Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.912 *На правах рукопису*

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: "*HR-сервіс підбору персоналу для IT із автоматизованим оцінюванням компетенцій*"

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ – __.__.__.___

Студент
ІПЗ-44 мс _____ /Дмитро ГАРБАР/

Науковий керівник
асист. _____ /Кирило КАДОМСЬКИЙ/

Консультант
з питань нормоконтролю
фахівець _____ /Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри
д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2021

Рішенням Екзаменаційної комісії
випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії

професор, доктор техн. наук Андрій БОНДАРЧУК

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО
Зав. кафедри програмних систем і технологій
_____(Олексій БИЧКОВ)
„_____”_____ 2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
СТУДЕНТУ

Гарбару Дмитру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема випускної кваліфікаційної бакалаврської роботи “HR-сервіс підбору персоналу для IT із автоматизованим оцінюванням компетенцій ”, керівник роботи асист. Кадомський К.К., затверджені на засіданні кафедри програмних систем і технологій, протокол №6 від «11» листопада 2020р.
2. Строк здачі студентом закінченої роботи _____
3. Вихідні дані до роботи: теоретичні відомості щодо розробки веб-додатків
4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)

1. Аналіз і огляд процесів підбору персоналу.
2. Проектування сервісу та платформи для автоматизації підбору персоналу.
3. Особливості реалізації платформи.
4. Опис здобутих результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов’язковихкреслень)

1. Етапи процесу найму (рис.1.1, стор. 12)
2. Етапи обробки речень (рис.2.1., стор. 20)
3. UML діаграма процесу взаємодії через платформу (3.5., стор. 37)
4. Схема класів модулю прескринінгу (3.8., стор. 39)

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
РОЗДІЛ 1	Кирило КАДОМСЬКИЙ		
РОЗДІЛ 2	Кирило КАДОМСЬКИЙ		
РОЗДІЛ 3	Кирило КАДОМСЬКИЙ		
РОЗДІЛ 4	Кирило КАДОМСЬКИЙ		

7. Дата видачі завдання «05» листопада 2020 р.

Керівник _____ (Кирило КАДОМСЬКИЙ)

Завдання прийняв до виконання _____(Дмитро ГАРБАР)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Уточнення постановки задачі	10.11.2020	
2	Аналіз літератури	03.12.2020	
3	Аналіз існуючих методів, концепцій та алгоритмів вирішення завдання	16.01.2021	
4	Опис розробленого алгоритму	01.02.2021	
5	Проектування програмного забезпечення	12.03.2021	
6	Розроблення програмного забезпечення	10.04.2021	
7	Тестування розробленого програмного забезпечення	16.05.2021	
8	Оформлення пояснювальної записки	06.06.2021	

Студент – бакалавр_____(Дмитро ГАРБАР)

Керівник роботи _____(Кирило КАДОМСЬКИЙ)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 70 с., 30 рис., 1 додат., 14 джерел.

Тема: HR-сервіс підбору персоналу для IT із автоматизованим оцінюванням компетенцій

Об'єкт дослідження: процес рекрутингу кваліфікованих кадрів**.**

Мета роботи: зменшення ресурсів на процес рекрутингу кандидатів за їх компетенцією в IT та в цілому.

Предмет дослідження: способи автоматизації відбору кандидатів за вимогами вакансії.

Результати дослідження:

Платформа та сервіс можуть застосовуватись HR-відділами на комерційних підприємствах для автоматизації підбору персоналу. Розроблена платформа призначення для комерційного використання та знаходиться на стадії MVP для залучення інвестицій. Проект бере участь у всеукраїнському конкурсі стартапів The Ukrainian Startup National Digital Competition 2021 від fibstartup.

Висновок:

Вирішується задача по зменшенню ресурсів на етапі відбору кандидатів за їх компетенцією в IT та в цілому. Запропонований метод семантичного аналізу для визначення міри співпадіння з правильною відповіддю був впроваджений в модуль прескринінгу кандидатів в платформі для рекрутингу. Це дозволило зменшити кількість кандидатів які не задовольняють вимогам до вакансії.

РЕКРУТИНГ, ПРЕСКРИНІНГ, СЕМАНТИЧНИЙ АНАЛІЗ, АВТОМАТИЧНЕ ОЦІНЮВАННЯ ВІДКРИТИХ ПИТАНЬ, CORENLP, SPRING FRAMEWORK

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 70 с., 30 рис., 1 прил., 14 источников.

Тема: HR-сервис подбора персонала для IT с автоматизированным тестированием компетенций

Объект исследования: процесс рекрутинга квалифицированных кадров.

Цель работы: Уменьшение ресурсов на процесс рекрутинга кандидатов за их компетенцией в ИТ и в целом.

Предмет исследования: способы автоматизации отбора кандидатов по требованиям вакансии.

Результаты исследования:

Платформа и сервис могут применяться HR-отделами на коммерческих предприятиях для автоматизации подбора персонала. Разработанная платформа, предназначенная для коммерческого использования, и находится на стадии MVP для привлечения инвестиций. Проект участвует во всеукраинском конкурсе стартапов The Ukrainian Startup National Digital Competition 2021 от fibstartup.

Вывод:

Решается задача по уменьшению ресурсов на этапе отбора кандидатов по их компетенциям в ИТ и в целом. Предложенный метод семантического анализа для определения степени совпадения с правильным ответом был внедрен в модуль прескринингу кандидатов в платформе для рекрутинга. Это позволило уменьшить количество кандидатов, не удовлетворяющих требованиям к вакансии.

РЕКРУТИНГ, ПРЕСКРИНИНГ, СЕМАНТИЧЕСКИЙ АНАЛИЗ, АВТОМАТИЧЕСКОЕ ОЦЕНИВАНИЕ ОТКРЫТЫХ ВОПРОСОВ, CORENLP, SPRING FRAMEWORK

ANNOTATION

Graduation qualifying bachelor's thesis: 70 pp., 30 fig., 14 sources, 1 supplement.

Topic: HR-service for recruiting personnel for IT with automated testing of competencies

Object of study: the process of recruiting qualified personnel.

The goal of the work: Reducing resources for the recruiting process of candidates for their competence in IT and in general.

Subject of study: ways to automate the selection of candidates according to the requirements of a vacancy.

Results of the research:

The platform and service can be used by HR departments in commercial enterprises to automate recruitment. The developed platform is intended for commercial use and is at the MVP stage to attract investments. The project participates in the Ukrainian Startup National Digital Competition 2021 from fibstartup.

Conclusion:

The task of reducing resources at the stage of selecting candidates according to their competencies in IT and in general is being solved. The proposed method of semantic analysis to determine the degree of coincidence with the correct answer was implemented in the candidate pre-screening module in the recruiting platform. This made it possible to reduce the number of candidates who did not meet the vacancy requirements.

RECRUITING, PRESCRINING, SEMANTIC ANALYSIS, AUTOMATIC ASSESSMENT OF OPEN QUESTIONS, CORENLP, SPRING FRAMEWORK.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

MVP – Minimum viable product, мінімально життєздатний продукт.

JSON – JavaScript Object Notation, текстовий формат обміну даних, заснований на мові програмування JavaScript.

SQL – Structured Query Language, декларативна мова програмування, що застосовується для управління, створення та модифікації даних в реляційній базі, керованої відповідною системою управління базами даних.

API – Application Programming Interface, опис з способів якими одна комп'ютерна програма може взаємодіяти з іншою програмою.

MVC – Model-View-Controller, архітектура побудови програмного забезпечення, що полягає в розділені даних застосунку, користувацького інтерфейсу та бізнес логіки.

ORM – Object Relational Mapping, технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування.

DTO – Data Transfer Object, об'єкт який пересилає дані між модулями.

JVM – Java Virtual Machine, механізм, що забезпечує середовище виконання для керування кодом Java.

DOM – Document Object Model, незалежний від платформи і мови програмний інтерфейс, що дозволяє програмам і скриптам отримати доступ до вмісту HTML, XHTML і XML документів.

HTML – HyperText Markup Language, мова розмітки гіперктекстових документів.

ВСТУП

Актуальність роботи

На сьогоднішній день, завдання підбору персоналу стоять практично перед будь-яким підприємством. Пошук і відбір персоналу – це основа системи управління персоналом підприємства, що має більш якісно оцінюватися та коригуватися саме на стадії формування його кадрового складу підприємства.

Компанії зацікавлені в швидкому підборі кандидатів які можуть покрити специфічні проблеми та влитися в колектив. При пошуку працівників, основні грошові та часові витрати компанії, йдуть саме на процес виокремлення кандидата з найбільшим рівнем співпадіння професійних та особистих якостей до вимог. Особливо складно знайти якісного кандидата початківця через перенасиченість ринку низько кваліфікованими фахівцями. Часто виникає ситуація, коли бажане видають за дійсне. Рекрутери довго не можуть знайти кандидатів, а початківцям немає звідки взяти практичний досвід роботи. Тому вони йдуть на обман і додають у своєму резюме досвід роботи та навички яких вони не мають. Також погіршують ситуацію неефективні витрати часу спеціаліста компанії на проведення інтерв'ю.

Мета і задачі дослідження

Метою дипломної роботи є зменшення ресурсів на процес рекрутингу кандидатів за їх компетенцією в IT та в цілому.

Досягнення мети включало розв'язання таких задач:

1) аналіз існуючих засобів та процесів рекрутингу;
2) формулювання невдоволеної наявної проблеми в процесі рекрутингу;
3) проектування рішення для вирішення сформованої проблеми;
4) розробка системи для рекрутингу.

Об'єкт дослідження є процес рекрутингу кваліфікованих кадрів**.**

Предмет дослідження є способи автоматизації відбору кандидатів за вимогами вакансії.

Методи дослідження: інтерв'ю з рекрутерами малих, середніх, та великих компаній. Методи системного аналізу для порівняння існуючих систем та процесів для рекрутингу. Методи обробки природної мови для автоматизації аналізу відкритих питань.

Практичне значення одержаних результатів.

Платформа та сервіс можуть застосовуватись HR-відділами на комерційних підприємствах для автоматизації підбору персоналу. Розроблена платформа призначення для комерційного використання та знаходиться на стадії MVP для залучення інвестицій. Проект бере участь у всеукраїнському конкурсі стартапів The Ukrainian Startup National Digital Competition 2021 від fibstartup.

Публікації.

За реалізацією визначення компетенції кандидатів за відкритими питаннями, підготовлено доповідь для участі в міжнародній науково-практичній Інтернет конференції "Глобальні та регіональні проблеми інформатизації в суспільстві і природокористуванні 2021", секція Технології обробки даних та розробки програмних систем, яка відбулася 13-14 травня 2021 р. в м. Києві на базі факультету інформаційних технологій Національного університету біоресурсів і природокористування.

Структура та обсяг роботи.

Дипломна робота складається з: вступу, чотирьох розділів, що включають 29 підрозділів, висновків, переліку джерел посилань із 14 найменувань. У листі дипломної роботи міститься 30 рисунків та 2 таблиці. Загальний обсяг роботи 70 листів.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО ПІДБОРУ ПЕРСОНАЛУ

1.1 Аналіз процесу рекрутингу

Наразі важливе завдання, як пошук персоналу, набуло значної актуальності у роботі сучасних підприємств. Це сприяло виникненню такого процесу на ринку праці як «рекрутинг». Зростає актуальність завдання ефективного оцінювання, яке враховує всі можливі показники роботи того чи іншого відділу підприємства. Керівники підприємств та організацій прагнуть бачити на роботі висококваліфікованих спеціалістів, тому необхідно приділяти значну увагу процесам рекрутингу та неодмінно удосконалювати методи та шляхи пошуку персоналу.

Процес рекрутингу складається з процесу пошуку, відбору та найму нових працівників у компанію. Цей процес має три ключові сегменти: планування, набір та відбір співробітників. Більш детально, послідовні етапи, зображені на рис 1.1.

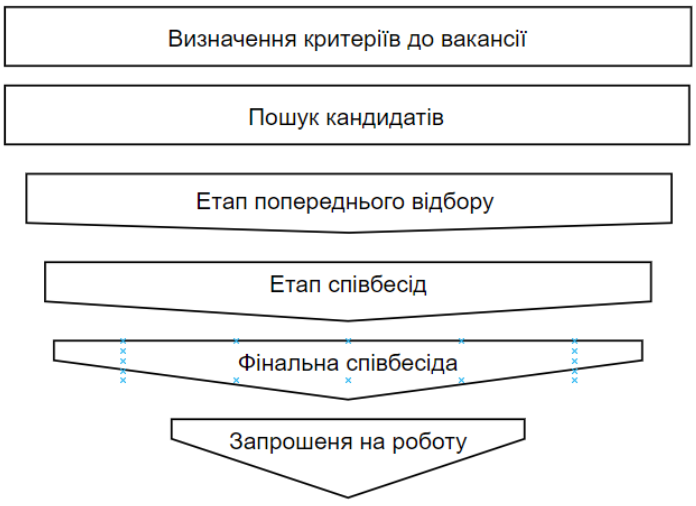


Рис. 1.1 Етапи процесу найму

Першим етапом є визначення критеріїв до вакансії, тобто компанії необхідно чітко розуміти хто саме необхідний команді. В основі цього процесу лежить

підготовка посадової інструкції – документа який описує основні функції співробітника, що займає дане робоче місце. Зазвичай посадова інструкція готується відділом кадрів разом із керівниками вакантних відділів. Використовуючи посадові інструкції для оцінки кандидатів на вакантні посади, експерти повинні визначити, наскільки кандидат співпадає з вимогами. Для того, щоб полегшити процес відбору кандидатів, в організаціях створюється документи з описом основних характеристик, які повинні мати співробітники для успішної роботи на посаді.

Кваліфікаційна картка – це сукупність різних кваліфікаційних характеристик які описують ідеального працівника на посаді. Як правило, вона складаються з вимог до рівня освіти, володіння іноземними мовами, різними технічними знаннями в сфері тощо. Еталонні рівні вимог за кожним критерієм розробляються виходячи з характеристик вже працюючих на підприємстві працівників які добре справляються зі своїми обов'язками. Відбір може зайняти багату часу якщо список вимог до працівника з боку організації буде занадто великий. Кваліфікаційна картка дозволяє оцінити кандидатів за кожним критерієм та порівняти кандидатів між собою. Але вона не є найліпшим інструментом, тому що дозволяє визначити наявність кваліфікаційних характеристик, а не наявність практичних здібностей виконувати певні задачі. Також цей метод фокусується на набутих знаннях кандидата, та не враховує особистих характеристик та потенціалу професійного розвитку.

Карта компетенції допомагає представити особисті характеристики людини, типи поведінки та соціальні ролі кандидата. Наприклад звернення уваги на інтереси клієнтів, здатність працювати в команді, впевненість у собі та оригінальність мислення. Складання карт компетентності вимагає спеціальних знань, тому вони складаються за допомогою професійних консультантів або спеціально навчених працівників відділу кадрів. При оцінці кандидатів карта здібностей також використовується як кваліфікаційна картка, тобто компетенції кандидата порівнюється із компетенціями ідеального працівника.

Визначивши вимоги до кандидата, відділ персоналу може приступити до реалізації наступного етапу – залучення кандидатів, основне завдання якого це створення списку кваліфікованих кандидатів для подальшого відбору. На етапі

пошуку кандидатів, рекрутер часто використовує два способи пошуку технічних експертів: активний та пасивний. Активний набір охоплює роботу на веб-сайтах LinkedIn та Djinni, Dou та різних платформах для пошуку найкращих IT талантів, а також рекомендації працівників компанії. Пасивний же складається із платформ для пошуку роботи та інших соціальних мереж.

Попередній відбір проводиться на основі, офіційно встановлених мінімальних вимог та перегляду резюме або профілю користувача. Метою етапу попереднього відбору є зменшення витрат за рахунок зменшення кількості претендентів, що підлягають більш детальній оцінці на наступному етапі. Протягом цього періоду рекрутери будуть наполегливо працювати над тим, щоб зменшити потік невідповідних кандидатів на основі критеріїв вакансій. Проаналізувавши резюме та профілі кандидатів буде сформований "довгий список", що представляє собою список усіх кандидатів які відповідають вимогам. Найголовнішим критерієм на цьому етапі це наявність комерційного досвіду в сфері, так як людина що займалася подібною діяльністю раніше, затратить менше часу на навчання та принесе в команду нові знання.

Наступним кроком є проведення більш детальної експертизи кандидатів. Формується так званий «короткий список», до якого входять кандидати, які пройшли кілька перших етапів. Цей етап дуже трудомісткий, оскільки рекрутер проводить коротке інтерв’ю для оцінки вмінь кандидата яких немає у резюме: робота в команді, соціальна взаємодія, робота в команді тощо. Для рекрутерів одним із ключових критеріїв відбору на етапі співбесіди може бути тип особистості кандидата. Наприклад для роботи, яка вимагає великої співпраці з людьми, кандидати з кращими соціальними навичками будуть більш пріоритетними, ніж добре освічені, але більш замкнуті кандидати. Також на цьому етапі, кандидату може бути запропоноване проходження тесту для визначення компетенції. Але тест не може надати всебічну характеристику навичкам кандидата, та багато кандидатів, просто відмовляються продовжувати процес найму. Тому для визначення реальних навичок залучається спеціаліст який вже підтвердив свою компетентність. Тобто чим більше кандидатів відфільтрувалися на попередніх етапах, тим більше часу спеціаліста буде збережено.

Після проходження всіх етапів, найкращі кандидати можуть бути запрошені на фінальну співбесіду з командою та керівниками підрозділів для прийняття фінального рішення.

1.2 Формулювання основних проблем в сучасному рекрутингу

Після аналізу процесів та проведення інтерв'ю з кандидатами та рекрутерами, можна визначити основні проблеми з якими вони стикаються.

Наприклад 73% шукачів роботи зазначають що рекрутер не є достатньо кваліфікованим щоб описати потенційні задачі або технічні аспекти вакансії [1]. Можна сказати, що це не задача рекрутера, але типове рішення по залученню спеціаліста є доволі ресурсозатратним, та інколи не є доступним.

Наступна проблема витікає з попередньої, а саме засипання кандидатів нерелевантними вакансіями. З цією проблемою стикається 61% кандидатів після цього в них з'являється ментальний блок на всі вакансії [1].

Також 76% кандидатів вважають що немає нічого гіршого, ніж пройти весь процес набору в організації та змусити чекати рішення про найм, яке ніколи не прийде [2]. Менеджери з найму повинні бути оперативними та чесними, коли надсилають лист про відмову. Щоб вирішити цю проблему, рекрутерам потрібно затратити більше зусиль для відслідковування кожного кандидата та його етапу найму, але це дозволяє уникнути поганого досвіду взаємодії з компанією збоку кандидата.

Поганий досвід роботи з кандидатами може легко змусити людей поспішати ділитися своїми історіями про компанію на сайтах роботодавців та соціальних мережах. За даними CareerArc, 72% тих, хто шукає роботу, повідомили, що діляться поганим досвідом кандидатів на таких сайтах, як Glassdoor, у соціальних мережах або безпосередньо з колегою чи другом [3]. Такий досвід може обмежити майбутній фонд талантів компанії і в кінцевому підсумку коштуватиме більше часу та грошей. Опитування LinkedIn показало, що 27% кандидатів, які мали негативний досвід,

“активно відбиватимуть” інших претендувати на роботу в цій компанії [4]. Це означає, що більше чверті тих, хто шукає роботу, які мали менше ніж хороший досвід в процесі, можуть активно зашкодити рекрутинговим зусиллям компанії. З часом це означає, що стає важче залучати кандидатів (особливо при наймі на подібні посади), змушуючи бізнес звертатися до менш кваліфікованих претендентів на порожні робочі місця, оскільки найбажаніші кандидати більше не зацікавлені там працювати. Крім того, збиток нанесений бренду внаслідок поганого процесу найму, може зашкодити сприйняттю бренду, а отже, продажам та прибуткам. Наприклад, 41% заявників із поганим кандидатом втрачають лояльність до бренду та уникають придбання продукції цієї компанії [5].

Ключова проблема компаній при наймі, це саме відокремлення компетентних працівників. Якість цього процесу залежить в основному, від проведених співбесід та кваліфікації працівників. Рекрутингові агенства та великі компанії, можуть мати локальну базу кандидатів де можуть вести певні замітки про них, але ця інформація не є доступною іншим компаніям де цей кандидат безпосередньо проходить співбесіду. І навряд чи кандидат сам розповість про інциденти на попередній роботі. Особливо це критично при значній ціні помилки в процесі найму. За статистикою, компанії в США витрачають близько 4 тисяч доларів та 52 дні, щоб наняти нового співробітника, а кожен промах коштує компанії близько 15 тис. доларів [6].

1.3 Аналіз існуючих методів оцінки компетенції

Найрозповсюджений метод це аналіз портфоліо яке містить докази знань, здібностей чи компетенції на основі минулого досвіду. Досвід вказаний в портфоліо може бути потужними для демонстрації компетентності, оскільки він дає чіткі докази того, що зробила людина. Відгуки роботодавців, однолітків та клієнтів можуть бути включені в портфоліо. Воно може описати невдалі приклади разом із виправною роботою, яку кандидат зробив для усунення недоліку. Даний метод піддається

автоматизації при підтримці однієї структури резюме всіма кандидатами або за допомогою спеціальних аналізаторів тексту.

Одним із способів оцінки компетенцій є безпосереднє спостереження за кандидатом який виконує практичне завдання в режимі реального часу. Це дозволяє оцінити фактичні процеси, що застосовуються при виконані специфічної задачі. Оцінка фокусується на процесі та результатах. Безпосереднє спостереження можна проводити на робочому місці або в іншому середовищі за допомогою технологічних засобів. Також, дуже розповсюджене використання відеозаписів. Оцінювач не може оцінити процес мислення кандидата за допомогою відеозапису, але це може бути виправлено шляхом подальших співбесід та опитувань. Оскільки безпосереднє спостереження обмежується тим, що відбувається на той момент, воно, як правило, поєднується з іншими інструментами для охоплення частин роботи, які трапляються рідше. Такий спосіб складно автоматизувати, адже важко відстежити виконання всіх критерій до поставленої задачі.

Тести - це більш традиційний метод оцінки знань. У деяких випадках є єдиним варіантом щоб зробити оцінку практичною та економічно ефективною. Тест може бути у формі співбесіди, письмових тестів або самооцінки. Використовуються різні види стилів опитування, включаючи вибір варіантів, короткі відповіді та інші відкриті питання. Але за результатами тестів дуже складно скласти всебічне представлення реальних компетенцій кандидата. Тестування є найлегшим із запропонованих методів для автоматизації.

Висновки до розділу

Проаналізовані існуючі процеси та методи рекрутингу. Також була зібрана статистика існуючих проблем. Серед них була виділена проблема оцінки компетенції співробітників. Було проаналізовані існуючі методи її вирішення та вибраний оптимальний для реалізації автоматизації.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ

2.1 Метод визначення семантичної подібності речень

Семантична подібність - це метрика, що визначається набором термінів, де ідея відстані між предметами базується на подібності їх значення або семантичному змісті на відміну від лексикографічної подібності. Для представлення міри семантичного зв’язку за допомогою числового опису, між одиницями мови або поняттями, використовуються різні математичні інструменти. Ці інструменти порівнюють з інформацією яка підтверджує їх значення або опису їх природи. Термін семантична подібність часто плутають із семантичною спорідненістю. Семантична спорідненість включає будь-яке відношення між двома термінами, тоді як семантична подібність включає лише відносини "є". Наприклад, "машина" схожа на "автобус", але також пов'язана з "дорогою" та "керуванням автомобілем".

Обчислити семантичну подібність можна, визначивши топологічну подібність, використовуючи онтології для визначення відстані між термінами або поняттями. Наприклад, наївна метрика для порівняння понять, упорядкованих у частково впорядкованому наборі та представлених як вузли спрямованого ациклічного графа, буде найкоротшим шляхом, що пов'язує два вузли поняття. Семантичну спорідненість між одиницями мови можна оцінити на основі аналізу тексту, використовуючи статистичні засоби, такі як векторна модель простору для кореляції слів та текстового контексту з відповідної частини тексту. Оцінка показників семантичної подібності або спорідненості оцінюється двома основними способами. Перший заснований на використанні наборів даних, розроблених експертами та складених із пар слів з їх оцінкою семантичної подібності та спорідненості. Другий спосіб заснований на інтеграції заходів всередині конкретних додатків, таких як пошук інформації, рекомендаційні системи, обробка природної мови тощо.

2.1.1 Визначення

X та Y – множини виокремлених смислових елементів тексту, таких як нормальні форми слів або граматичні сполучення слів.

$simFunc(x_i, y_i) \in [0;\ 1]$ – функція, що оцінює міру семантичної подібності над множиною $X \times Y$.

$Exclusive(X, Y, simFunc)$ – набір перших n пар елементів X та Y із множини $\{(x_i, y_j) \in X \times Y\}$, таких що $x_i ! = x_k$ при $i \neq k$ та $y_j ! = y_k$ при $j \neq k$, які мають найвище значення $simFunc(x_i, y_i)$. Далі будемо називати такі пари ексклюзивними.

Трійка – це множина с трьох значень, суб'єкта, зв'язку та об'єкта.

2.1.2 Алгоритм роботи методу

Для автоматизованого визначення компетенцій за допомогою відкритих питань, в роботі реалізовано метод, що використовує оцінку семантичної подібності відповіді кандидата до еталону. В представленому методі, вимірювання семантичної подібності базується на принципі композиційності, згідно з яким семантичне значення тексту визначається набором відношень між його складовими частинами. Обробка речень тексту відбувається у кілька послідовних етапів зображених на рис. 2.1.

На першому етапі виконується пошук поєднаних частин речення, що утворюють трійки, складені із суб'єкта, зв'язку та об'єкта. Цей етап виконується за допомогою спеціального функціоналу Open Information Extraction і названий відповідно нього OpenAI. Наприклад, реченню "Apple invented Swift" відповідає одна трійка: $'invent'('Apple', 'Swift')$.

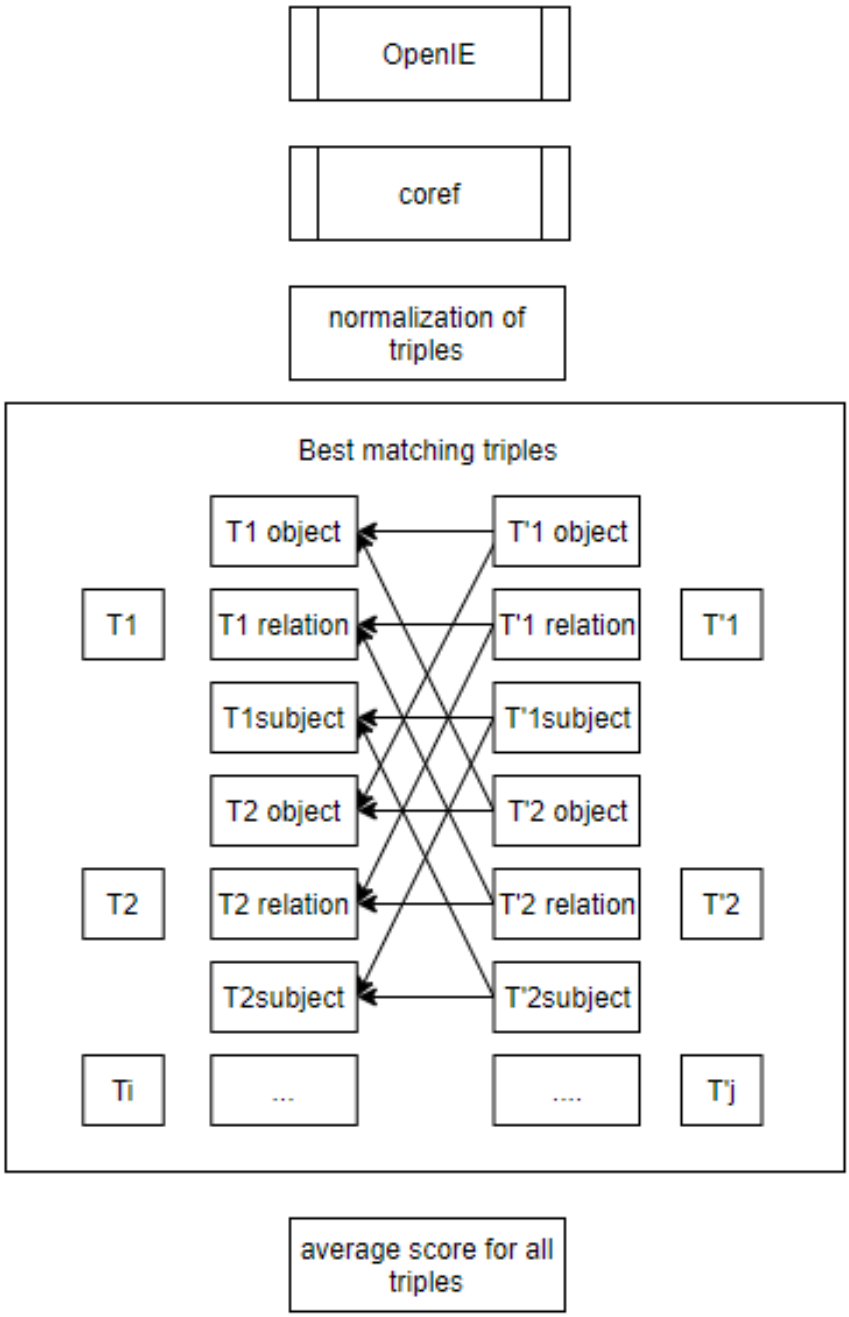


Рис. 2.1. Етапи обробки речень

Далі кожна трійка нормалізується шляхом видалення стоп-слів, які не мають самостійного смислового значення, таких як “are”, “over”, “these”, тощо. Також всі згадки про одну й ту саму сутність у тексті приводяться до одного виду, за допомогою функціоналу для знаходження тотожних сутностей coref, наприклад сутності “Engineer” та “he” можуть бути означеннями однієї особи в декількох реченнях.

Останнім кроком нормалізації є заміна суб’єкта на об’єкт: наприклад, трійка в еталонному реченні $‘invent’(‘Apple’, ‘Swift’)$ та трійка у відповіді

‘$create$’(‘$Swift$’, ‘$Apple$’) будуть зведені до однієї форми ‘$invent$’(‘$Apple$’, ‘$Swift$’) та ‘$create$’(‘$Apple$’, ‘$Swift$’) відповідно. Таким чином, для кожного речення s формується множина трійок $\{ t \mid t\ TripleRel(s) \}$, де: $t = (t_{subj},\ t_{rel},\ t_{obj})$, t_{subj} – суб’єкт, t_{rel} – відношення, t_{obj} – об’єкт.

$$SimTriple(t, \hat{t}) = \frac{1}{3}(SimWs(t_{subj}, \widehat{t_{subj}}, 1) + SimWs(t_{rel}, \widehat{t_{rel}},\ 1) +\ SimWs(t_{obj}, \widehat{t_{obj}}, 1) \quad (2.1)$$

де: $SimWs(ws_a, ws_b, i)$ – міра співпадіння ексклюзивних пар словосполучень.

$$SimWs(ws_a, ws_b, i) = (sum(Exclusive(ws_a, ws_b, SimWord)) \,/\, size(ws_a) + q(i) \quad (2.2)$$

де: $SimWord(w_a, w_b)$ – функція, що оцінює подібність слів згідно методу [7]; $q(i)$ – коригуючий додаток, який впливає на результат у разі словосполучень із різною кількістю слів та обернено пропорційний до параметра i.

Для суб’єкту t_{subj} та відношення t_{rel} параметр i дорівнює 1, тому трійки вважаються пов’язаними навіть при неповному співпадінні цих частин. Наприклад, для речення “*Continuous delivery is a software development practice*”, розбіжність між суб’єктами “*Continuous delivery*” та “*delivery*” незначно знижує відповідність всієї трійки еталонові ($i = 1$). Водночас у реченні “*Facebook is using continuous delivery*” аналогічна розбіжність у об’єкті “*Continuous delivery*” та “*delivery*” знижує відповідність трійки значно сильніше ($i = 5$) – тому слова, пропущені у цій частині відповіді, значно знижують відповідність всієї трійки еталонові.

$$Sim(T, T) =\ avg(Exclusive(t, \hat{t}, SimTriple)) \quad (2.3)$$

Для визначення міри подібності між двома реченнями розраховується середнє значення подібності ексклюзивних пар трійок еталону $\hat{t}$ та наданої відповіді t.

2.2 Обґрунтування методу

Даний метод був спеціально розроблений для роботи з відповідями на відкриті питання. Для отримання більш репрезентативного зображення однієї відповіді, на першому етапі проводиться нормалізація. Це необхідно, тому що даний метод, як правило, застосовують для великих текстів, а відкриті питання потребують два або три речення для відповіді. Саме тому весь метод зосереджений на оптимізації порівнянь невеликого об’єму текстів.

Також, на етапі порівняння, використовується параметр який вказує наскільки важливе співпадіння однієї частини трійки, тобто вказує на міру її значущості у відповіді кандидата. Експериментально було визначено що саме об’єкт в трійці являється головним показником, тому для нього був виставлений найбільший коефіцієнт, коли для суб’єкту та зв’язку найменший. Але якщо суб’єкт та зв’язок мають високу міру співпадінь то міра співпадіння об’єкту є критичною, і повинна перевищувати 50%. Дані коефіцієнти можливо

2.3 Експериментальне дослідження

Представлений метод реалізовано програмно (дивись Розділ 3), це дозволило оцінити практичну застосовність для вирішення проблеми відбору кваліфікованих кадрів, за допомогою оцінки відкритих питань.

Ефективність запропонованого методу семантичного порівняння речень оцінювалась в задачі автоматичного оцінювання відкритих текстових відповідей шляхом порівняння з еталоном.

У таблиці 2.1 наведені результати роботи обох методів у випадку, коли відповідь за значенням близька до еталону.

Таблиця 2.1

Результати визначення семантичної схожості для змістовно близьких речень

Еталонна відповідь	Надана відповідь	Метод семантичного порівняння вилучених пар	Метод жадібного сполучення
Class breaks the SRE principle because it handles business logic and creates new objects	Class violates SRE principle	0.74447	0.17963
He is usually taking a shower	He likes to take a shower in the evening	1	0.34714
User can create new commands	User can configure custom commands	0.57332	0.35610
Apple invented Swift	Swift created by Apple	0.74471	0.25
Mongo is more flexible than mysql	Mongo has more flexible scheme than mysql	0.543209	0.33538

У таблиці 2.2 подані відповідні результати для відповідей, що сильно відрізняються від еталона.

Таблиця 2.2

Результати визначення семантичної схожості для істотно відмінних речень

Еталонна відповідь	Надана відповідь	Метод семантичного порівняння вилучених пар	Метод жадібного сполучення
class uses inheritance.	class uses composition	0	0.34313

Продовження таблиці 2.2

class breaks the sre principle because it handles business logic and creates new objects.	class has a large interface, so it breaks interface segregation principle.	0.37710	0.3703
command works on Windows.	command works on Linux.	0	0.5
Java is the best	Java is the worst language ever	0	NaN
The procedure is performed in the second or third trimester	The child is used during the first trimester of university.	0.36352	0.12765

Тестова вибірка містила 210 пар еталон-відповідь, в яких більшість слів співпадає, але значення відповіді може бути як тотожним, так і протилежним до еталона. На цій вибірці результат роботи методу порівнювався із існуючим методом обчислення семантичної схожості через жадібне сполучення та семантику слів [8]. Відмінністю запропонованого методу є врахування зв’язків між словами у реченні замість попарного порівняння окремих слів. Аби метод мав інформативність для оцінювання відповідей, для змістовно близьких речень його результат має бути близьким до 1, а для істотно відмінних речень – до 0. Запропонований метод забезпечує середнє значення відповідності 0.72114 у першому випадку і 0.19217 – у другому. На тій же вибірці метод жадібного сполучення надає середні значення відповідно 0.29365 і 0.30655. Це дозволяє зробити висновок, що врахування відношень між словами згідно запропонованого методу надає перевагу при оцінюванні як завідомо правильних, так і неправильних відповідей у випадку статистично схожих речень, коли існуючий метод жадібного сполучення є неефективним.

Висновки до розділу

Був реалізований алгоритм методу для визначення семантичної подібності речень, які представляють собою відповіді на відкриті питання. Також були обґрунтовані реалізовані оптимізації для покращення визначення міри співпадінь. Ефективність запропонованого методу семантичного порівняння речень оцінювалась в задачі автоматичного оцінювання відкритих текстових відповідей шляхом порівняння з еталоном. Представлений метод був порівняний з існуючим на вибірці з 210 пар еталон-відповідь. За результатами були визначено, що розроблений метод являється більш ефективний та краще підходить для автоматизації оцінювання відповідей на відкриті питання для оцінки компетенцій при наймі співробітника.

Даний метод слугував основою доповіді для участі в міжнародній науково-практичній Інтернет конференції “Глобальні та регіональні проблеми інформатизації в суспільстві і природокористуванні 2021” з темою “Метод семантичного порівняння речень для систем оцінювання із відкритими питаннями” [9].

РОЗДІЛ 3
ПРОЕКТУВАННЯ СЕРВІСУ ТА ПЛАТФОРМИ ДЛЯ ПІДБОРУ ПЕРСОНАЛУ ІЗ АВТОМАТИЗОВАНИМ ОЦІНЮВАННЯМ КОМПЕТЕНЦІЙ

3.1 Загальні архітектурні рішення

Рекрутингова платформа складається з клієнтської та серверної частини. Інтерфейс повинен бути максимально спрощеним та ергономічним за для майбутнього редизайну та зменшення витрат часу.

Основна логіка розташована на серверній частині, за для можливості масштабування системи та реалізації майбутнього мобільного додатку. За основу для серверу була взята трирівнева архітектура клієнт-сервер яка показана на рис 3.1.

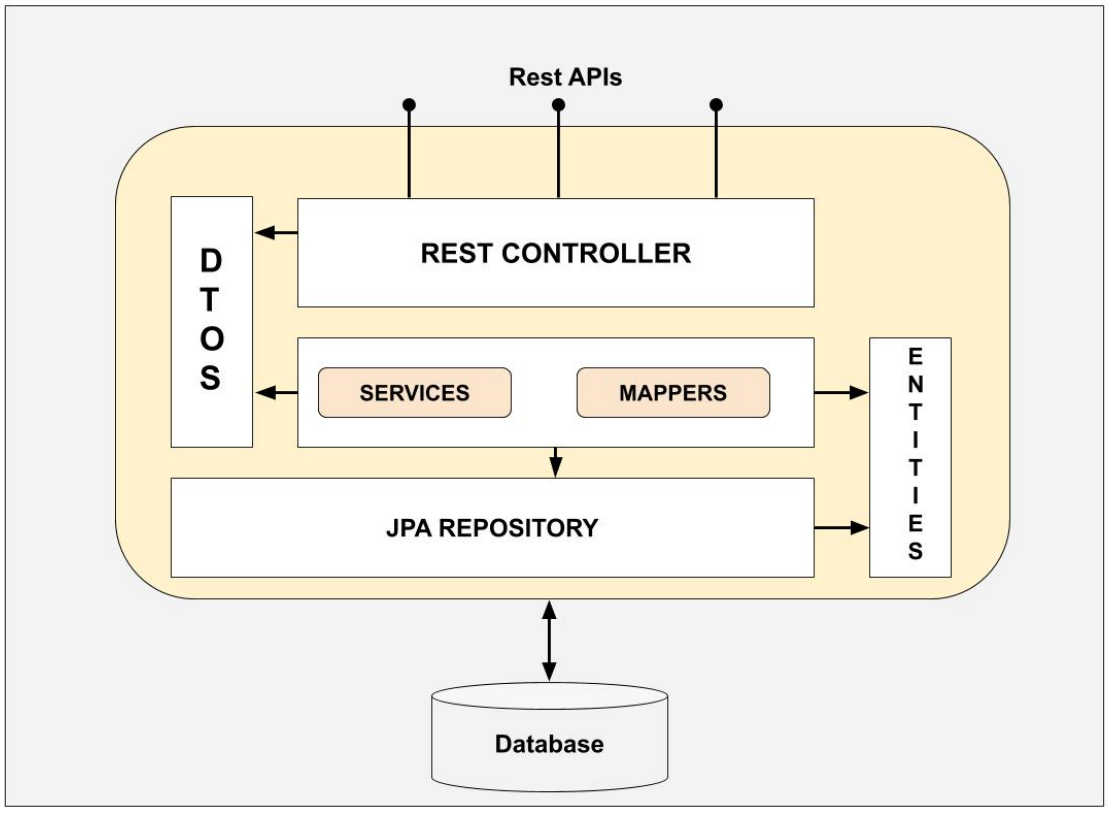


Рис. 3.1. Трирівнева архітектура клієнт-сервер

В даній архітектурі можна виділити три основних складові які взаємодіють між собою передаючи DTO об’єкти:

- Клієнтський або презентаційний рівень, відповідає за взаємодію з додатком, та оброблює вхідні запити. Для зручної взаємодії з додатком був представлений REST API реалізований за допомогою Controller фреймворку Spring.
- Наступний рівень відповідає за основну бізнес логіку додатку. В ньому знаходиться всі обчислення та допоміжні структури. Саме бізнес рівень взаємодіє з рівнем даних.
- Рівень даних повинен виконувати просту функцію взаємодії за базою даних. Тобто цей рівень не повинен знати як ця інформація буде використана або навіщо, його задача якнайліпше опрацювати запит до сховища даних.

Дана архітектура була вибрана через її гнучку структуру та чіткі рамки модулів. Це призводить до деякої складності в реалізації при зберіганні незалежності рівнів, але дана проблема може нехтуватись при певному рівні кваліфікованості розробника.

Платформа спроектована як 4 незалежні модуля які поділяються на менші підмодулі:

- модуль прескринінгу;
- модуль управління процесом найму;
- модуль автентифікації та авторизації;
- модуль рекрутера;
    1. підмодуль управління профілем;
    2. підмодуль управління компанією та командою;
    3. підмодуль таймслотів робочого графіку
    4. підмодуль налаштування вакансії;
- модуль шукача;
    1. підмодуль управління профілем;
    2. підмодуль збережених вакансій.

3.2 Вибір технологій та засобів реалізації для платформи

Головними критеріями для вибору фреймворків та технологій для платформи були:

- швидкість розробки, яка включає в себе обширну документацію, легкість в розробці, та інше;
- релевантність технології, для отримання практичних навичок які знадобляться в реальних задачах.

3.2.1 Бібліотеки для семантичного аналізу

Для модулю прескринінгу платформи використовується метод семантичного порівняння текстів на основі виділення пов’язаних трійок. Цей метод реалізовано у вигляді пайплайну CoreNLP. Stanford CoreNLP надає набір інструментів аналізу природної мови, які можуть дати основні форми слів, їх частини мови, нормалізувати дати, час та числові величини та позначити структуру речень з точки зору фраз та залежностей від слів [10]. Також вона дає можливість знайти які саме фрази стосуються одних і тих самих завдань. В ньому пошук семантично поєднаних трійок виконується компонентом OpenIE із використанням стандартних моделей для англійської мови. Нормалізація виконується за допомогою компоненту coref, який виділяє згадки про одну й ту саму суть у тексті. Це дозволяє представити підмет та присудок в одному вигляді при декількох згадках у реченні.

Для оцінки семантичної подібності слів використовувався індекс подібності, наданий бібліотекою векторизації DISCO [11]. Використання DISCO потребує попередньо розрахованої бази даних (простору слів), що містить векторне представлення кожного слова. В роботі використано простір англійських слів enwili13slm [12].

3.2.2 Засоби реалізації серверної частини

Вся серверна частина платформи написана об’єктно-орієнтованою мовою програмування Java. Цей вибір обумовлений тим, що платформа Java має велику кількість технологій та фреймворків, створених для вирішення різних прикладних проблем. Ще однією перевагою Java є те, що вона забезпечує кросплатформну віртуальну машину (JVM).

Окрім кросплатформенності, перевага цієї мови полягає у високій надійності її роботи, оскільки вона була розроблена як об’єктно-орієнтована мова високого рівня з жорсткою типізацією. З мови було виключено множинне наслідування для запобігання неоднозначності. Натомість було введене поняття інтерфейсу, тобто контрактом класу, який його реалізує від нього. Ще одним фактором високої надійності цієї мови програмування є система винятків та обробки помилок, які поділяються на два типи:

- помилки які можуть відбутися по вині самого програміста наприклад: переповнення пам’яті через рекурсію або вихід за межі масива;
- обов’язкові для обробки виняткової ситуації, що можуть виникнути під час роботи програми, наприклад: необхідно відкрити файл якого не існує або програміст хоче зупинити наявний потік.

Java-програмісту не потрібно стежити за розподіленням пам'яті, так як збирач сміття управляє пам'яттю автоматично. Збирач сміття запускається віртуальною машиною Java. Збирач сміття – це фоновий процес, який запускається з певною періодичністю і звільняє пам'ять використану об'єктами, які більше не потрібні. Різні віртуальні машини мають відмінні один від одного алгоритми збору сміття.

Програмний продукт написаний на базі Spring Framework який забезпечує комплексну модель розробки і конфігурації для сучасних бізнес-додатків. Ключовий елемент Spring – підтримка інфраструктури на рівні програми, тому розробники можуть зосередитися на бізнес-логіці без зайвих налаштувань в залежності від

середовища виконання. Але при розробці більш складних програмних продуктів, без розуміння процесу роботи фреймворку, додати або кастомізувати функціонал дуже складно.

Spring Framework скоріше загальна назва для цілого ряду невеликих фреймворків, кожен з яких виконує якусь свою роботу. На рис 3.2 абстрактно зображені модулі цього фреймворку.

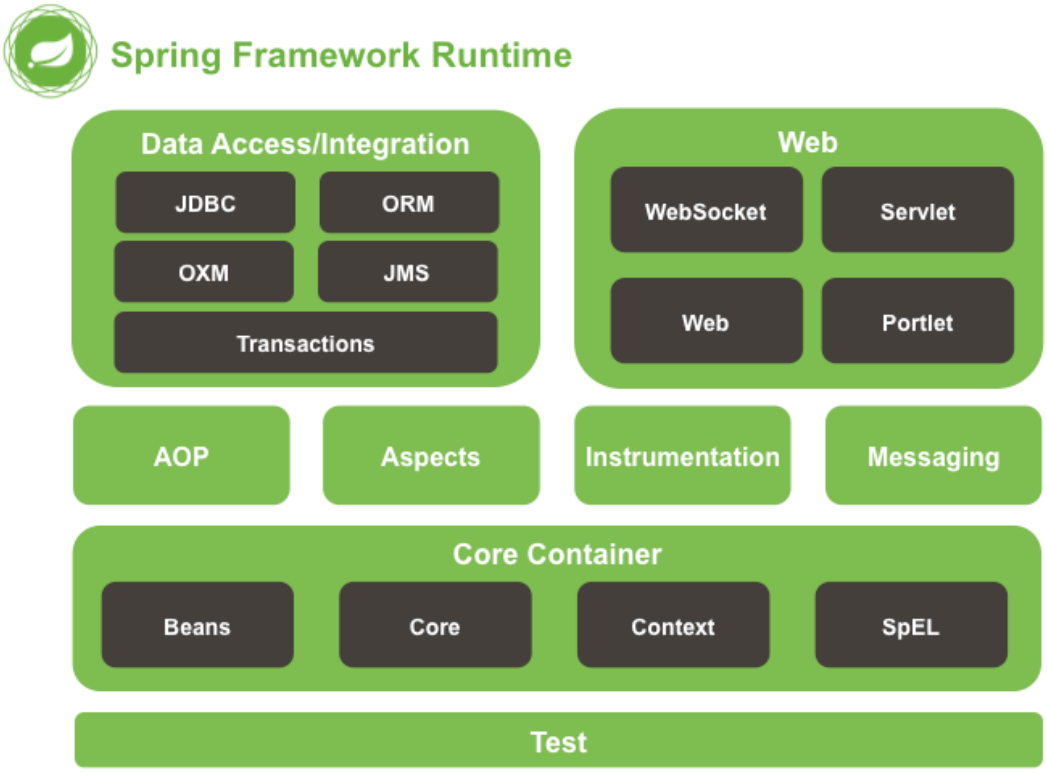


Рис. 3.2. Модулі Spring Framework

Програмний фреймворк – це комплекс готових до використання програмних рішень, що включають в себе архітектуру побудови проекту, логіку та базову функціональність системи або підсистеми. Фреймворк містить в собі набір бібліотек, що пропонують готовий набір рішень, також може містити допоміжні програми, скрипти та загалом все те, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Одна з головних переваг використання фреймворків – це стандартизованість структури застосування.

Головні переваги Spring Framework:

- Підтримує різні способи конфігурації.
- Spring Framework може бути задіяний на всіх архітектурних шарах, що застосовуються при розробці web-додатків.
- Позбавляє від самостійного створення фабричних класів.
- Підтримує декларативне програмування.
- Використовує модель POJO при написанні класів.
- Дозволяє вільно пов'язувати модулі і легко їх тестувати.

Незважаючи на наявність стількох переваг, якими володіє Spring, тривала процедура підготовки, пов'язана з його налаштуванням, сприяла появі Spring Boot, який дозволяє:

- уникнути розгортання war файлів;
- використовувати Tomcat, Jetty або Undertow безпосередньо вибудуваний в додаток;
- зменшити обсягу вихідного коду;
- мати багату додаткову функціональність «з коробки»;
- спростити запуск;
- спростити запуск та управління програмою завдяки функції автоконфігурування, яка позбавляє від написання зайвого коду і непотрібного налаштування.

Тому в сучасній розробці Java асоціюється з Spring та всі додатки, як правило, використовують Spring Boot.

Spring Data – пов'язує велику кількість різних підпроектів, які прагнуть забезпечити послідовний інтерфейс програмування від SQL та Postgres, до баз даних типу NoSQL, таких як MongoDB, Cassandra та Redis. Він також підтримує такі технології, як Hibernate, JDBC, LDAP, KeyValue, Geode та GemFire. Модулі ком’юніті додають підтримку для багатьох інших проектів, таких як Couchbase, DynamoDB, Elasticsearch, Neo4j та Apache Solr. Головна ідея полягає в тому, щоб поєднати всі ці

технології разом із послідовним і звичним для розробників Spring інтерфейсом, який полегшує впровадження та підтримку програм, що використовують ці технології.

Фреймворк може надавати ряд важливих функцій які дозволяють розробнику швидко і легко оголосити об'єкти даних (ентіті), використовуючи прості анотовані класи Java. Spring Data автоматично відобразить анотований клас Java до відповідної структури таблиці. Він також забезпечує інтерфейс класу сховища, який управляє читанням та записом у базу даних, використовуючи інтуїтивно зрозумілу структуру методу, схожу на запит. Spring Data чудово працює з Spring Boot, забезпечуючи надзвичайно ефективний спосіб створення веб-додатків без необхідності кодування великої кількості конфігурації та типового коду. Використовуючи Spring Boot та Spring MVC, веб-служби RESTful можна створити за лічені хвилини лише з кількома файлами та кількома десятками рядків коду. В ідеалі один і той же код моделі даних та код бізнес-логіки можна використовувати з різними базами даних, і що перемикання між ними вимагатиме лише зміни деяких залежностей та реалізованого інтерфейсу сховища. Насправді для цього потрібно багато абстракції. Як правило, вибирають SQL або NoSql через особливості, характерні для цих типів баз даних. Spring Data забезпечує баланс між спрощенням та абстракцією, зберігаючи особливості бази даних та гнучке налаштування.

Для проекту був вибраний Spring Data JPA який базується на Hibernate – це ORM фреймворк для Java з відкритим вихідним кодом. ORM – об'єктно-реляційне відображення об'єктно-орієнтованої моделі даних в реляційні базі даних. Він створює зв'язок між таблицями бази даних і Java-класами. Це звільняє розробників від величезної кількості рутинної роботи, в якій дуже легко допустити помилку і дуже важко потім її знайти. Він підтримує зв'язок класів з таблицями бази даних і типів даних мови програмування із типами даних SQL, та надає засоби автоматичної побудови SQL запитів для зчитування або запису даних. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних. Тому Hiberante значно зменшує час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Технологія Hibernate забезпечує використання

SQL-подібної мови Hibernate Query Language, яка дозволяє виконувати SQL-подібні запити, записані поряд з об'єктами даних Hibernate.

Рекрутер та користувач повинні мати можливість реєструватись в системі та мати доступ до своїх ресурсів. Для цього використовується Spring Security – це фреймворк, який зосереджений на забезпеченні механізмів автентифікації та авторизації додатків Spring. На додаток до автентифікації та авторизації, Spring Security можна налаштувати для захисту додатка від багатьох поширених атак.

Механізм взаємодії був реалізований з використанням JWT або JSON Web Tokens – це стандарт, який в основному використовується для захисту API REST. Після цього кроку клієнт повинен надати переданий сервером токен у заголовку авторизації. Сервер перевірить дійсність цих даних та дозволить або відхилить запит. Токен може також зберігати ролі користувачів і авторизувати запити на основі даних повноважень. Рядок токену складається з трьох блоків, між якими ставлять крапку: заголовок, набір полів (claims) і підпис. Структура токену показана на рис 3.3.

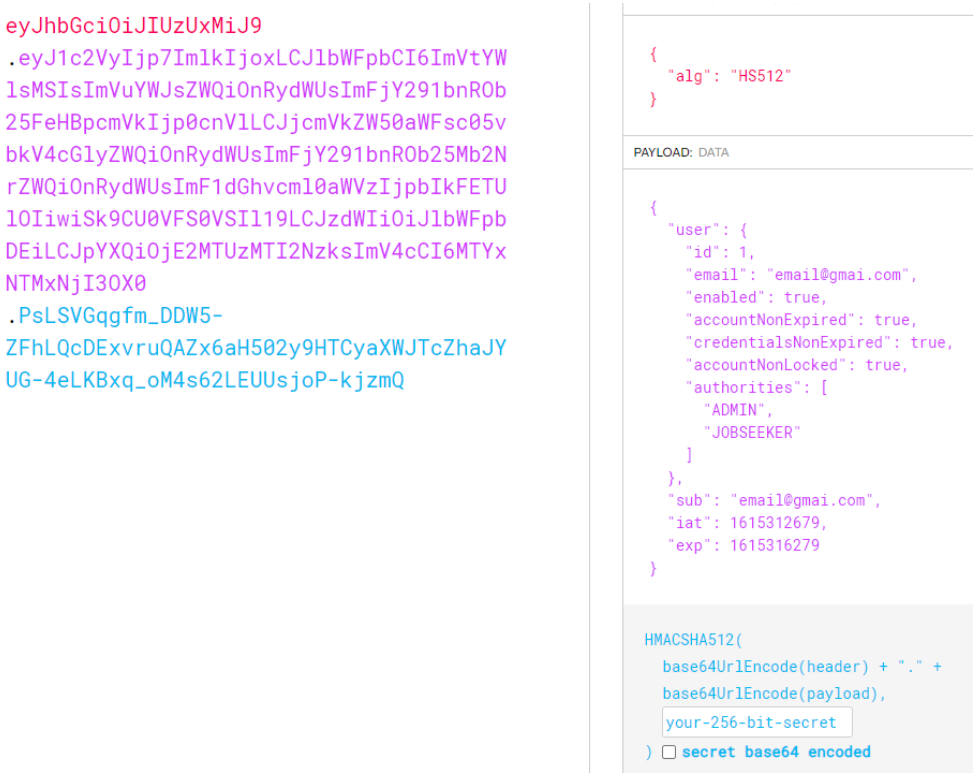


Рис. 3.3. Структура токену JWT

Перші два блоки представлені в JSON форматі і додатково закодовані в формат base64. Набір полів містить довільні пари ім'я-значення, також стандарт JWT визначає кілька зарезервованих імен (iss, aud, exp та інші). Підпис може генеруватися за допомогою симетричних алгоритмів шифрування так і асиметричних. Дані може прочитати кожен, але їх валідність перевіряється за підписом на сервері за допомогою ключа.

3.2.3 Засоби реалізації клієнтської частини

Фронт-енд частина виконана з використанням JavaScript – це інтерпретована мова програмування з об’єктно-орієнтованими можливостями. Мова є однією з трьох основних технологій у веб-розробці з HTML, що описує вміст, CSS – як відображається вміст. Сам по собі, JavaScript може працювати на всіх сучасних браузерах без будь-яких додаткових плагінів або компіляторів і використовується на більшості сучасних веб-сайтів.

Основні особливості JavaScript:

- мова не має підтримки строгої типізації. Ця властивість звільняє розробників від необхідності формувати складні ієрархії класів та зменшують складність всієї системи. Але також ускладнює процес розробки в великих проектах, де потрібно чітко розуміти структуру та типи даних які використовуються. Тому для розширення функціоналу використовують TypeScript який базується на JavaScript але є типізованим;
- функції в JavaScript це об’єкти першого класу, тобто функції можна використовувати як і звичайний тип даних. Це дозволяє застосовувати підходи властиві функціональним мовам програмування.

React – найпопулярніша інтерфейсна бібліотека JavaScript у галузі веб-розробки. Її використовують великі компанії та «новоспечені стартами». Він

створений для побудови швидких та інтерактивних користувальницьких інтерфейсів для веб та мобільних додатків. Також, це відкрита компонентна та інтерфейсна бібліотека, відповідальна лише за рівень перегляду програми. У архітектурі Model View Controller (MVC) рівень перегляду відповідає за те, як виглядає та реагує на зміни програма. Популярність технології дуже швидко збільшується завдяки [13]:

- простоті створення динамічних додатків: React спрощує створення динамічних веб-додатків, оскільки вимагає менше кодування та пропонує більше функціональних можливостей, на відміну від JavaScript, де кодування часто дуже швидко ускладняється;
- використання для розробки веб і мобільних додатків: Існує фреймворк, який називається React Native, похідний від самого React, який надзвичайно популярний і використовується для створення прекрасних мобільних додатків. Отже, насправді React можна використовувати для створення як веб, так і мобільних додатків;
- використання багаторазових компонентів: компоненти є будівельними блоками будь-якої програми React, і одна програма, як правило, складається з декількох компонентів. Ці компоненти мають свою логіку та елементи керування, і їх можна використовувати повторно протягом усієї програми, що, у свою чергу, різко скорочує час розробки програми;
- односпрямованому потоку даних: React слідує за односпрямованим потоком даних. Це означає, що при розробці програми React розробники часто вкладають дочірні компоненти в батьківські компоненти. Оскільки дані рухаються в одному напрямку, стає простіше налагоджувати помилки та знати, де проблема виникає у програмі на даний момент;
- дуже низький поріг входу: React легко засвоїти, оскільки він здебільшого поєднує основні концепції HTML та JavaScript з деякими корисними доповненнями;
- спеціальним інструментам для легкого налагодження: Facebook випустив розширення Chrome, яке можна використовувати для налагодження програм

React. Це робить процес налагодження веб-програм React швидшим та простішим.

– покращеної продуктивності: React використовує віртуальний DOM, тим самим швидше створюючи веб-програми. Віртуальний DOM порівнює попередні стани компонентів та оновлює лише ті елементи в Real DOM, які були змінені, замість того, щоб знову оновлювати всі компоненти, як це роблять звичайні веб-програми. Цей процес схематично показаний на рис 3.4. на якому змінюється частина дерева.

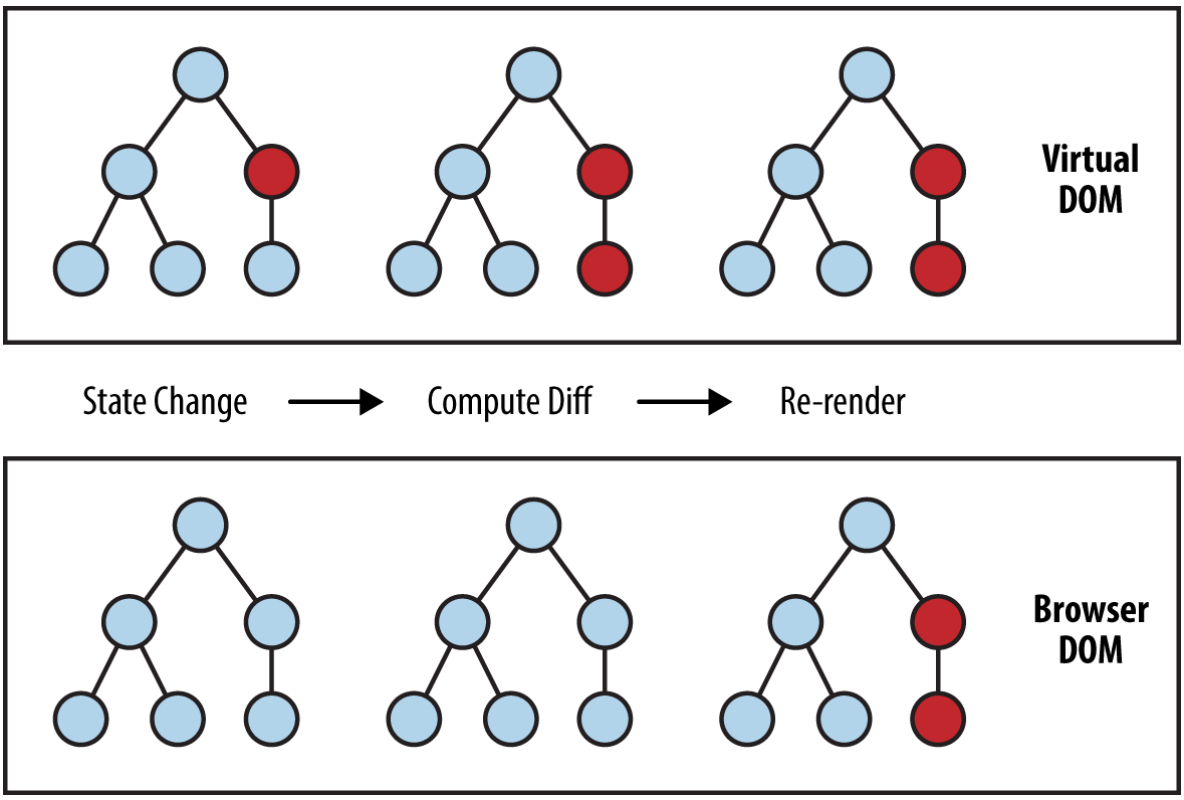


Рис. 3.4. Синхронізація віртуального DOM з реальним

Також істотною різницею від інших фреймворків є використання шаблонного синтаксису JSX – це розширений синтаксис JavaScript, який дозволяє використовувати схожий на HTML синтаксис для опису структури інтерфейсу.

3.3 Процес рекрутингу на платформі

Процес рекрутингу пошуку кваліфікованих кандидатів є ключовим в платформі. Даний процес показаний в UML діаграмі процесу на рис. 3.5.

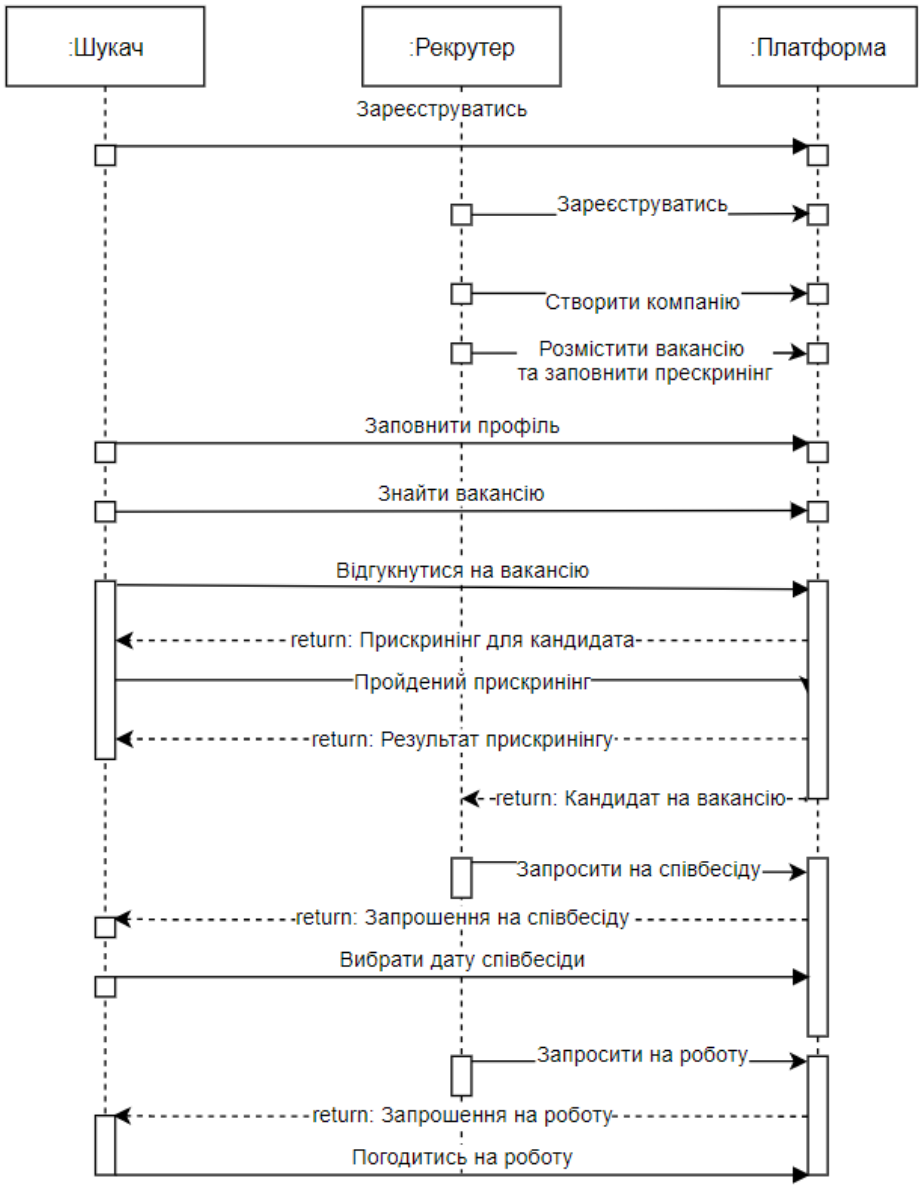


Рис. 3.5. UML діаграма процесу взаємодії через платформу

Потрібно зазначити, що користувач та рекрутер не взаємодіють безпосередньо між собою. Всі етапи контролює платформа, для пришвидшення процесу найму.

3.4 Проектування модулів

Кожен модуль був спроектований окремо для зменшення залежностей модулів. Функціонал модулів використовується у відповідних частинах клієнтської частини, дизайн якої був розроблений першим етапом.

3.4.1 Модуль прескринінгу

Платформа передбачає можливість створювати спеціальні автоматичні тести для відсіювання некваліфікованих кандидатів. Для цього був створений модуль прескринінгу який дозволяє створювати прості тести з декількома відповідями та більш специфічні – відкриті питання які потрібно проаналізувати. Дизайн налаштування автоматизації показаний на рис. 3.6 та рис. 3.7.

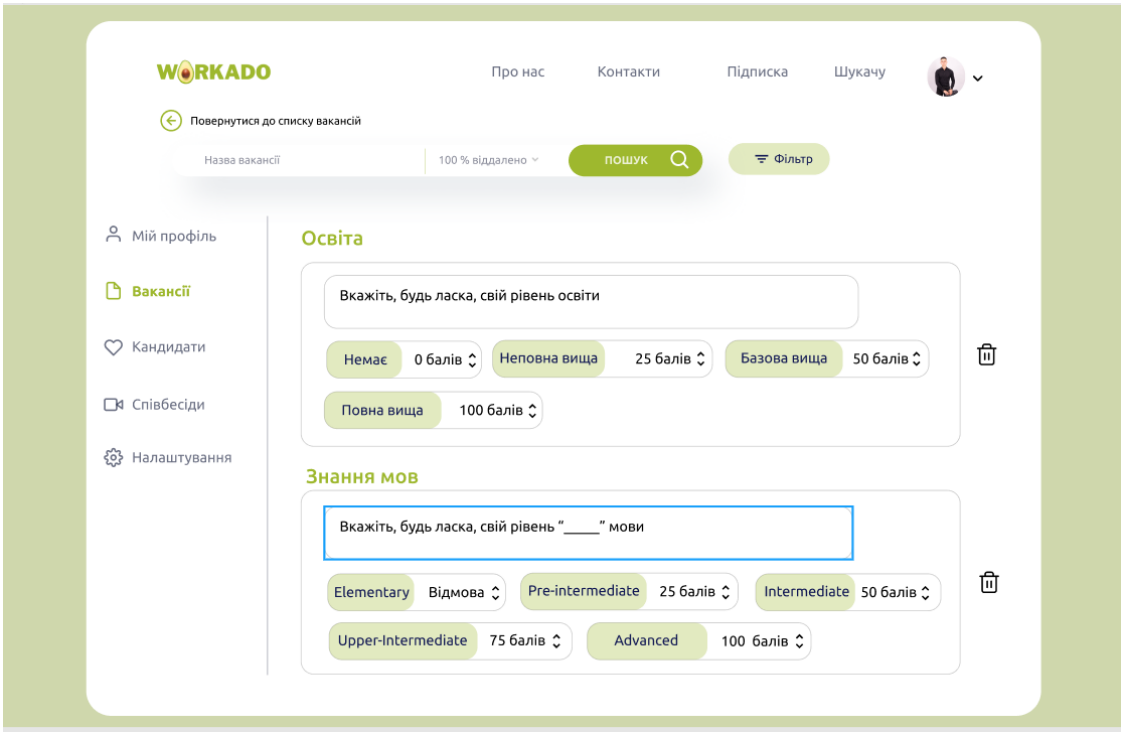


Рис. 3.6. Дизайн налаштування основних питань прескринінгу

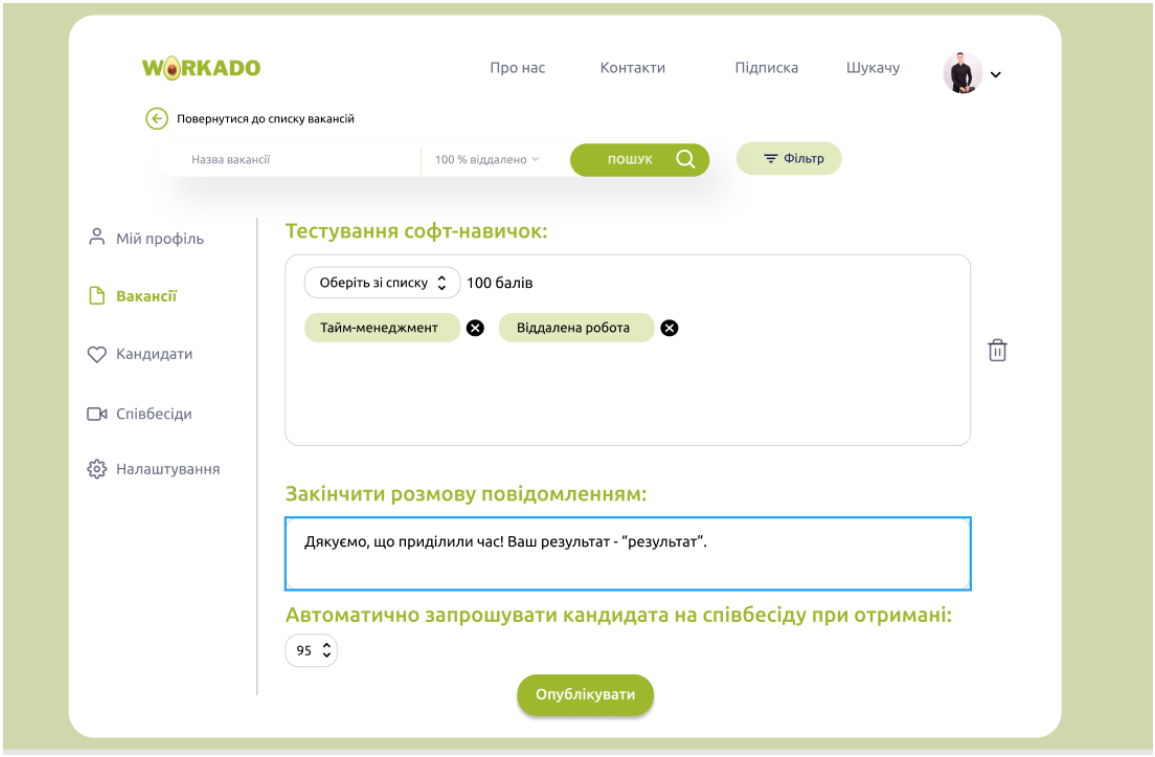


Рис. 3.7. Дизайн налаштування тестових питань прескринінгу

Цей модуль було розбито на дві частини які відповідають за питання до користувача і його відповіді. Схема класів зображена на рис. 3.8.

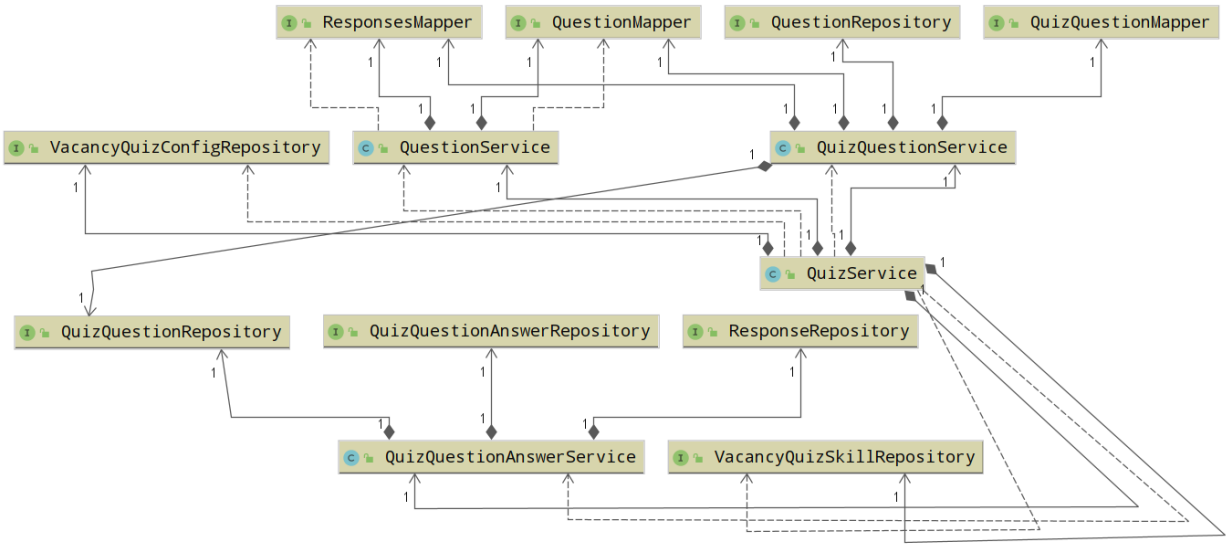


Рис. 3.8. Схема класів модулю прескринінгу

3.4.2 Модуль управління процесом найму

Дизайн сторінки рекрутера для управління етапами користувача знаходиться на рис 3.9.

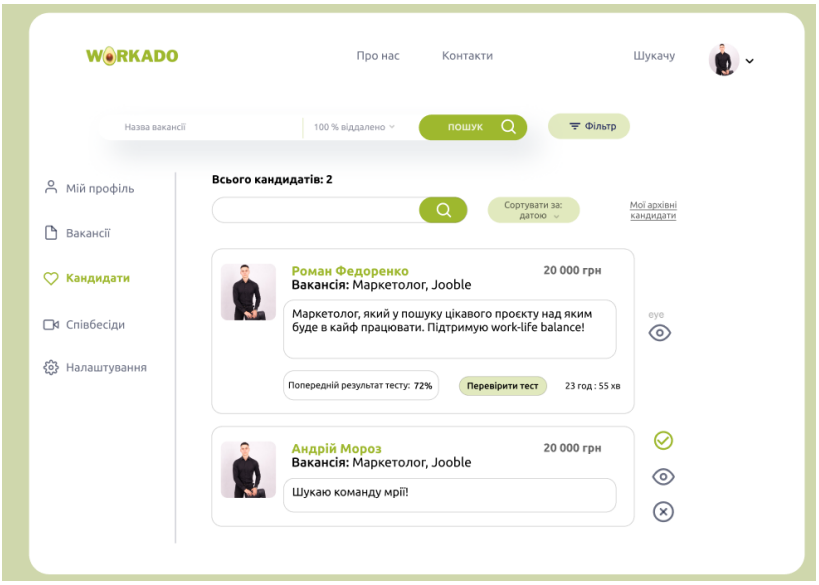


Рис. 3.9. Дизайн сторінки рекрутера для управління процесом найму

Дизайн для користувача для управління етапами рекрутингу на рис. 3.10. Сторінка надає коротку статистику про кожний етап відбору. Також картки на сторінці відсортовані за їх релевантністю.

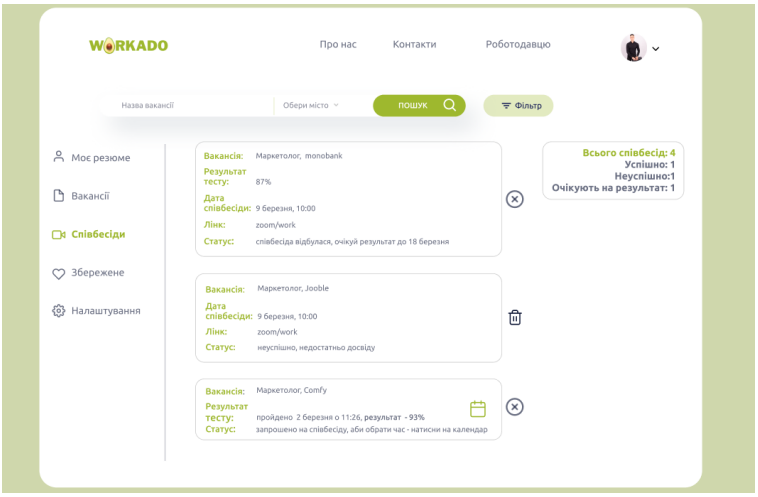


Рис. 3.10. Дизайн сторінки шукача для управління процесом найму

Схема класів модулю показана на рис. 3.11.

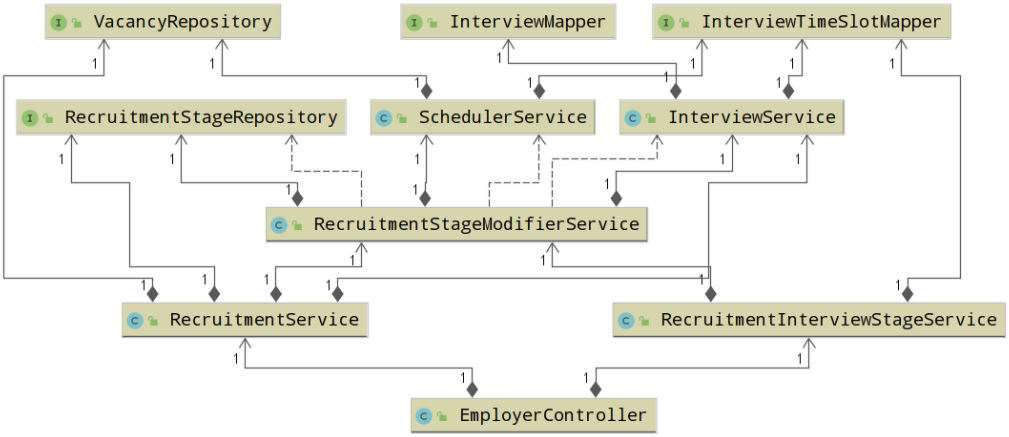


Рис. 3.11. Схема класів модулю управління процесом найму

Процес рекрутингових етапів розділений на два класи: RecruitmentService та RecruitmentInterviewStageService. Ці класи використовують RecruitmentStageModifierService який виступає в ролі стейт-машини, тобто змінює етап рекрутингу залежно від дій користувача викликаючі всі необхідні залежні компоненти.

3.5.3 Модуль рекрутера

В системі модуль рекрутера являється найбільшим, та був відділений від модулю шукача. Це дозволило зібрати всю складну бізнес та домену логіку в одному місці, що дозволило розділити всю систему на частини з більш чіткими рамками.

Рекрутер повинен мати можливість персоналізувати свою сторінку для більш теплого контакту з кандидатами. Сторінка повинна бути мінімалістична та не перенасичена інформацією. Дизайн сторінки приведений на рис. 3.12

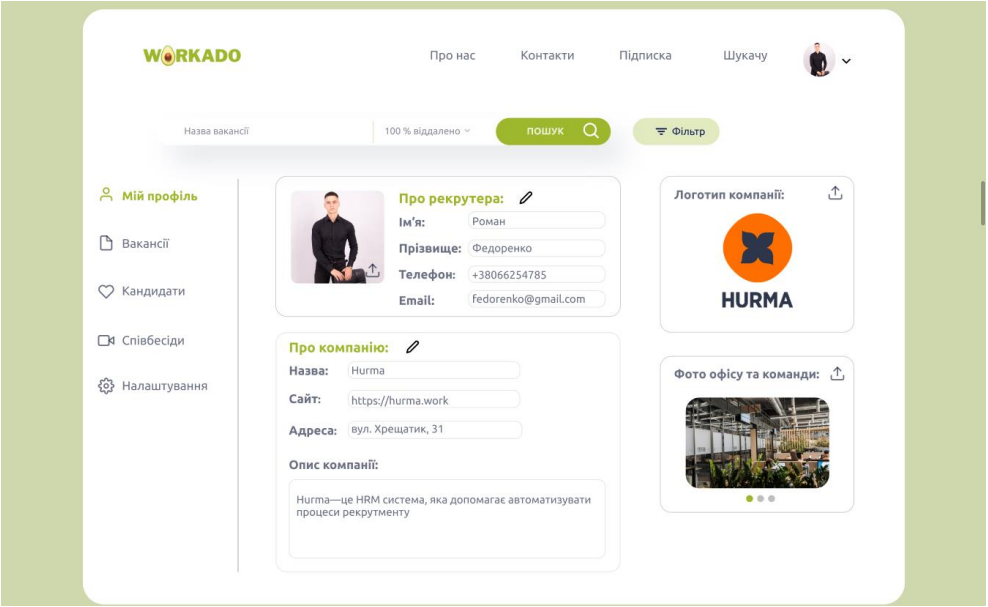


Рис. 3.12. Дизайн сторінки профілю рекрутера

Основні класи підмодулю представлені на рис 3.13.

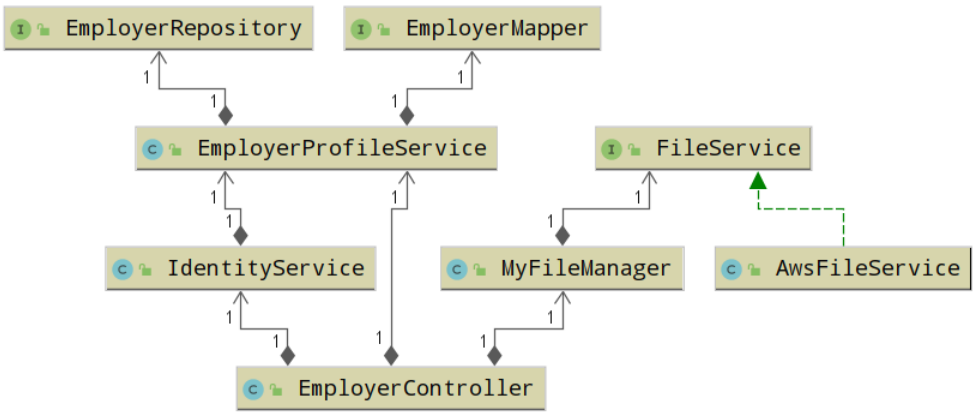


Рис. 3.13. Схема класів підмодулю профілю рекрутера

Головним класом являється EmployerProfileService який абстрагує в собі роботу з профілем. За допомогою нього можна створити або оновити інформацію профіля.

Для створення вакансії, ректрутеру необхідна компанія на яку він працює та налаштовані таймслоти. В компанії може бути більше одного рекрутера та їх менеджер, тому було вирішено надати можливість доєднатись до команд. Рекрутер може створити свою компанію, заповнивши відповідні дані на сторінці профілю, або доєднатись до вже існуючої за допомогою спеціального запрошення, яке потрібно

згенерувати одному з учасників компанії. Вигляд сторінки роботи з командою можна побачити на рис. 3.14.

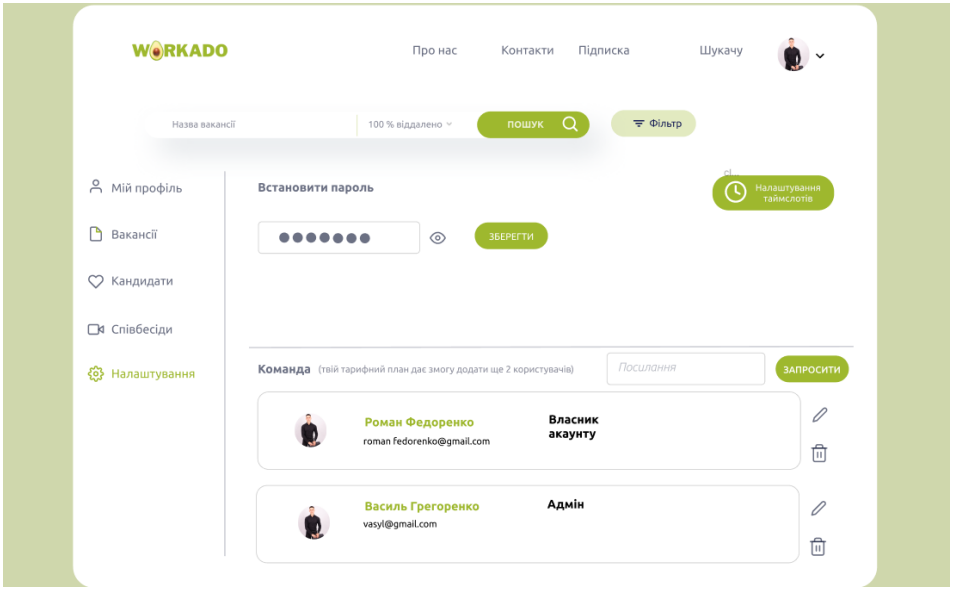


Рис. 3.14. Дизайн сторінки для взаємодії з командою

Схема класів цього модулю показана на рис. 3.15. Головними класами є EmployerCompanyService – для взаємодії з компанією та CompanyTeamService – для взаємодії з командою. Вони були спеціально спроектовані як незалежні класи, за для можливого розширення системи новими типами спеціалістів. Наприклад при додаванні типу менеджеру, якщо для нього необхідні унікальні можливості, необхідно буде створити нову вітки взаємодії, але модуль команди так і залишиться незмінним.

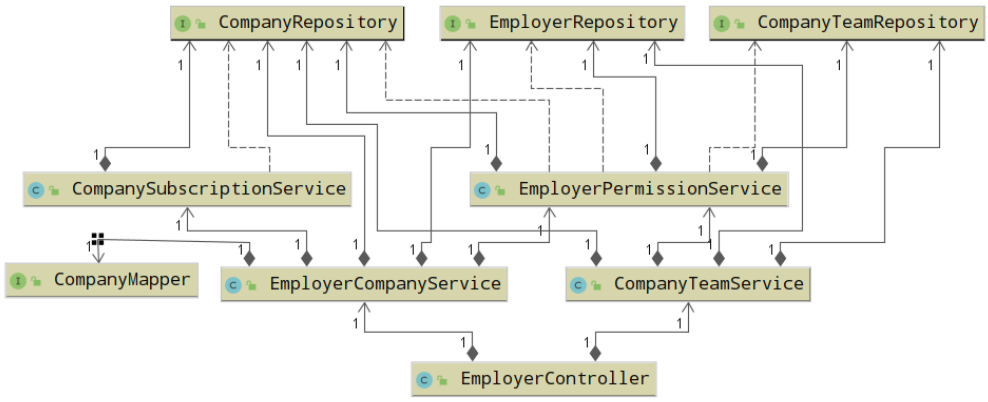


Рис. 3.15. Схема класів модулю управління компанією та командою

Схема таблиць пов’язаних з даним модулем показана на рис. 3.16.

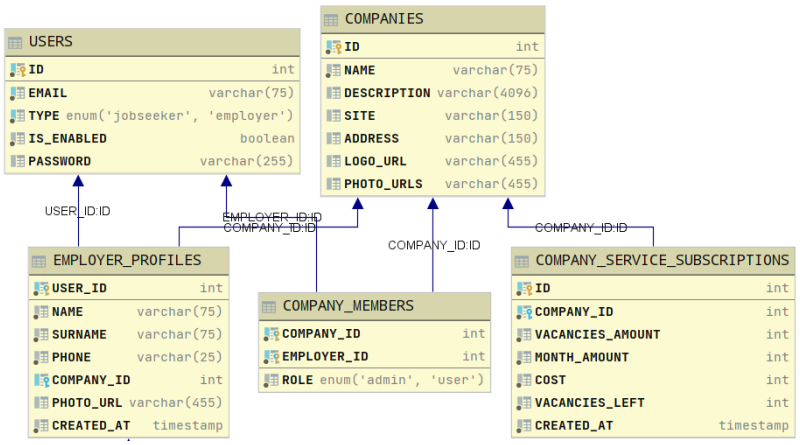


Рис. 3.16. Схема таблиць модулю управління компанією та командою

Через можливість майбутнього масштабування, схема передбачає розширення новими типами кандидатів. Тому таблиця company_members зв’язана з таблицею users, а не employer_profiles.

Однією з додаткових функцій платформи, є можливість розміщувати на ній вакансії, та слугувати ще одним каналом пошуку для рекрутерів. Дизайн сторінки показаний на рис. 3.17.

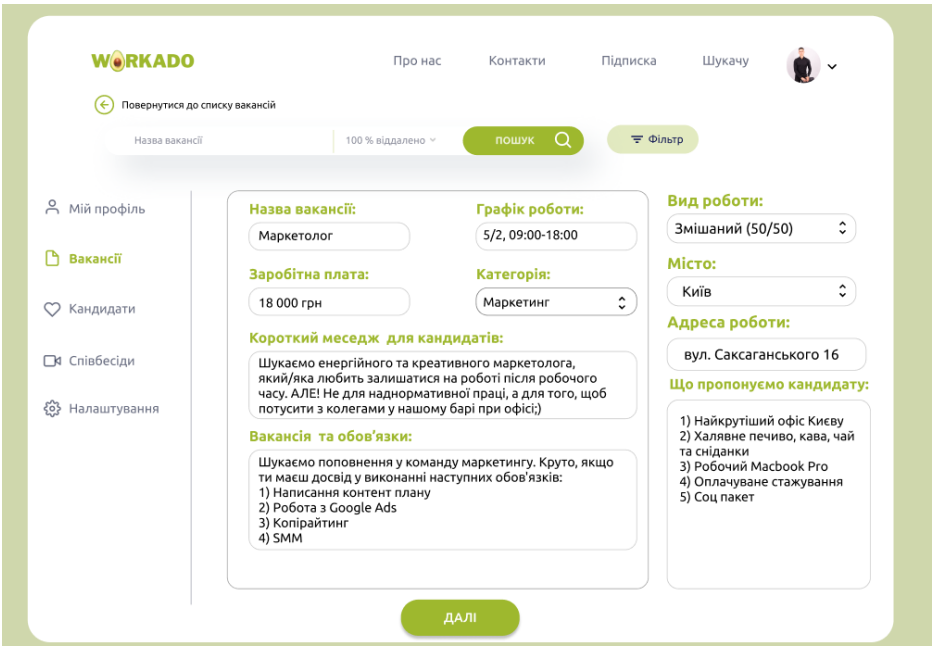


Рис. 3.17. Дизайн сторінки для налаштування вакансії

Схема взаємодії класів показана на рис. 3.18. Головним класом виступає EmployerVacancyService. Цей клас дозволяє створити рекрутеру вакансію, перевіривши наявність у рекрутера компанії, налаштованих таймслотів та валідність підписки компанії.

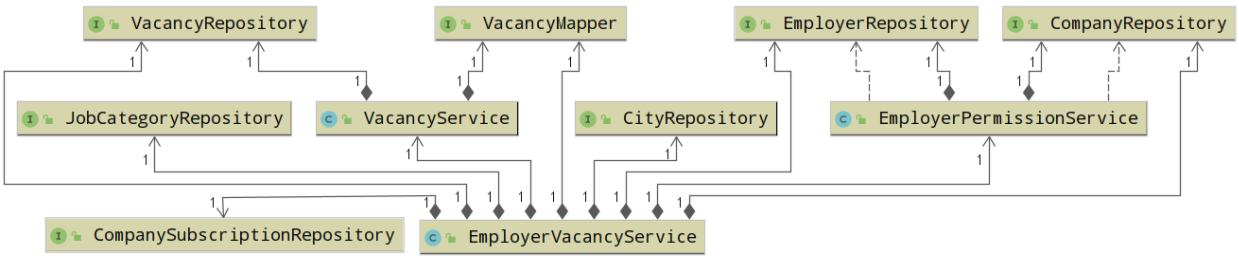


Рис. 3.18. Схема класів підмодулю налаштування вакансії

Схема таблиць модулю показана на рис. 3.19.

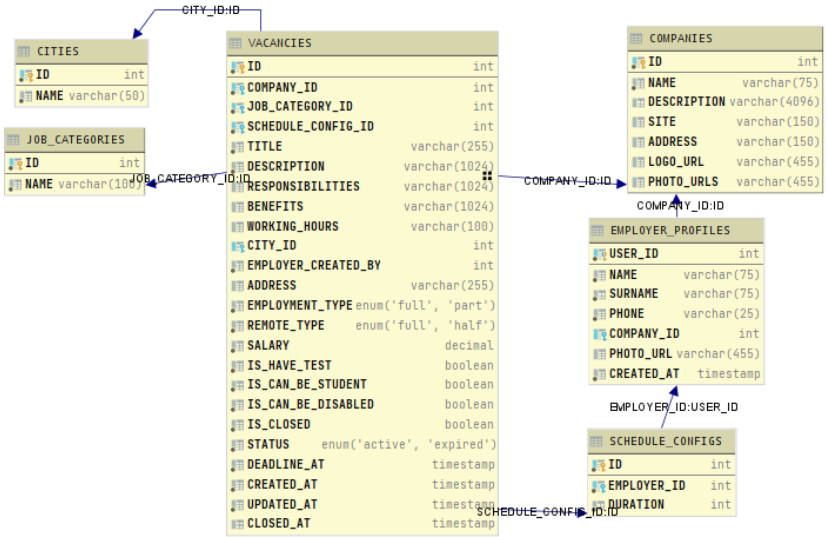


Рис. 3.19. Схема таблиць підмодулю налаштування вакансії

На схемі показана головна таблиця Vacancies, від якої відходять допоміжні таблиці Cities, Job_categories, Companies. Дана схема не є ідеальною, тому що типи полів в ній можна розбити на більш гранулярні, але, через те що, модуль вирішує не головну проблему всієї платформи, було вирішено обрати більш спрощену версію.

Платформою надана можливість конфігурувати робочі години рекрутера. За якими кандидат зможе вибрати для себе найліпший час для етапу інтерв'ю. Це був вимушений крок через відказ від створення чату, але весь процес тепер більше автоматизований, та дозволяє зменшити потік інформації яку ректрутеру потрібно тримати про кожного кандидата. Дизайн сторінки налаштування таймслотів показаний на рис 3.20.

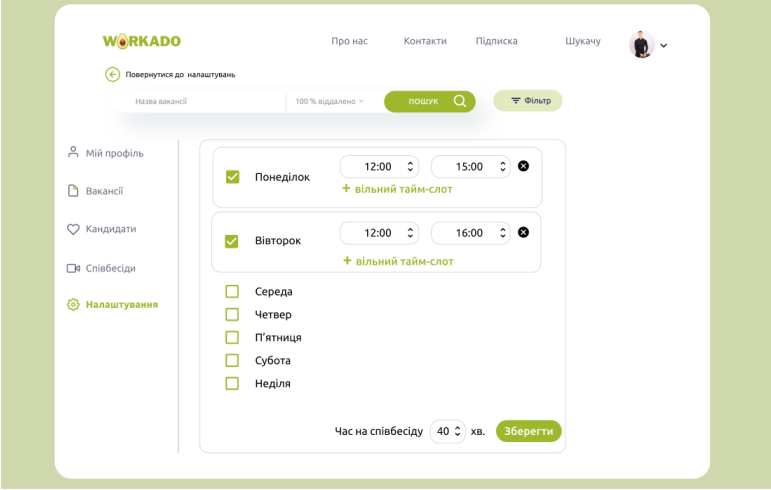


Рис. 3. 20. Дизайн сторінки таймслотів робочого графіку

Схема класів причасних до даного модуля вказана на рис. 3.21. Клас SchedulerConfigurationService виконує основні функції для роботи з конфігурацією рекрутера. При створені або оновлені конфігурації таймслотів, проводиться валідація класом ScheduleValidator, який перевіряє чи всі таймслоти не перетинаються та іншу валідність даних. Потім кожен проміжок створює новий або перезаписує існуючий в базі даних.

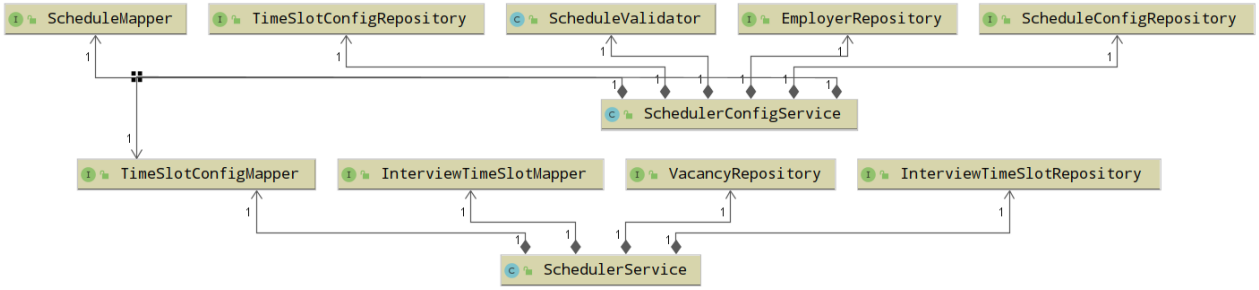


Рис. 3.21. Схема класів таймслотів робочого графіку

Для отримання інформації основаної на конфігурації таймслотів рекрутера, використовується клас ScheduleService, за допомогою якого, шукач може побачити всі вільні таймслоти для проходження інтерв'ю.

3.5.4 Модуль шукача

Для платформи необхідно було створити спеціальний модуль шукача для можливості відслідковувати активність кожного з них. Цей модуль являється більше допоміжним і слугує як додатковий функціонал, тому що в подальшому, пройти прескринінг зможе кандидат без профілю на платформі.

Шукач може надати інформацію необхідну рекрутеру у формі резюме. Також відслідковується наповненість профілю, за для заохочення надавати більше інформації. Дизайн можна побачити на рис. 3.22.

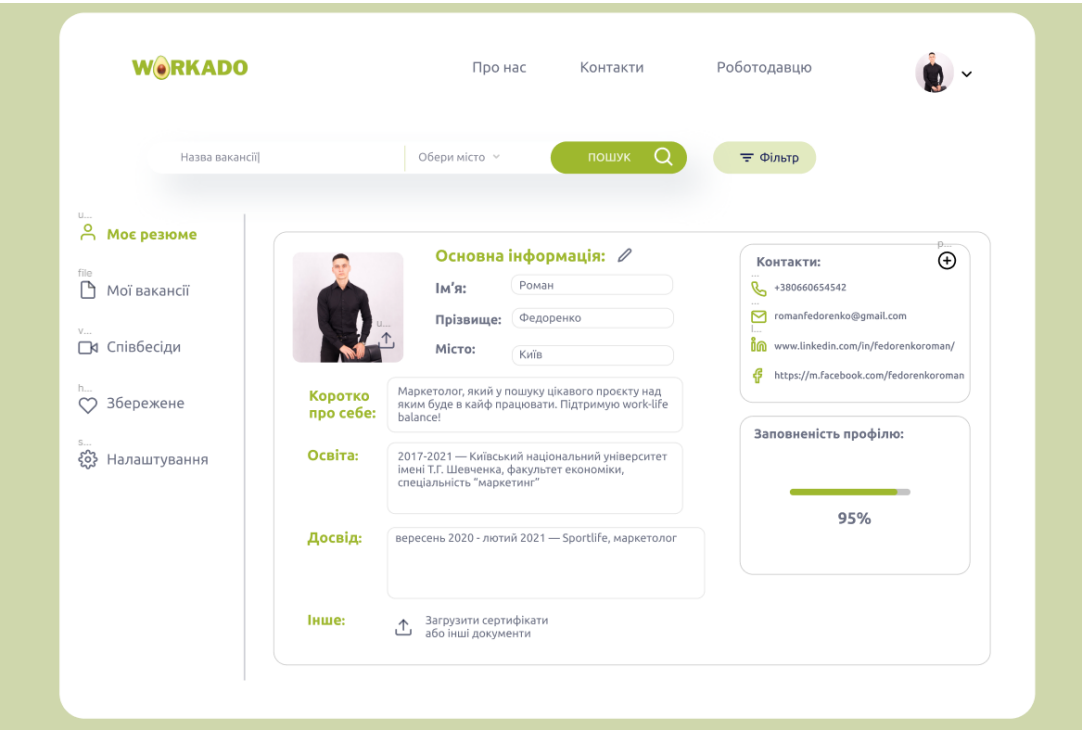


Рис. 3.22. Дизайн сторінки управління профілем шукача

Сторінка представляє собою звичайний шаблон резюме. В майбутньому, поле про досвід та навички кандидата, стануть більш інтерактивними.

Платформа надає користувачу зберігати вакансії щоб переглянути їх пізніше. Даний функціонал було обмежено для оптимізації, і користувач може зберегти вакансію тільки на сторінці самої вакансії. Дизайн сторінки вакансії приведений на рис. 3.23.

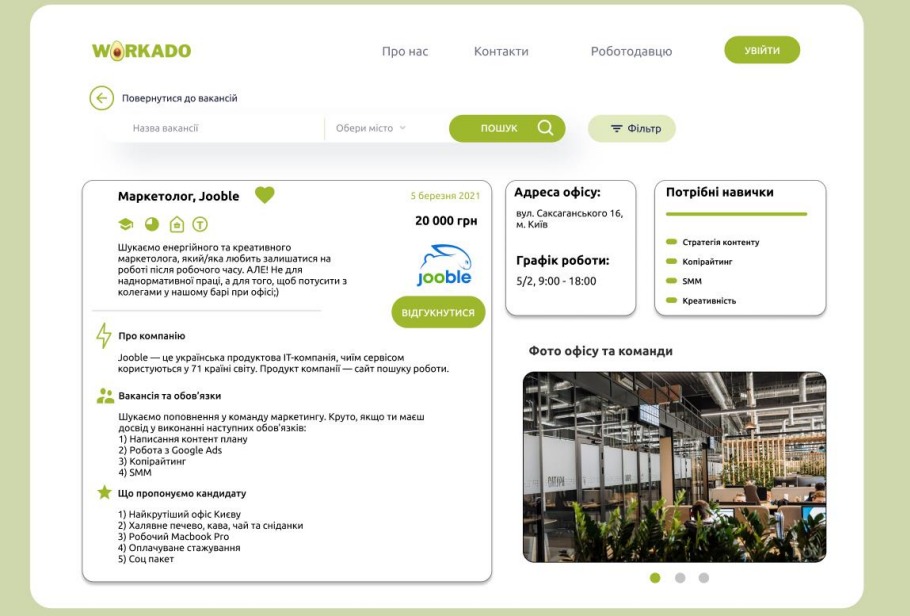


Рис. 3.23. Дизайн сторінки вакансії збереженої користувачем

На даній сторінці можна видалити вакансію зі збережених натиснувши на серце.

3.5.5 Модуль автентифікації та авторизації

Для ідентифікації користувача та роботи з JWT токеном, необхідно було модифікувати стандартний воркфлоу Spring Security. Для цього були розроблені реалізації основних класів які показані на рис. 3.24.

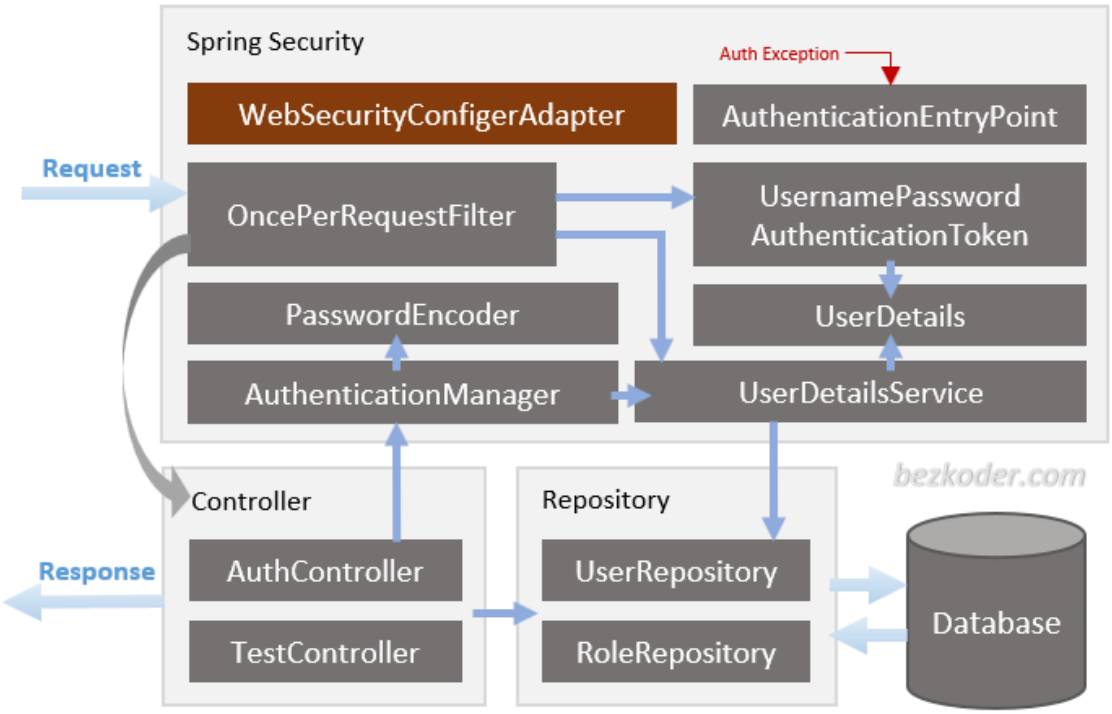


Рис. 3.24. Діаграма воркфлоу Spring Security

WebSecurityConfigurerAdapter головний клас для реалізації безпеки. Він надає конфігурації HttpSecurity для налаштування cors, csrf, управління сесіями, правил для захищених ресурсів. Інтерфейс UserDetailsService має метод завантаження користувача за іменем користувача та повертає об’єкт UserDetails, який Spring Security використовує для автентифікації та перевірки привілегій. UserDetails містить необхідну інформацію наприклад: ім’я користувача, пароль, повноваження для побудови об’єкта автентифікації. UsernamePasswordAuthenticationToken отримує ім'я користувача та пароль із запиту на вхід. AuthenticationManager використовуватиме його для автентифікації облікового запису для входу. AuthenticationManager має DaoAuthenticationProvider для перевірки об’єкта UsernamePasswordAuthenticationToken. У разі успіху AuthenticationManager повертає повністю заповнений об’єкт автентифікації (включаючи надані повноваження). OncePerRequestFilter робить одне виконання для кожного запиту до API. Він надає метод doFilterInternal(), за допомогою якого здійснено синтаксичний аналіз та перевірку JWT.

Для оптимізації, ролі користувача в системі було поміщено в самий JWT токен. Це дозволить зменшити кількість можливих запитів до бази даних вдвічі, але відкриває особливості роботи системи, адже саму інформацію може прочитати кожен.

Основні реалізовані класи показані на рис 3.25.

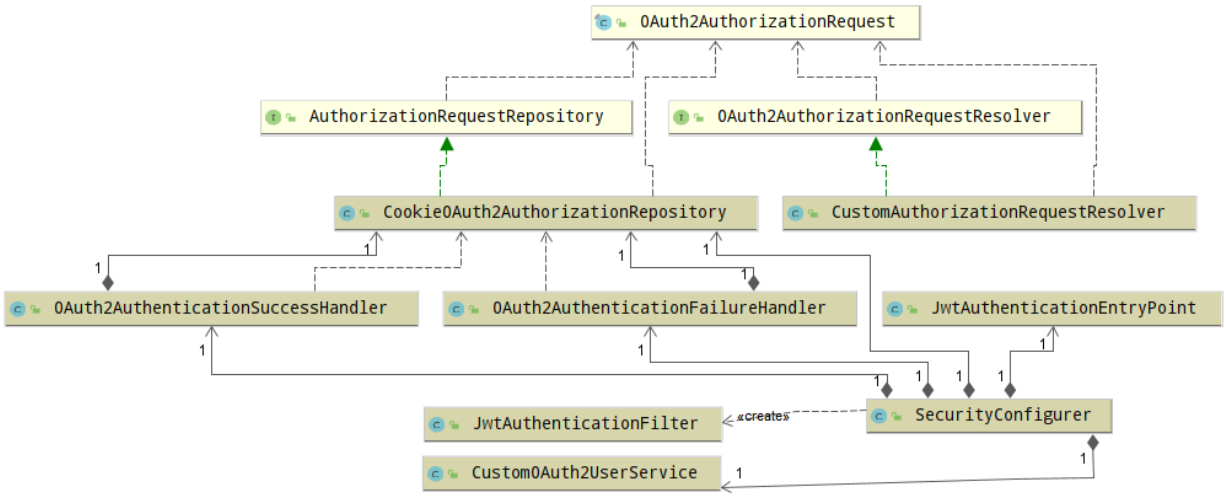


Рис. 3.25. Діаграма реалізованого модулю автентифікації та авторизації

Також класом CustomOAuth2UserService реалізована можливість зареєструватись та ввійти за допомогою Google. Фреймворк вже має налаштовані ендпоінти на які потрібно надіслати інформацію користувача, тому необхідно законфігурувати інформацію щодо зареєстрованого додатку, який буде запитувати доступ до певної інформації профілю.

Так як в платформі присутня клієнтська частина, необхідно зберегти інформацію щодо сторінки редиректу яку передав клієнт. Для цього використовуються Cookie браузеру, які потім знищуються сервером. Клас CookieOAuth2AuthorizationRepository відповідає за встановлення та роботу з Cookie при аутентифікації через Google.

Для забезпечення правильної роботи з JWT токеном, був реалізований інтерфейс OncePerRequestFilter класом JwtAuthenticationFilter який перевіряє правильність та валідність переданого токену та зчитує дані з нього дані.

Схема таблиць цього модулю представлена на рис 3.26.

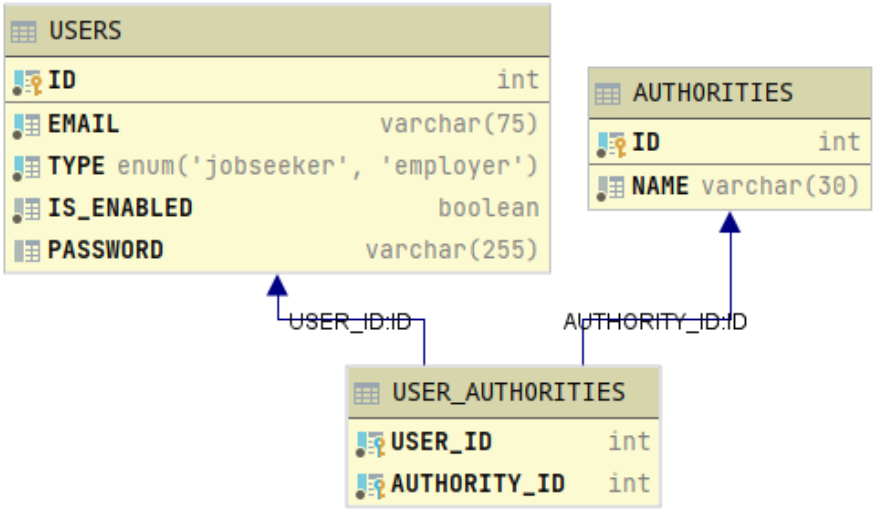


Рис. 3.26. Схема таблиць модулю автентифікації та авторизації

Дані користувачів зберігаються в таблиці Users з закодованим їх паролем. Дані про авторизацію знаходяться у таблиці Authorities, яка поєднана відношенням багато до багато через таблицю Users.

Висновки до розділу

Для платформи підбору персоналу з автоматизованим оцінюванням компетенцій були вибрані інструменти реалізації які дозволяють швидко розробити прототип системи. Також був створений дизайн та були спроектовані схеми основних класів модулів. Модулі спроектовані окремо, тому можлива легка їх модифікація в майбутньому.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ТА ВИКОРИСТАННЯ СЕРВІСА

4.1 Реалізація метода семантичного порівняння текстів

Алгоритм для семантичного аналізу був реалізований за допомогою інструментів представлених у Розділі 3. Трійки представлені у вигляді класу об’єкта Triple який містить в собі три частини трійки. Також реалізація має спеціальний клас ScoredPair який містить пари об’єктів та їх міру подібності.

Це дозволяє спростити код для виділення ексклюзивних пар який показаний на рис. 4.1.

```
private List<ScoredPair<Triple>> constructPairs(Collection<Triple> firstTextTokens,
                                                Collection<Triple> secondTextTokens) {
    List<ScoredPair<Triple>> possibleTriples = firstTextTokens.stream() Stream<Triple>
            .flatMap(t1 -> secondTextTokens.stream()
                    .map(t2 -> ScoredPair.of(getSimilarity(t1, t2), Pair.of(t1, t2)))
                    .filter(sp -> sp.score > similarityThreshold)) Stream<TextsComparat
            .sorted(scoredPairComparator.reversed())
            .collect(Collectors.toList());

    Collection<Triple> tokens1 = new ArrayList<>(firstTextTokens);
    Collection<Triple> tokens2 = new ArrayList<>(secondTextTokens);
    List<ScoredPair<Triple>> results = new ArrayList<>();
    for (ScoredPair<Triple> scoredPair : possibleTriples) {
        Pair<Triple> words = scoredPair.getWords();
        Triple firstWord = words.getWord1();
        Triple secondWord = words.getWord2();

        if (tokens1.contains(firstWord) && tokens2.contains(secondWord)) {
            tokens1.remove(firstWord);
            tokens2.remove(secondWord);

            results.add(scoredPair);
        }

        if (tokens1.isEmpty() || tokens2.isEmpty())
            break;
    }
    return results;
}
```

Рис. 4.1. Реалізація частини алгоритму по знаходженню ексклюзивних пар

4.2 Реалізація модуля управління процесом найму

Платформа керує всім процесом найму. Для цього необхідно було реалізувати стейт-машину для переходу між всіма можливими етапами. При аналізі етапів рекрутингу були виділені та представлені в системі етапи:

- TEST_IN_PROGRESS – кандидат розпочав прескринінг;
- TEST_FAILED – поріг проходження прискрінінгу був не пройдений кандидатом;
- EMPLOYER_ENTRY_PENDING – кандидатура на розгляді у рекрутера;
- ENTRY_REJECTED_BY_JOBSEEKER – рекрутер відмовив кандидату на етапі попереднього відбору;
- JOBSEEKER_ENTRY_REJECTED_BY_EMPLOYER – кандидат відмовився при знаходженні на етапі попереднього відбору;
- INTERVIEW_NOT_SCHEDULED – очікується вибір кандидатом вільного таймслота для проведення інтерв’ю з рекрутером;
- INTERVIEW_PENDING – очікується проведення інтерв’ю;
- INTERVIEW_REJECTED_BY_JOBSEEKER – кандидат відмовився від інтерв’ю;
- JOBSEEKER_INTERVIEW_REJECTED_BY_EMPLOYER – рекрутер відмовив кандидатові після інтерв’ю;
- JOBSEEKER_OFFER_PENDING – кандидатові був надісланий запрошення на роботу в компанію;
- OFFER_REJECTED_BY_JOBSEEKER – кандидат відмовився від запрошення на роботу;
- OFFER_ACCEPTED – кандидат прийняв запрошення на роботу;
- VACANCY_CLOSED – вакансія була закрита.

Залежно від етапу, рекрутер та кандидат можуть вибрати дії: прийняти, відмовити або видалити. Наприклад рекрутер не може відмовити кандидату на етапі інтерв'ю або до часу проведення інтерв'ю.

Приклад конфігурування класу відповідального за перехід між етапами на рис.4.2.

```
 = StageTransitionHolder.of();
.addTransition(EMPLOYER_ENTRY_PENDING,  approvalStageTransition: null, this::toEntryRejectedByJobseeker);
.addTransition(INTERVIEW_NOT_SCHEDULED,  approvalStageTransition: null, this::toInterviewRejectedByJobseeker);
.addTransition(INTERVIEW_PENDING,  approvalStageTransition: null, this::toInterviewRejectedByJobseeker);
.addTransition(JOBSEEKER_OFFER_PENDING, this::toOfferAcceptedTransition, this::toOfferRejectedTransition);

.addDeletion(TEST_FAILED, this::hideRecruitmentByJobseeker);
.addDeletion(JOBSEEKER_ENTRY_REJECTED_BY_EMPLOYER, this::hideRecruitmentByJobseeker);
.addDeletion(JOBSEEKER_INTERVIEW_REJECTED_BY_EMPLOYER, this::hideRecruitmentByJobseeker);
.addDeletion(OFFER_ACCEPTED, this::hideRecruitmentByJobseeker);
.addDeletion(VACANCY_CLOSED, this::hideRecruitmentByJobseeker);


= StageTransitionHolder.of();
addTransition(EMPLOYER_ENTRY_PENDING, this::toInterviewNotScheduledTransition,
                                      this::toEntryRejectedByEmployerTransition);
addTransition(INTERVIEW_PENDING, this::toOfferPendingTransition, this::toInterviewRejectedByEmployer);

addDeletion(INTERVIEW_REJECTED_BY_JOBSEEKER, this::hideRecruitmentByEmployer);
addDeletion(OFFER_REJECTED_BY_JOBSEEKER, this::hideRecruitmentByEmployer);
```

Рис. 4.2. Конфігурація класу для автоматичного переходу між етапами рекрутингу

Для спрощення клієнтської частини, використовується запит який має тільки три статуси:

- ACCEPT – допустити кандидата до наступного етапу, або прийняти запрошення на інтерв'ю чи на роботу;
- REJECT – відмовити кандидату в проходженні наступного етапу, або відмовитись від запрошення на інтерв'ю чи на роботу;
- REMOVE – видалити статус з наявного на інтерфейсі.

Наявні запити щодо інформації про всі етапи рекрутингу містять функціонал для фільтрації по етапам. Це дозволило зробити серверну систему більш гнучкою, але клієнтська частина повинна знати про всі наявні статуси.

4.3 Використання сервісу

Для користування платформою, необхідний вихід в інтернет та браузер. Платформа доступна за посиланням https://workado.io. Наразі вона знаходиться на етапі MVP, тому деякий функціонал недоступний для звичайного користувача.

Сайт призначений для рекрутерів та дозволяє їм зменшити кількість непідходящих за вакансією кандидатів через попередній прескринінг платформою. Також сайт слугує зручною системою для управління етапами найму за кожною вакансією та кандидату.

Для залучання кандидатів на платформу, спочатку створений, який доступний за посиланням https://t.me/workadojobs. Це зроблено за для створення бренду і набору аудиторії перед запуском.

Висновки до розділу

Був реалізований алгоритм метода семантичного порівняння текстів за для впровадження даного функціоналу в модулі прескринінгу. Реалізація всієї платформи рекрутингу була виконана з використання об’єктно-орієнтованого стилю та найкращих практик. Для реалізації модуля управління процесом найму, була реалізована стейт-машина для автоматизації переходу при зміні етапу рекрутингу.

Розроблена платформа призначення для комерційного використання та знаходиться на стадії MVP для залучення інвестицій. Проект бере участь у всеукраїнському конкурсі стартапів The Ukrainian Startup National Digital Competition 2021 від fibstartup [14].

ВИСНОВКИ

В дипломній роботі був проаналізований процес рекрутингу, а саме етап відбору кандидатів на вакансію. Проблема фільтрації великої кількості некваліфікованих кандидатів призводить до збільшення ресурсів які необхідні для закриття однієї вакансії. Також це було підтверджено рекрутерами з різних сфер в процесі інтерв'ю та опитування.

Для вирішення цієї проблеми, було побудовано платформу з використанням ReactJS та серверної частини на Java. Платформа включає можливість створювати профіль для рекрутера та шукача роботи, розміщати вакансії, контролювати процес рекрутингу з обох сторін, налаштовувати таймслоти для інтерв'ю та проходити прескринінг. Це було необхідно за для тестування гіпотез для стартапу "Workado.io", але після ітерацій продукту, було вирішено сконцентруватись на проблемі прескринінгу яка являється наразі найперспективнішою з існуючих.

Модуль прескринінгу включає в себе просте тестування з одним або багатьма варіантами відповіді, та аналізу відповідей на відкриті питання. Головна вимога до задачі була в максимальній автоматизації процесу, тому для аналізу відповідей був реалізований метод семантичного порівняння текстів на основі семантичних трійок, який було порівняно з методом порівняння через жадібне сполучення [8] за для визначення ефективності.

Також, реалізований модуль оцінки правильності відкритих питань може бути використаний не тільки для даної системи, але наприклад для перевірки текстів на новизну та плагіат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ТОП-5 проблем українського IT-рекрутинга – DOU – [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/26660/.

2. 6 признаков того, что профессия рекрутера не для вас – Laba – [Електронний ресурс]. – Режим доступу: https://l-a-b-a.com/blog/2186-6-priznakov-togo-chto-professiya-rekrutera-ne-dlya-vas.

3. 23 Surprising Stats on Candidate Experience Infographic – careerarc – [Електронний ресурс]. – Режим доступу: https://www.careerarc.com/blog/candidate-experience-study-infographic.

4. The true business cost of a poor candidate experience – Human Resource Executive – [Електронний ресурс]. – Режим доступу: https://hrexecutive.com/the-true-business-cost-of-a-poor-candidate-experience/.

5. 5 Data-backed Insights That Will Improve Your Candidate Experience – LinkedIn – [Електронний ресурс]. – Режим доступу: https://www.linkedin.com/business/talent/blog/talent-strategy/insights-on-how-to-improve-your-candidate-experience.

6. Nearly Three in Four Employers Affected by a Bad Hire, According to a Recent CareerBuilder Survey – careerbuilder – [Електронний ресурс]. – Режим доступу: http://press.careerbuilder.com/2017-12-07-Nearly-Three-in-Four-Employers-Affected-by-a-Bad-Hire-According-to-a-Recent-CareerBuilder-Survey.

7. Han J., Pei J. Cosine similarity. Data Mining (third edition). 2012..

8. M. Lintean and V. Rus, "Measuring Semantic Similarity in Short Texts through Greedy Pairing and Word Semantics", in Twenty-Fifth International Florida Artificial Intelligence Research Society Conference, Florida, 2012.

9. Conference System of NULES of Ukraine, Global and regional problems of informatization in society and nature using 2021 – [Електронний ресурс]. – Режим доступу: http://econference.nubip.edu.ua/index.php/grpi/grpi21/paper/view/2530.

10. CoreNLP – stanfordnlp – [Електронний ресурс]. – Режим доступу: https://stanfordnlp.github.io/CoreNLP/.

11. "DISCO compute semantic similarity between words", linguatools, 2020. [Online]. Available: https://www.linguatools.de/disco/. [Accessed: 03- May- 2021].

12. enwiki-20130403-sim-lemma-mwl-lc– linguatools – [Електронний ресурс]. – Режим доступу: https://www.linguatools.de/disco/disco-wordspaces.html#enwiki13slm.

13. ReactJS Tutorial: A Step-by-Step Guide To Learn React – simplilearn – [Електронний ресурс]. – Режим доступу: https://www.simplilearn.com/tutorials/reactjs-tutorial.

14. Змагання «Start-UP – National Digital Competition» – [Електронний ресурс]. – Режим доступу: https://fibstartup.com/.

ДОДАТКИ

Додаток А

Software Architecture Document

Visual object tracker system
Software Architecture Document (SAD)

CONTENT OWNER: Harbar Dmytro

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION DATE:
• 1.0.0	• 1.0.0	• 30.05.2021

Table of Contents

Documentation Roadmap

1.1 Document Management and Configuration Control Information

- Revision Number: 1.0.0
- Revision Release Date: 28.05.2021
- Purpose of Revision: first release of the document

1.2 Purpose and Scope of the SAD

This Software Architecture Document (SAD) specifies the software architecture for **recruiting system platform.** All information regarding the software architecture may be found in this document, although much information is incorporated by reference to other documents. SAD provides a comprehensive architectural overview of the recruiting system. It presents a number of different architectural views to depict the different aspects of the system. These structures will be represented in the views of the software architecture that are provided in Section 3.

1.3 Viewpoint Definitions

As required by ANSI/IEEE 1471-2000, this SAD employs a stakeholder-focused, multiple view approach to architecture documentation. A viewpoint identifies the set of concerns to be addressed and a view is a viewpoint applied to a system. In current section are presented and defined viewpoints used in this SAD. The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns. In the next subsections each viewpoint is shortly presented.

Use Case view

Audience: all the stakeholders of the system, including the end-users.
Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax.
Related Artifacts : Use-Case Model, Use-Case documents

Logical view

Audience: architects.
Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.
Related Artifacts: Design model

Process view

Audience: Integrators.
Area: Non-functional requirements: describes the design's concurrency and synchronization aspects.
Related Artifacts: (no specific artifact).

Deployment view

Audience: Deployment managers.
Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.
Related Artifacts: Deployment model

2 Architecture Background

2.1 Problem Background

2.1.1 System Overview

The purpose of the system is to provide recruiters with a convenient platform for finding qualified candidates. The entire path of the candidate from search to receiving a job offer must be presented on the platform. The recruiter should be able to conveniently post jobs that the job seeker can view. If a job seeker wants to apply for a vacancy, he can fill out a short pre-screening to determine his qualifications. Qualification should be done automatically by the system using tests and open-ended questions. The system determines how suitable the applicant is and shows a certain percentage to the recruiter.

2.1.2 Goals and Context

The main task is to develop the server side of the platform with a convenient REST API that can be easily integrated with your favorite client application. System components should not be highly dependent as many modifications are planned to test future hypotheses.

The platform should provide:

- subscription plan for recruiter;
- interactive among vacancies;

The candidate must be able to:

- register on the platform;
- manage candidate profile information;
- choose recruiter timeslot for interview;
- deny recruiter proposal;
- fill out pre-screening;

The recruiter should be able to:

- register on the platform;
- manage recruiter profile information;
- be a member and join a company;
- create, modify and delete vacancies;
- set up prescreening;
- view company vacancies;
- create timeslots for interviews;
- refuse and accept a candidatee;

2.1.3 Significant Driving Requirements

The main key requirement is the functionality of the prescreening module. This module should be able to work with open-ended questions that come from the candidate. This functionality can be achieved by implementing the semantic analysis method.

2.2 Solution Background

2.2.1 Architectural Approaches

A three-tier architecture was chosen for the platform. There are three main components that interact with each other by transmitting DTO objects:

- Client or presentation level, responsible for interacting with the application, and processes incoming requests. For easy interaction with the application was presented REST API implemented using the Controller Spring framework.
- The next level is responsible for the basic business logic of the application. It contains all the calculations and auxiliary structures. It is the business layer that interacts with the data level.
- The data layer should perform a simple database interaction function. That is, this level should not know how this information will be used or why, its task is to best process the request to the data warehouse.

This architecture was chosen because of its flexible structure and clear module framework. This leads to some difficulty in implementing while maintaining the independence of the levels, but this problem can be neglected with a certain level of skill of the developer.

2.2.2 Analysis Results

To check the quality of determining the correctness of the answers to open-ended questions, the screening method was tested, namely the effectiveness of the implemented method of semantic comparison of sentences was evaluated in the problem of automatic evaluation of open text answers by comparison with the standard.

The test sample contained 210 pairs of reference-answers, in which most words coincide, but the value of the answer can be both identical and opposite to the standard. In this sample, the result of the method was compared with the existing method of calculating semantic similarity through greedy combination and semantics of words. In order for the method to be informative for evaluating answers, for meaningfully close sentences its result should be close to 1, and for significantly different sentences - up to 0. The proposed method provides an average correspondence value of 0.72114 in the first case and 0.19217 in the second. In the same sample, the greedy pairing method gives average values of 0.29365 and 0.30655, respectively. This allows us to conclude that taking into account the relationship between words according to the proposed method is advantageous in assessing both knowingly correct and incorrect answers in the case of statistically similar sentences, when the existing method of greedy combination is ineffective.

2.2.3 Requirements Coverage

The main requirement was the quality implementation of the screening module. The results of the module were compared with the existing method, where the implemented module showed better results. Additional requirements and with wider functionality were also implemented. For example, the login was implemented through synchronization with Google.

The main requirements are covered and results of analysis provided in previous section.

3 Views

This section contains the views of the software architecture described in Section 1.3.

3.1 Use Case view

3.1.1 View Description

This view shows use-cases for recruiter and job seeker.

3.1.2 View Overview

View represented as use-case diagram UML.

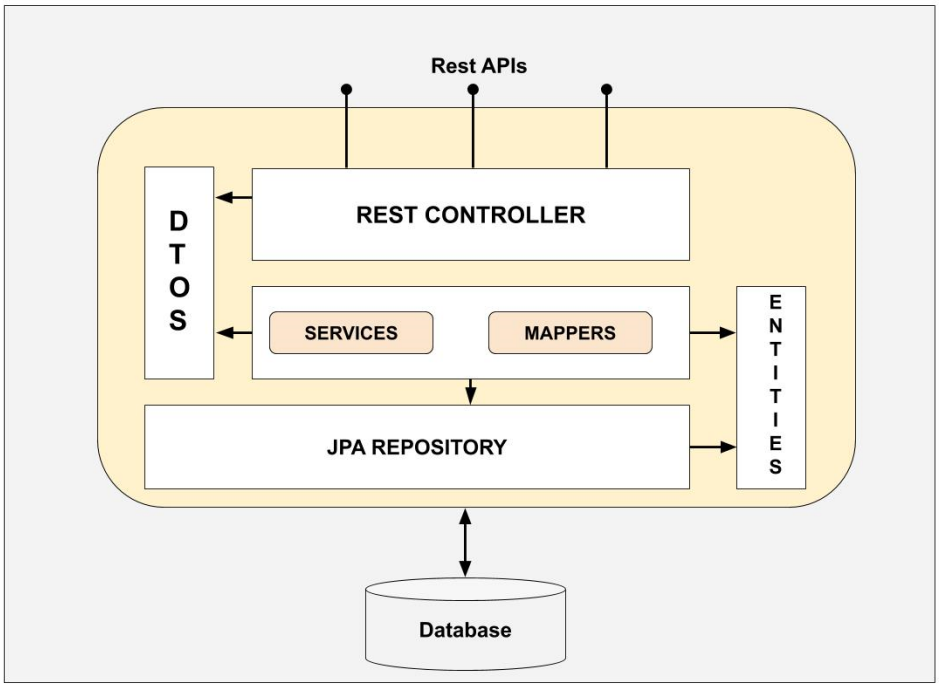
3.2 Logical View

3.2.1 View Description

A three-tier architecture was chosen for the platform. There are three main components that interact with each other by transmitting DTO objects.

3.2.2 View Overview

View represented as scheme based on architecture of the platform.



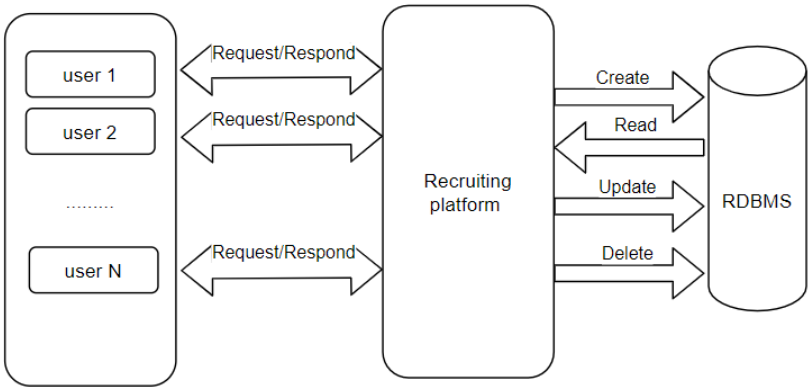
3.3 Process View

3.3.1 View Description

This view shows the process of user interaction with the system.

3.3.2 View Overview

View shows the process of user interaction with the system. Each request is stateless and therefore does not require creating and maintaining a session for each new request.



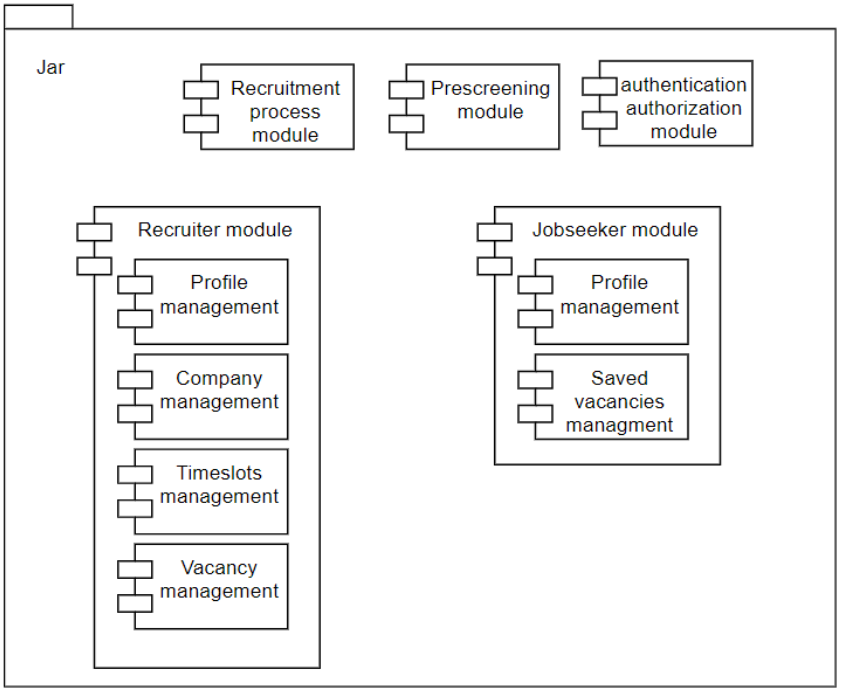
3.4 Deployment View

3.4.1 View Description

This view allocates software elements which are native to a component & connector style to the hardware of the computing platform on which the software executes.

3.4.2 View Overview

View represented as diagram built with informal graphical notations that use boxes, circles, lines, arrows, and so on to represent the software and environmental elements.



4 Referenced Materials

Clements 2010	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2010.
IEEE 1471	ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000.

5 Directory

5.1 Glossary

Term	Definition
software architecture	The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.
view	A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.
viewpoint	A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view.

5.2 Acronym List

API	Application Programming Interface
SAD	Software Architecture Document
UML	Unified Modeling Language
DTO	Data transfer object
REST	Representational State Transfer