    protected InMemoryDatabase database;

    InMemoryAbstractDao(Map<Integer, T> entities, Function<T, Integer> idGetter,
        BiConsumer<T, Integer> idSetter,
        InMemoryDatabase database) {
        this.entities = entities;
        this.idGetter = idGetter;
        this.idSetter = idSetter;
        this.database = database;
    }

    @Override
    public T get(Integer id) {

```

```

        return entities.get(id);
    }

    @Override
    public Collection<T> findAll() {
        return entities.values();
    }

    @Override
    public void insert(T entity, boolean generatedId) {
        if (generatedId) {
            int maxId = entities.keySet().stream()
                .mapToInt(Integer::intValue)
                .max()
                .orElse(0);
            idSetter.accept(entity, maxId + 1);
        }
        entities.put(idGetter.apply(entity), entity);
    }

    @Override
    public void delete(T entity) {
        entities.remove(idGetter.apply(entity));
    }

    @Override
    public void update(T entity) {
        entities.put(idGetter.apply(entity), entity);
    }
}

```

InMemoryDaoFactory

```

package com.example.dao.impl.inmemory;

import com.example.dao.DaoFactory;
import com.example.dao.UserDao;
import com.example.dao.VacancyDao;

public class InMemoryDaoFactory implements DaoFactory {
    InMemoryDatabase database;

    VacancyDao vacancyDao;
    UserDao userDao;

    InMemoryDaoFactory(InMemoryDatabase database) {
        this.database = database;

        vacancyDao = new InMemoryVacancyDao(database);
        userDao = new InMemoryUserDao(database);
    }

    @Override
    public UserDao getUserDao() {
        return userDao;
    }

    @Override

```

```

    public VacancyDao getVacancyDao() {
        return vacancyDao;
    }
}

```

InMemoryDatabase

```

package com.example.dao.impl.inmemory;

import com.example.dao.DaoFactory;
import com.example.models.*;
import java.util.*;

public class InMemoryDatabase {
    Map<Integer, User> users;
    Map<Integer, Vacancy> vacancies;

    public InMemoryDatabase() {
        vacancies = new TreeMap<>();
        users = new TreeMap<>();
    }

    public DaoFactory getDaoFactory() {
        return new InMemoryDaoFactory(this);
    }
}

```

InMemoryTestData

```

package com.example.dao.impl.inmemory;

import com.example.models.*;
import java.util.Arrays;
import java.util.List;

public class InMemoryTestData {

    public static void generateTo(InMemoryDatabase database) {
        database.users.clear();
        database.vacancies.clear();

        User admin = new User(1, "admin", "admin@gmail.com", "admin");
        User alice = new User(2, "Alice", "alice@example.com", "passwordhash");
        User bob = new User(3, "Bob", "bob@example.com", "passwordhash");
        User charlie = new User(4, "Charlie", "charlie@example.com", "passwordhash");
        User diana = new User(5, "Diana", "diana@example.com", "passwordhash");
        List<User> users = Arrays.asList(admin, alice, bob, charlie, diana);
        users.forEach(user -> database.users.put(user.getUserId(), user));

        String desc1 = "Join us in shaping no code computing. At JourneyXP we develop a no code computing platform, JXP Cloud, that simplifies how software is build, operated, and scaled.";
        Vacancy vac1 = new Vacancy(1, admin.getUserId(), "Junior Java Developer", desc1, new VacRequirements(2, false));

        String desc2 = "One of the world's largest oil field service companies has a need to build a modern solutions that improve efficiency, increase recovery and maximize production.This is a position for those interested in working with the latest Microsoft modern technology with the user in focus.";
    }
}

```

```

        Vacancy vac2 = new Vacancy(2, alice.getUserId(), "Intern C# Developer", desc2,
new VacRequirements(0, true));

        String desc3 = "Luxoft team is developing a family of condition-based and
predictive health management products that uses various online and offline data
sources to provide preventive intervention recommendations based on grid asset
condition, probability of failure, criticality and risk assessments. Data lake
collects telemetrics data, makes clearance, manages archives. Rich UI to visualize
results of analysis and diagnostics, prognostics, including 3D representation, builds
prognosis on assets health and performance. Offline data is gathered via hybrid mobile
application and through external connectors.\n" +
        "We are working with BigData scale and using latest technologies like
Java 11, SpringBoot 2, Spring Cloud, Kafka, Ignite, Cassandra, PostgreSQL, Jupyter,
Docker and Kubernetes, AWS, Angular 10.\n" +
        "Our team is distributed between Ukraine and France (client).\n" +
        "Opportunities: study cutting-edge technologies and participate in
multi-year project for one of the biggest companies in the World, contribute to the
platform that will change work experience and results of energy grid engineers across
the globe.";

        Vacancy vac3 = new Vacancy(3, bob.getUserId(), "Java Intern", desc3, new
VacRequirements(0, false));

        String desc4 = "Our client is an American multinational corporation, one of
the world's largest providers of products and services that make the oil and gas
production possible.\n" +
        "Our Products are two scientifically grounded applications (Desktop
and Web) aimed to Design, Optimize and Monitor specific processes in field, simulate
or calculate various operations, view and evaluate real-time data acquisition, execute
various file imports, reporting and integrations with multiple other software
products.";

        Vacancy vac4 = new Vacancy(4, charlie.getUserId(), "Manual QA Intern", desc4,
new VacRequirements(0, false));

        String desc5 = "At Very Good Security ("VGS") we are on a mission to protect
the world's sensitive data - and we'd love to have you along for this journey.\n" +
        "\n" +
        "VGS was founded by highly successful repeat entrepreneurs, and is
backed by world-class investors like Goldman Sachs, Andreessen Horowitz, and Visa. We
are building an amazing global team spread across four cities. As a young and growing
company, we are laser-focused on delighting our customers and hiring talented and
entrepreneurial-minded individuals.";

        Vacancy vac5 = new Vacancy(5, diana.getUserId(), "Senior Software Engineer",
desc5, new VacRequirements(5, true));

        List<Vacancy> vacancies = Arrays.asList(vac1, vac2, vac3, vac4, vac5);
        vacancies.forEach(vacancy -> database.vacancies.put(vacancy.getId(),
vacancy));
        for (User user : users) {

user.setMyVacs(database.getDaoFactory().getVacancyDao().findByUserID(user.getUserId())
);

        }

}

```



```
}
```

InMemoryUserDao

```
package com.example.dao.impl.inmemory;

import com.example.dao.UserDao;
import com.example.models.User;
import com.example.models.Vacancy;

public class InMemoryUserDao extends InMemoryAbstractDao<User> implements UserDao {
    InMemoryUserDao(InMemoryDatabase database) {
        super(database.users, User::getUserId, User::setUserId, database);
    }

    @Override
    public User getByLogin(String login) {
        return database.users.values()
            .stream()
            .filter(user -> user.getLogin().equals(login))
            .findFirst()
            .orElse(null);
    }

    @Override
    public void addVac(User user, Vacancy vacancy) {
        user.getMyVacs().put(vacancy.getVacId(), vacancy);
    }

    @Override
    public void delVac(User user, Vacancy vacancy) {
        user.getMyVacs().remove(vacancy.getVacId());
    }

    @Override
    public void newUser(User user) {
        this.insert(user, true);
        user.setUserId(this.idGetter.apply(user));
    }
}
```

InMemoryVacancyDao

```
package com.example.dao.impl.inmemory;

import com.example.dao.VacancyDao;
import com.example.models.User;
import com.example.models.Vacancy;

import java.util.Collection;
import java.util.HashMap;
import java.util.function.Function;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class InMemoryVacancyDao extends InMemoryAbstractDao<Vacancy> implements
VacancyDao {
    InMemoryVacancyDao(InMemoryDatabase database) {
        super(database.vacancies, Vacancy::getVacId, Vacancy::setVacId, database);
    }
}
```

```

    }

    @Override
    public Collection<Vacancy> findByText(String string) {
        String[] words = string.toLowerCase().split(" ");
        return database.vacancies.values().stream()
            .filter(movie -> containsAllWords(movie, words))
            .collect(Collectors.toList());
    }

    @Override
    public HashMap<Integer, Vacancy> findByUserID(Integer userId) {
        return database.vacancies.values()
            .stream()
            .filter(vacancy -> vacancy.getUserId().equals(userId))
            .collect(Collectors.toMap(Vacancy::getVacId, Function.identity(),
                (prev, next) -> next, HashMap::new));
    }

    @Override
    public void addVac(Vacancy vacancy, User user) {
        this.insert(vacancy, true);
        vacancy.setVacId(this.idGetter.apply(vacancy));
        HashMap newVac = findByUserID(user.getUserId());
        user.setMyVacs(newVac);
    }

    @Override
    public void delVac(Vacancy vacancy, User user) {
        this.delete(vacancy);
        HashMap newVac = findByUserID(user.getUserId());
        user.setMyVacs(newVac);
    }

    private static boolean containsAllWords(Vacancy vacancy, String[] words) {
        String string = vacancy.getVacName() + " " + vacancy.getDescription();
        string = string.toLowerCase();
        return Stream.of(words).allMatch(string::contains);
    }
}

```

ДОДАТОК Б

Рівень бізнес-логіки

UserService

```
package com.example.service;

import com.example.models.User;

public interface UserService {
    User getByLogin(String login);
    User getById(Integer userId);
    boolean checkPass(User user, String password);
    boolean checkUser(String user);
    void newUser(User user);
}
```

UserServiceImpl

```
package com.example.service;

import com.example.dao.DaoFactory;
import com.example.models.User;

import java.util.function.UnaryOperator;

public class UserServiceImpl implements UserService {
    DaoFactory daoFactory;
    UnaryOperator<String> passHasher;

    public UserServiceImpl(DaoFactory daoFactory, UnaryOperator<String> passHasher) {
        this.daoFactory = daoFactory;
        this.passHasher = passHasher;
    }

    @Override
    public User getByLogin(String login) {
        return daoFactory.getUserDao().getByLogin(login);
    }

    @Override
    public User getById(Integer userId) {
        return daoFactory.getUserDao().get(userId);
    }

    @Override
    public boolean checkPass(User user, String password) {
        return user.getPassword().equals(passHasher.apply(password));
    }

    @Override
    public boolean checkUser(String user) {
        if (daoFactory.getUserDao().getByLogin(user) == null) {
            return false;
        }
        return true;
    }

    @Override
```

```

    public void newUser(User user) {
        daoFactory.getUserDao().newUser(user);
    }
}

```

VacancyService

```

package com.example.service;

import com.example.models.User;
import com.example.models.Vacancy;

import java.util.Collection;

public interface VacancyService {
    Collection<Vacancy> getAllVacancies();
    Collection<Vacancy> searchByText(String string);
    Vacancy getVacById(Integer vacId);
    void delVac(Vacancy vacancy, User user);
    void updVac(Vacancy vacancy);
    void newVac(Vacancy vacancy, User user);
}

```

VacancyServiceImpl

```

package com.example.service;

import com.example.dao.DaoFactory;
import com.example.models.User;
import com.example.models.Vacancy;

import java.util.Collection;

public class VacancyServiceImpl implements VacancyService {
    DaoFactory daoFactory;

    public VacancyServiceImpl(DaoFactory daoFactory) {
        this.daoFactory = daoFactory;
    }

    @Override
    public Collection<Vacancy> getAllVacancies() {
        return daoFactory.getVacancyDao().findAll();
    }

    @Override
    public Collection<Vacancy> searchByText(String string) {
        if (string == null || string.equals("")) {
            return getAllVacancies();
        }
        return daoFactory.getVacancyDao().findByText(string);
    }

    @Override
    public Vacancy getVacById(Integer vacId) {
        return daoFactory.getVacancyDao().get(vacId);
    }
}

```

```
@Override
public void delVac(Vacancy vacancy, User user) {
    daoFactory.getVacancyDao().delVac(vacancy, user);
}

@Override
public void updVac(Vacancy vacancy) {
    daoFactory.getVacancyDao().update(vacancy);
}

@Override
public void newVac(Vacancy vacancy, User user) {
    daoFactory.getVacancyDao().addVac(vacancy, user);
}
}
```

ДОДАТОК В

Рівень інтерфейсу користувача

ApplicationContextListener

```
package com.example.web;

import com.example.dao.DaoFactory;
import com.example.dao.impl.inmemory.InMemoryDatabase;
import com.example.dao.impl.inmemory.InMemoryTestData;
import com.example.service.UserService;
import com.example.service.UserServiceImpl;
import com.example.service.VacancyService;
import com.example.service.VacancyServiceImpl;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;
import java.util.function.UnaryOperator;

@WebListener
public class ApplicationContextListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        InMemoryDatabase database = new InMemoryDatabase();
        InMemoryTestData.generateTo(database);
        DaoFactory daoFactory = database.getDaoFactory();

        VacancyService vacService = new VacancyServiceImpl(daoFactory);
        sce.getServletContext().setAttribute("vacService", vacService);

        UserService userService = new UserServiceImpl(daoFactory,
        UnaryOperator.identity());
        sce.getServletContext().setAttribute("userService", userService);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
    }
}
```

FrontController

```
package com.example.web;

import com.example.models.*;
import com.example.service.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.IOException;
import java.util.Collection;

@WebServlet(name = "FrontController", urlPatterns = {"/do/*"})
public class FrontController extends HttpServlet {
    VacancyService vacService;
    UserService userService;
```

```

@Override
public void init(ServletConfig config) throws ServletException {
    vacService = (VacancyService)
config.getServletContext().getAttribute("vacService");
    userService = (UserService)
config.getServletContext().getAttribute("userService");
}

protected void processRequest(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    String pathInfo = request.getPathInfo();
    if (pathInfo == null) {
        pathInfo = "/";
    }
    try {
        switch (pathInfo) {
            case "/login":
                login(request, response);
                break;
            case "/logout":
                logout(request, response);
                break;
            case "/main":
                main(request, response);
                break;
            case "/vacancy":
                vacancy(request, response);
                break;
            case "/vacedit":
                vacedit(request, response);
                break;
            case "/vacnew":
                vacnew(request, response);
                break;
            case "/vacdel":
                vacdel(request, response);
                break;
            case "/vacsave":
                vacsave(request, response);
                break;
            case "/vacupdate":
                vacupdate(request, response);
                break;
            case "/register":
                register(request, response);
                break;
            case "/":
            case "/search":
            default:
                main(request, response);
                break;
        }
    } catch (RuntimeException ex) {
        error(request, response, "Oops, " + ex.getMessage());
    }
}

```

```

    protected void register(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    request.getSession().invalidate();
    String login = request.getParameter("login");
    if (userService.checkUser(login)) {
        error(request, response, "Sorry, user with " + login + " is already
exists");
        return;
    }
    String password = request.getParameter("password");
    User user = new User(login, login, password);

    userService.newUser(user);
    request.getSession().setAttribute("user", user);
    response.sendRedirect(".");
}

    protected void vacsave(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    User user = (User) request.getSession().getAttribute("user");

    Boolean highEducation = false;
    if (request.getParameter("highEducation") != null) {
        highEducation = true;
    }

    Vacancy vacancy = new Vacancy(user.getUserId(),
request.getParameter("vacName"), request.getParameter("description"),
        new VacRequirements(Integer.parseInt(request.getParameter("exp")),
highEducation));

    vacService.newVac(vacancy, user);
    response.sendRedirect("main");
}

    protected void vacupdate(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    int vacId = Integer.parseInt(request.getParameter("vacId"));
    Vacancy vacancy = vacService.getVacById(vacId);

    Boolean highEducation = false;
    if (request.getParameter("highEducation") != null) {
        highEducation = true;
    }

    vacancy.getRequirements().setHighEducation(highEducation);

    if (request.getParameter("exp") != null) {
vacancy.getRequirements().setExperience(Integer.parseInt(request.getParameter("exp")))
;
    }

    if (request.getParameter("vacName") != null) {
        vacancy.setVacName(request.getParameter("vacName"));
    }
}

```



```

        if (request.getParameter("description") != null) {
            vacancy.setDescription(request.getParameter("description"));
        }

        vacService.updVac(vacancy);
        response.sendRedirect("main");
    }

    protected void vacnew(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        request.getSession().getAttribute("user");
        request.getRequestDispatcher("/WEB-INF/jsp/vacnew.jsp").forward(request,
        response);
    }

    protected void vacdel(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        int vacId = Integer.parseInt(request.getParameter("vacId"));
        Vacancy vacancy = vacService.getVacById(vacId);
        User user = (User) request.getSession().getAttribute("user");

        if (vacId == user.getMyVacs().get(vacId).getUserId()) {
            vacService.delVac(vacancy, user);
        } else error(request, response, "Not your vacancy");
        response.sendRedirect("main");
    }

    protected void vacedit(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        int vacId = Integer.parseInt(request.getParameter("vacId"));
        Vacancy vacancy = vacService.getVacById(vacId);
        request.setAttribute("vacancy", vacancy);
        request.getRequestDispatcher("/WEB-INF/jsp/vacedit.jsp").forward(request,
        response);
    }

    protected void vacancy(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        int vacId = Integer.parseInt(request.getParameter("vacId"));
        Vacancy vacancy = vacService.getVacById(vacId);
        request.setAttribute("vacancy", vacancy);
        request.getRequestDispatcher("/WEB-INF/jsp/vacancy.jsp").forward(request,
        response);
    }

    protected void main(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        String searchText = request.getParameter("search");
        Collection<Vacancy> vacancies;
        if (searchText != null) {
            vacancies = vacService.searchByText(searchText);
        } else
            vacancies = vacService.getAllVacancies();
        request.setAttribute("vacancies", vacancies);
        request.setAttribute("text", searchText);
    }

```

```

        request.getRequestDispatcher("/WEB-INF/jsp/vacancies.jsp").forward(request,
response);
    }

    protected void login(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        request.getSession().invalidate();

        String login = request.getParameter("login");
        User user = userService.getByLogin(login);
        if (!userService.checkUser(login)) {
            error(request, response, "Sorry, user with login " + login + " not
exists");
            return;
        }
        String password = request.getParameter("password");

        if (!userService.checkPass(user, password)) {
            error(request, response, "Sorry, wrong password");
            return;
        }

        request.getSession().setAttribute("user", user);
        response.sendRedirect(".");
    }

    protected void logout(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        request.setAttribute("vacancies", vacService.getAllVacancies());
        request.getSession().invalidate();
        response.sendRedirect(".");
    }

    protected void error(HttpServletRequest request, HttpServletResponse response,
String message) throws ServletException, IOException {
        request.setAttribute("message", message);
        request.getRequestDispatcher("/WEB-INF/jsp/error.jsp").forward(request,
response);
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        processRequest(request, response);
    }
}

```

Vacancies.jsp

```

<%--
Created by IntelliJ IDEA.
User: anton

```

Date: 27.04.2021

Time: 17:21

To change **this** template use File | Settings | File Templates.

```
--%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Vacancies</title>
    <%@ include file="style.jspf" %>
</head>
<body>
<%@ include file="header.jspf" %>
<c:forEach var="vacancy" items="${vacancies}">
    <div class="job-wrapper">
        <h1><a href="vacancy?vacId=${vacancy.vacId}"><c:out
value="${vacancy.vacName}" /></a></h1>
        <hr>
        <ul class="requirements">
            <c:if test="${vacancy.requirements.experience <= 0}">
                <li>Job Experience: <span style="color: #22ff22"><c:out value="No exp
required" /></span></li>
            </c:if>
            <c:if test="${vacancy.requirements.experience > 0}">
                <li>Job Experience: <span style="color: red"><c:out
value=" ${vacancy.requirements.experience}
year(s)" /></span></li>
            </c:if>
            <c:if test="${vacancy.requirements.highEducation}">
                <li>Higher Education: <span style="color: red"><c:out
value="Required" /></span></li>
            </c:if>
            <c:if test="${!vacancy.requirements.highEducation}">
                <li>Higher Education: <span style="color: #22ff22"><c:out value="Not
Required" /></span></li>
            </c:if>
            <li class="description">
                <c:out value="${vacancy.description}" />
            </li>
        </ul>
        <div class="more">
            <a href="vacancy?vacId=${vacancy.vacId}">See More</a>
        </div>
    </div>
</c:forEach>
<%@ include file="footer.jspf" %>
</body>
</html>
```



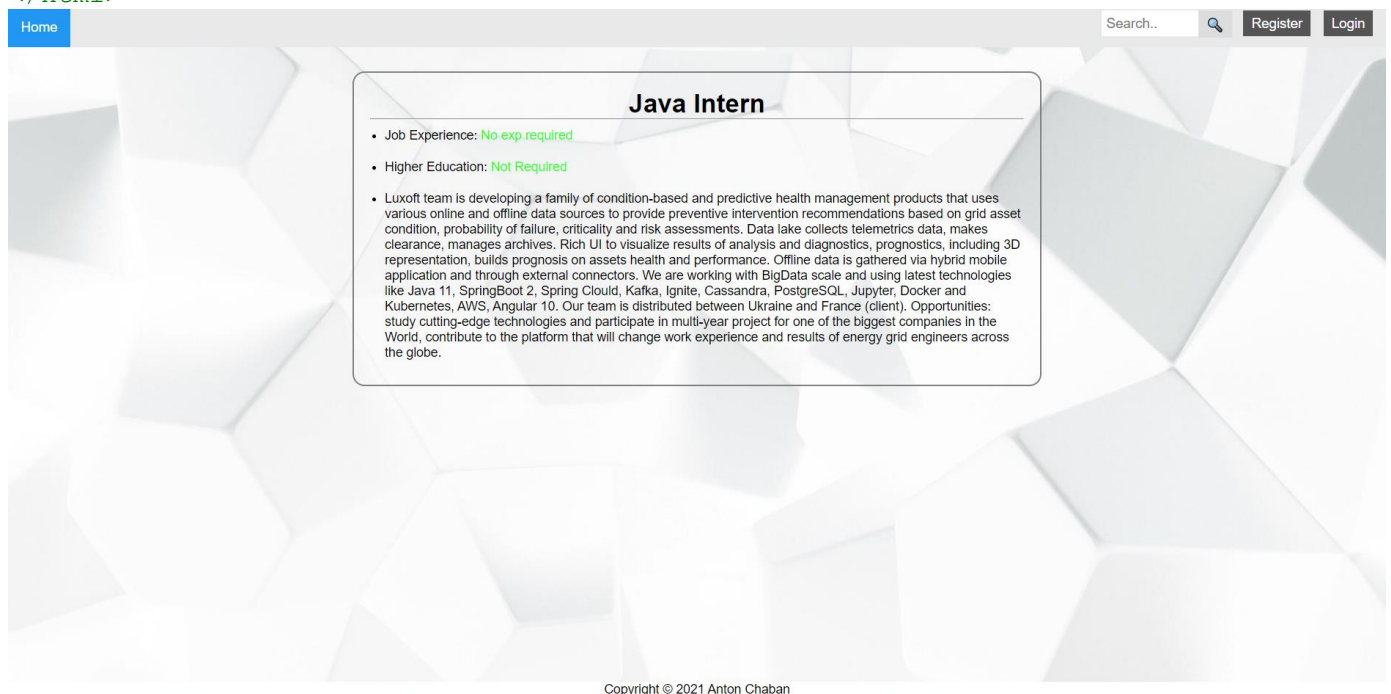
Vacancy.jsp

```
<%--
Created by IntelliJ IDEA.
User: anton
Date: 28.04.2021
Time: 12:29
To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title><c:out value="\${vacancy.vacName}"/></title>
    <%@include file="style.jspf" %>
    <style>
        .footer {
            position: fixed;
            left: 0;
            bottom: 0;
        }
    </style>
</head>
<body>
<%@include file="header.jspf" %>
<div class="vacinfo-center">
    <div class="vacinfo-wrapper">
        <h1 align="center"><c:out value="\${vacancy.vacName}"/></h1>
        <hr>
        <ul class="requirements">
            <c:if test="\${vacancy.requirements.experience <= 0}">
                <li>Job Experience: <span style="color: #22ff22"><c:out value="No exp
required"/></span></li>
            </c:if>
            <c:if test="\${vacancy.requirements.experience > 0}">
                <li>Job Experience: <span style="color: red"><c:out
```

```

        value="Exp required: ${vacancy.requirements.experience}
years"/></span></li>
    </c:if>
    <c:if test="${vacancy.requirements.highEducation}">
        <li>Higher Education: <span style="color: red"><c:out
value="Required"/></span></li>
    </c:if>
    <c:if test="${!vacancy.requirements.highEducation}">
        <li>Higher Education: <span style="color: #22ff22"><c:out value="Not
Required"/></span></li>
    </c:if>
    <li class="description">
        <c:out value="${vacancy.description}"/>
    </li>
</ul>
<c:if test="${vacancy.userId == user.userId}">
    <form action="vacedit" method="POST">
        <input type="hidden" name="vacId" value="${vacancy.vacId}">
        <button class="more" type="submit">EDIT</button>
    </form>
</c:if>
</div>
</div>
<%@include file="footer.jspf" %>
</body>
</html>

```



Vacedit.jsp

```

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%--
    Created by IntelliJ IDEA.
    User: anton
    Date: 28.04.2021
    Time: 14:56
    To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>

```

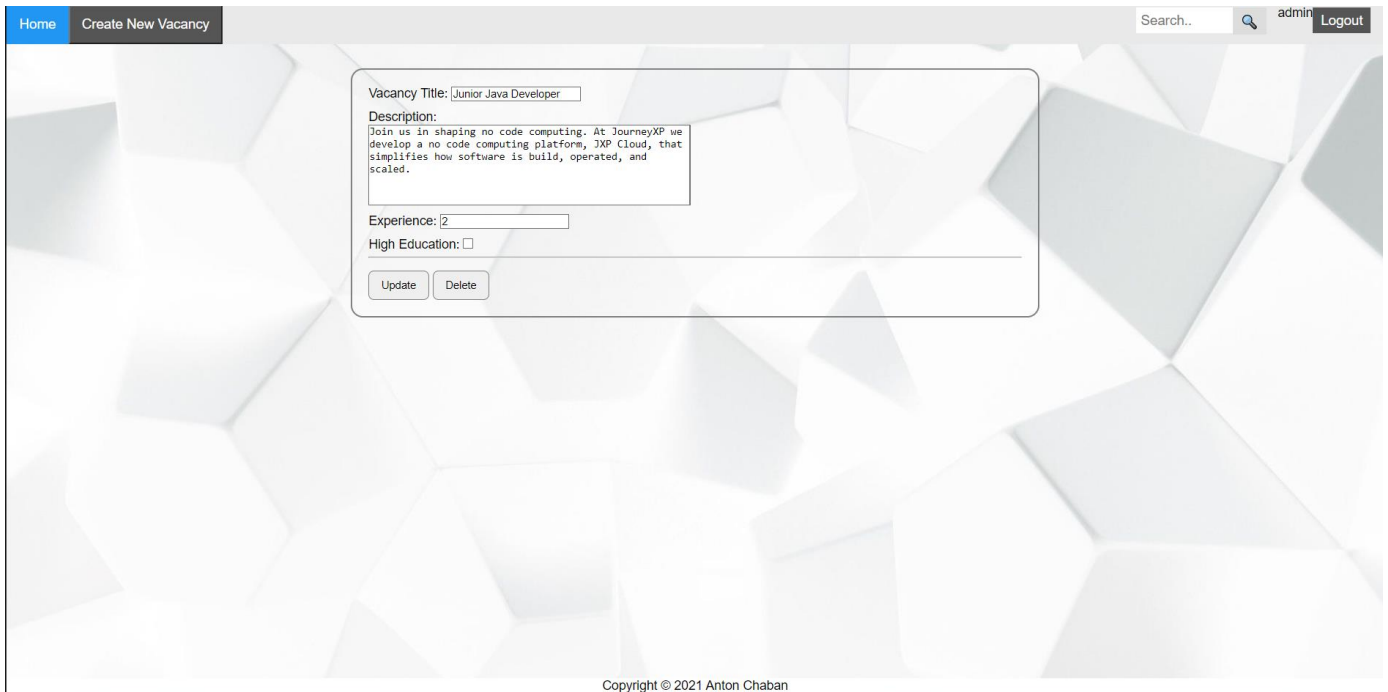
```

<html>
<head>
  <title><c:out value="Edit ${vacancy.vacName}"/></title>
  <%@include file="style.jspf" %>
  <style>
    textarea {
      width: 400px;
      height: 100px;
      resize: none;
    }

    .footer {
      position: fixed;
      left: 0;
      bottom: 0;
    }
  </style>

</head>
<body>
  <%@include file="header.jspf" %>
  <div class="vacinfo-center">
    <div class="vacinfo-wrapper">
      <form class="edit-form" action="vacupdate" method="POST">
        <input type="hidden" name="vacId" value="${vacancy.vacId}">
        <div class="v-title-edit">
          Vacancy Title: <input type="text" name="vacName"
value="${vacancy.vacName}" required/>
        </div>
        <div class="v-description-edit"> Description:
          <textarea type="text" name="description" required><c:out
value="${vacancy.description}"/></textarea>
        </div>
        <div class="v-exp-edit">
          Experience: <input type="number" name="exp"
value="${vacancy.requirements.experience}" required/>
        </div>
        <div class="v-education-edit">
          High Education: <input type="checkbox" name="highEducation"
          <c:if test="${vacancy.requirements.highEducation}">
            checked
          </c:if> />
        </div>
        <hr>
        <input class="more" id="update" type="submit" value="Update"/>
        <input class="more" id="delete" type="submit" value="Delete"
formaction="vacdel"/>
      </form>
    </form>
  </div>
  <%@include file="footer.jspf" %>
</body>
</html>

```



Vacnew.jsp

```
<!--
Created by IntelliJ IDEA.
User: anton
Date: 28.04.2021
Time: 19:06
To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Creating Vacancy</title>
    <%@include file="style.jspf" %>
    <style>
        textarea {
            width: 400px;
            height: 100px;
            resize: none;
        }

        .footer {
            position: fixed;
            left: 0;
            bottom: 0;
        }
    </style>
</head>
<body>
<%@include file="header.jspf" %>
<div class="vacinfo-center">
    <div class="vacinfo-wrapper">
        <form class="edit-form" action="vacsave" method="POST">
            <input type="hidden" name="vacId" value="${vacancy.vacId}">
            <div class="v-title-edit">
                Vacancy Title: <input type="text" name="vacName" required/>
            </div>
        </form>
    </div>
</div>
</body>
</html>
```

```

    </div>
    <div class="v-description-edit"> Description:
        <textarea type="text" name="description" required></textarea>
    </div>
    <div class="v-exp-edit">
        Experience: <input type="number" name="exp" required/>
    </div>
    <div class="v-education-edit">
        High Education: <input type="checkbox" name="highEducation"/>
    </div>
    <hr>
    <input class="more" id="save" type="submit" value="Save"/>

</form>

</div>
</div>
<%@include file="footer.jspf" %>
</body>
</html>

```

The screenshot displays a web interface for creating a new vacancy. The form is titled 'Create New Vacancy' and is located in the center of the page. It includes the following elements:

- Vacancy Title:** A text input field.
- Description:** A large text area for detailed input.
- Experience:** A number input field.
- High Education:** A checkbox.
- Save:** A button to submit the form.

The page layout includes a top navigation bar with 'Home' and 'Create New Vacancy' links, a search bar, and user authentication links for 'admin' and 'Logout'. The footer contains the copyright notice 'Copyright © 2021 Anton Chaban'.

Login.jsp

```

<%--
Created by IntelliJ IDEA.
User: anton
Date: 29.04.2021
Time: 20:37
To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Login Page</title>
    <style>
        @import url("https://rms.me/inter/inter-ui.css");
        ::selection {
            background: #2d2f36;
        }
    </style>

```



```

::-webkit-selection {
    background: #2d2f36;
}
::-moz-selection {
    background: #2d2f36;
}
body {
    background-image:
url("https://i.pinimg.com/originals/46/1d/25/461d254f8563f8945ce3a385f289df17.jpg");
    font-family: "Inter UI", sans-serif;
    margin: 0;
}
.page {
    background-image:
url("https://i.pinimg.com/originals/46/1d/25/461d254f8563f8945ce3a385f289df17.jpg");
    display: flex;
    flex-direction: column;
    height: calc(100%);
    position: absolute;
    place-content: center;
    width: calc(100%);
}
@media (max-width: 767px) {
    .page {
        height: auto;
        margin-bottom: 20px;
        padding-bottom: 20px;
    }
}
.container {
    display: flex;
    height: 320px;
    margin: 0 auto;
    width: 640px;
}
@media (max-width: 767px) {
    .container {
        flex-direction: column;
        height: 630px;
        width: 320px;
    }
}
.left {
    background: rgba(255, 255, 255, 0);
    height: calc(100% - 40px);
    top: 20px;
    position: relative;
    width: 50%;
}
@media (max-width: 767px) {
    .left {
        height: 100%;
        left: 20px;
        width: calc(100% - 40px);
        max-height: 270px;
    }
}
.login {
    font-size: 50px;
}

```

```

        font-weight: 900;
        margin: 50px 40px 40px;
    }
    .eula {
        color: #999;
        font-size: 14px;
        line-height: 1.5;
        margin: 40px;
    }
    .right {
        background: #474a59;
        box-shadow: 0px 0px 40px 16px rgba(0, 0, 0, 0.22);
        color: #f1f1f2;
        position: relative;
        width: 50%;
    }
    @media (max-width: 767px) {
        .right {
            flex-shrink: 0;
            height: 100%;
            width: 100%;
            max-height: 350px;
        }
    }
    svg {
        position: absolute;
        width: 320px;
    }
    path {
        fill: none;
        stroke: url(#linearGradient);
        stroke-width: 4;
        stroke-dasharray: 240 1386;
    }
    .form {
        margin: 40px;
        position: absolute;
    }
    label {
        color: #c2c2c5;
        display: block;
        font-size: 14px;
        height: 16px;
        margin-top: 20px;
        margin-bottom: 5px;
    }
    input {
        background: transparent;
        border: 0;
        color: #f2f2f2;
        font-size: 20px;
        height: 30px;
        line-height: 30px;
        outline: none !important;
        width: 100%;
    }
    input::-moz-focus-inner {
        border: 0;
    }

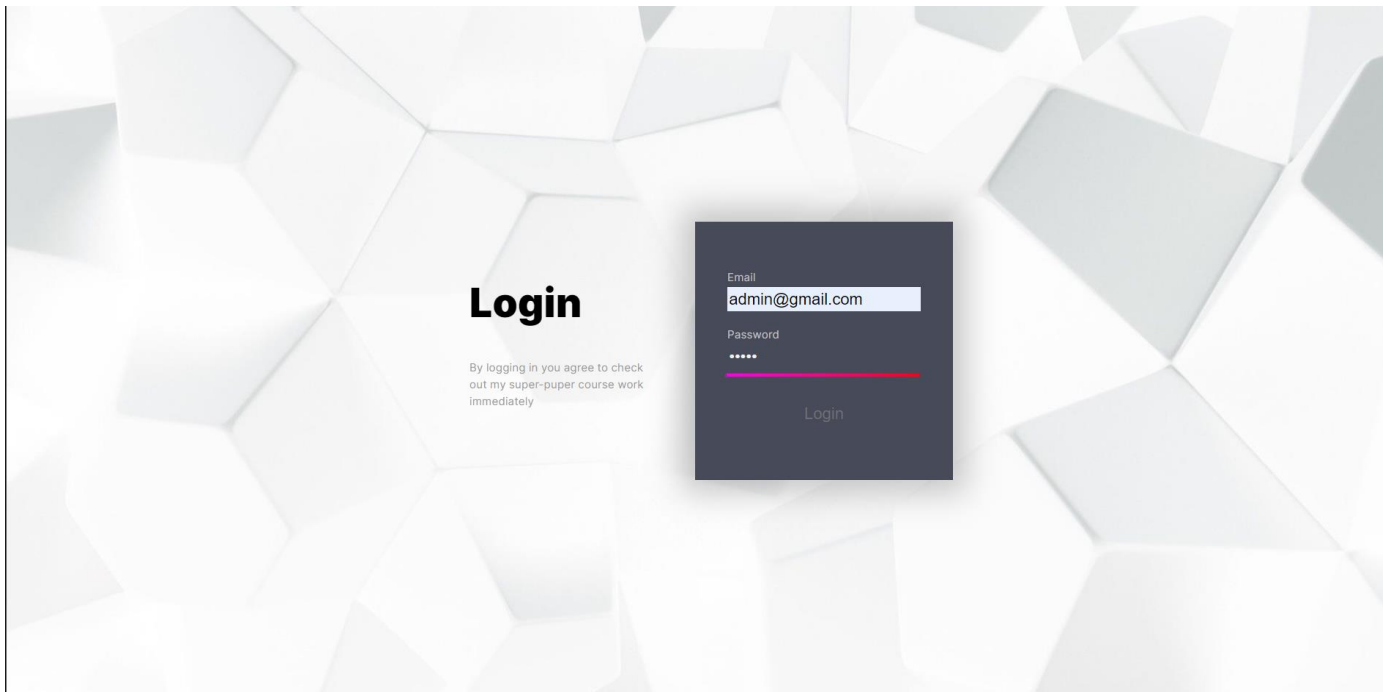
```

```

#submit {
    color: #707075;
    margin-top: 40px;
    transition: color 300ms;
}
#submit:focus {
    color: #f2f2f2;
}
#submit:active {
    color: #d0d0d2;
}

</style>
</head>
<body>
<div class="page">
    <div class="container">
        <div class="left">
            <div class="login">Login</div>
            <div class="eula">By logging in you agree to check out my super-puper
course work immediately</div>
        </div>
        <div class="right">
            <svg viewBox="0 0 320 300">
                <defs>
                    <linearGradient inkscape:collect="always" id="linearGradient"
x1="13" y1="193.49992" x2="307" y2="193.49992" gradientUnits="userSpaceOnUse">
                        <stop style="stop-color:#ff00ff;" offset="0" id="stop876" />
                        <stop style="stop-color:#ff0000;" offset="1" id="stop878" />
                    </linearGradient>
                </defs>
                <path d="m 40,120.00016 239.99984,-3.2e-4 c 0,0 24.99263,0.79932
25.00016,35.00016 0.008,34.20084 -25.00016,35 -25.00016,35 h -239.99984 c 0,-0.0205 -
25,4.01348 -25,38.5 0,34.48652 25,38.5 25,38.5 h 215 c 0,0 20,-0.99604 20,-25 0,-
24.00396 -20,-25 -20,-25 h -190 c 0,0 -20,1.71033 -20,25 0,24.00396 20,25 20,25 h
168.57143" />
            </svg>
            <form method="post" action="/do/login" class="form" >
                <label for="email">Email</label>
                <input type="email" id="email" name="login">
                <label for="password">Password</label>
                <input type="password" id="password" name="password">
                <input type="submit" id="submit" value="Login">
            </form>
        </div>
    </div>
</div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/animejs/2.0.2/anime.js"></script>
<script src="backend.js" charset="utf-8"></script>
<script src="/javascript/animation.js"></script>
</body>
</html>

```



Register.jsp

```
<!--
Created by IntelliJ IDEA.
User: anton
Date: 30.04.2021
Time: 16:45
To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Login Page</title>
    <style>
        @import url("https://rsms.me/inter/inter-ui.css");
        ::selection {
            background: #2d2f36;
        }
        ::-webkit-selection {
            background: #2d2f36;
        }
        ::-moz-selection {
            background: #2d2f36;
        }
        body {
            background-image:
url("https://i.pinimg.com/originals/46/1d/25/461d254f8563f8945ce3a385f289df17.jpg");
            font-family: "Inter UI", sans-serif;
            margin: 0;
        }
        .page {
            background-image:
url("https://i.pinimg.com/originals/46/1d/25/461d254f8563f8945ce3a385f289df17.jpg");
            display: flex;
            flex-direction: column;
            height: calc(100%);
            position: absolute;
```

```

        place-content: center;
        width: calc(100%);
    }
    @media (max-width: 767px) {
        .page {
            height: auto;
            margin-bottom: 20px;
            padding-bottom: 20px;
        }
    }
    .container {
        display: flex;
        height: 320px;
        margin: 0 auto;
        width: 640px;
    }
    @media (max-width: 767px) {
        .container {
            flex-direction: column;
            height: 630px;
            width: 320px;
        }
    }
    .left {
        background: rgba(255, 255, 255, 0);
        height: calc(100% - 40px);
        top: 20px;
        position: relative;
        width: 50%;
    }
    @media (max-width: 767px) {
        .left {
            height: 100%;
            left: 20px;
            width: calc(100% - 40px);
            max-height: 270px;
        }
    }
    .login {
        font-size: 50px;
        font-weight: 900;
        margin: 50px 40px 40px;
    }
    .eula {
        color: #999;
        font-size: 14px;
        line-height: 1.5;
        margin: 40px;
    }
    .right {
        background: #474a59;
        box-shadow: 0px 0px 40px 16px rgba(0, 0, 0, 0.22);
        color: #f1f1f2;
        position: relative;
        width: 50%;
    }
    @media (max-width: 767px) {
        .right {
            flex-shrink: 0;

```

```

        height: 100%;
        width: 100%;
        max-height: 350px;
    }
}
svg {
    position: absolute;
    width: 320px;
}
path {
    fill: none;
    stroke: url(#linearGradient);
    stroke-width: 4;
    stroke-dasharray: 240 1386;
}
.form {
    margin: 40px;
    position: absolute;
}
label {
    color: #c2c2c5;
    display: block;
    font-size: 14px;
    height: 16px;
    margin-top: 20px;
    margin-bottom: 5px;
}
input {
    background: transparent;
    border: 0;
    color: #f2f2f2;
    font-size: 20px;
    height: 30px;
    line-height: 30px;
    outline: none !important;
    width: 100%;
}
input::-moz-focus-inner {
    border: 0;
}
#submit {
    color: #707075;
    margin-top: 40px;
    transition: color 300ms;
}
#submit:focus {
    color: #f2f2f2;
}
#submit:active {
    color: #d0d0d2;
}

```

```

</style>
</head>
<body>
<div class="page">
    <div class="container">
        <div class="left">
            <div class="login">Sign-Up</div>

```

```

        <div class="eula">By registration in you agree to check out my super-puper
course work immediately</div>
    </div>
    <div class="right">
        <svg viewBox="0 0 320 300">
            <defs>
                <linearGradient inkscape:collect="always" id="linearGradient"
x1="13" y1="193.49992" x2="307" y2="193.49992" gradientUnits="userSpaceOnUse">
                    <stop style="stop-color:#ff00ff;" offset="0" id="stop876" />
                    <stop style="stop-color:#ff0000;" offset="1" id="stop878" />
                </linearGradient>
            </defs>
            <path d="m 40,120.00016 239.99984,-3.2e-4 c 0,0 24.99263,0.79932
25.00016,35.00016 0.008,34.20084 -25.00016,35 -25.00016,35 h -239.99984 c 0,-0.0205 -
25,4.01348 -25,38.5 0,34.48652 25,38.5 25,38.5 h 215 c 0,0 20,-0.99604 20,-25 0,-
24.00396 -20,-25 -20,-25 h -190 c 0,0 -20,1.71033 -20,25 0,24.00396 20,25 20,25 h
168.57143" />
        </svg>
        <form method="post" action="/do/register" class="form" >
            <label for="email">Email</label>
            <input type="email" id="email" name="login">
            <label for="password">Password</label>
            <input type="password" id="password" name="password">
            <input type="submit" id="submit" value="Register">
        </form>
    </div>
</div>
</div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/animejs/2.0.2/anime.js"></script>
<script src="backend.js" charset="utf-8"></script>
<script src="/javascript/animation.js"></script>
</body>
</html>

```

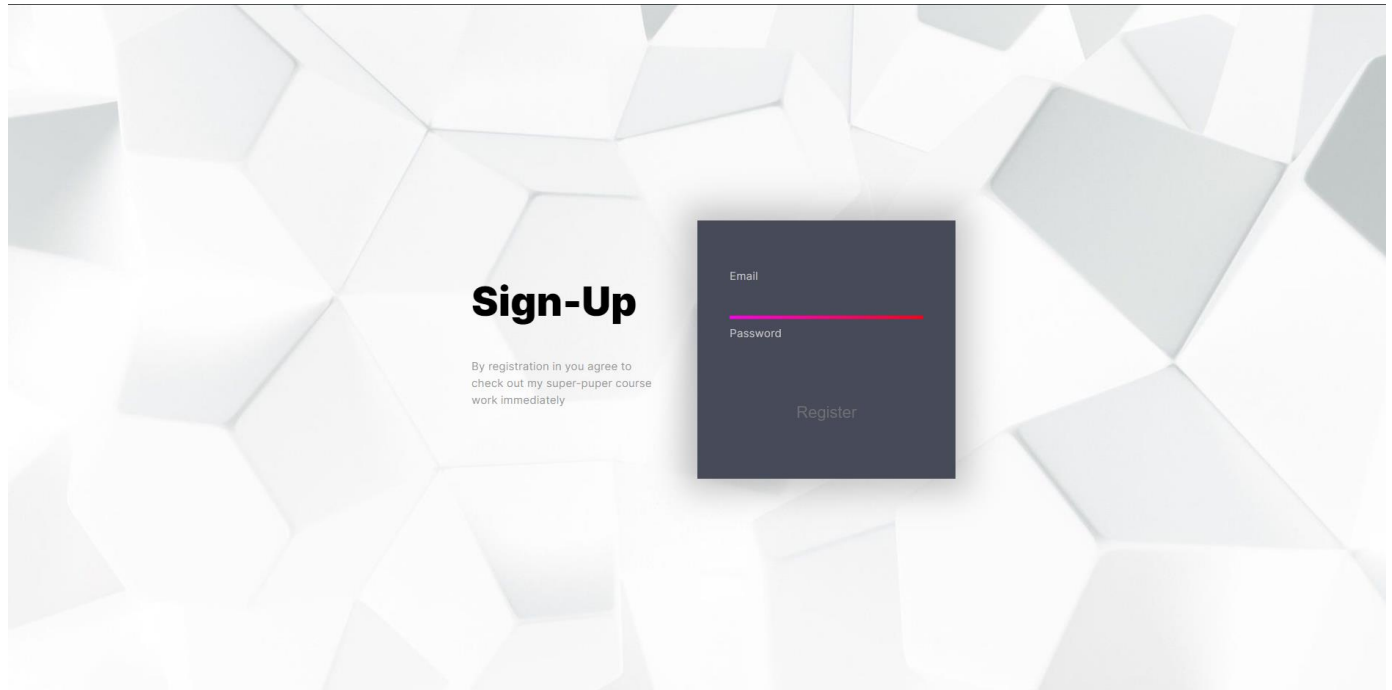
Error.jsp

```

<%--
    Created by IntelliJ IDEA.
    User: anton
    Date: 27.04.2021
    Time: 20:05
    To change this template use File | Settings | File Templates.
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Error</title>
    <%@include file="style.jspf"%>
</head>
<body>
<%@include file="header.jspf"%>
<section>
    <h1>${message}</h1>
    <a href=".">Go to main page</a>
</section>
<br>
<%@include file="footer.jspf"%>

```

```
</body>
</html>
```



Header.jsp

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page pageEncoding="UTF-8" %>
<header>
  <div class="topnav">
    <a class="active" href="/do/">Home</a>
    <c:if test="${!empty user}">
      <div class="creating">
        <form action="vacnew" method="POST">
          <button type="submit">Create New Vacancy</button>
        </form>
      </div>
    </c:if>
    <c:if test="${!empty user}">
      <div class="login-container">
        <form action="logout" method="POST">
          <c:out value="${user.name}"/>
          <button type="submit">Logout</button>
        </form>
      </div>
    </c:if>
    <c:if test="${empty user}">
      <div class="login-container">
        <form action="../../login.jsp" method="POST">
          <button type="submit">Login</button>
          <button type="submit"
formaction="../../register.jsp">Register</button>
        </form>
      </div>
    </c:if>
    <div class="search-container">
      <form action="search">
        <input type="text" placeholder="Search.." name="search">
        <button type="submit">🔍</button>
      </form>
    </div>
  </div>
```



```
        </form>
    </div>
</div>
</header>
```

Footer.jsp

```
<%@ page pageEncoding="UTF-8" %>
<footer>
    <div class="footer">
        <div class="container">Copyright &copy; 2021 Anton Chaban</div>
    </div>
</footer>
```