НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)

“__” _____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

на тему: HRCRM – система управління персоналом

Виконав : студент __4__ курсу, групи _ІП-83_
(шифр групи)

_____Подаш Антон Миколайович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____асистент Волокита І. М._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) професор, д.т.н., Сімоненко В. П._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Подаша Антона Миколайовича

1. Тема проєкту HRCRM – система управління персоналом
керівник проєкту Волокита Іван Миколайович асистент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 11 травня 2022 року №1139-с

2. Термін здачі студентом закінченого проєкту *30 травня 2022 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд існуючих систем.

Розділ 2. Огляд алгоритмів та технологій для розробки системи.

Розділ 3. Деталі розробки системи.

Розділ 4. Огляд реалізованої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, даталогічна схема, алгоритм дій користувача.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання «30» серпня 2021 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2021-15.12.2021*	
2.	*Вивчення та аналіз завдання*	*15.12.2021-15.02.2022*	
3.	*Розробка архітектури та загальної структури системи*	*15.02.2022-24.03.2022*	
4.	*Розробка структур окремих підсистем*	*01.04.2022-15.04.2022*	
5.	*Програмна реалізація системи*	*15.04.2022-25.04.2022*	
6.	*Оформлення пояснювальної записки*	*30.04.2022-30.05.2022*	
7.	*Захист програмного продукту*	*02.06.2022*	
8.	*Передзахист*	*11.06.2022*	
9.	*Захист*	*24.06.2022*	

Студент-дипломник _____ АНТОН ПОДАШ
(підпис)

Керівник проєкту _____ Артем ВОЛОКИТА
(підпис)

АНОТАЦІЯ

У даній роботі було детально розглянуто 2 популярній системи для управління персоналом. Був зроблений їх повноцінний технологічний та користувацький аналіз. На основі аналізу були вибрані найкращі практики в розробці подібних систем, функціонал та програмні модулі, які увійшли в основу розробленої програми. В результаті роботи було розроблено повноцінну систему, її серверну частину та користувацький інтерфейс. За основу було взято мікросервісну архітектуру, що дає можливість в майбутньому ризширювати функціонал системи, додаючи необхідні модулі, таким чином система є індивідуальною під кожного користувача. Система була розроблена технологічному стеку Node.js, Express.js, Next.js.

Ключові слова: рекрутинг, управління ефективністю, мікросервіси, персонал, Next.js

ANNOTATION

In this project, 2 popular personnel management systems were considered in detail. Their full-fledged technological and user analysis was made. Based on the analysis, the best practices in the development of such systems, functional and software modules were selected, which became the basis of the developed program. As a result, a full-fledged system, its server part and user interface were developed. The microservice architecture was taken as a basis, which allows to expand functional systems in the future, adding the necessary modules, so the system is individual for each user. The system was developed on the technology stack Node.js, Express.js, Next.js.

Keywords: recruitment, efficiency management, microservices, staff, Next.js

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***		3		
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	HRCRM – система	106		
			управління персоналом			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	HRCRM – система	1		
			управління персоналом			
			Структурна схема			
			системи			
	А4	***ІАЛЦ.4672008.005 Д2***	HRCRM – система	1		
			управління персоналом			
			Функціональна схема			
			(діаграма класів)			
	А4	***ІАЛЦ.467200.006 Д3***	HRCRM – система	1		
			управління персоналом			
			Алгоритм дій програмного			
			забезпечення			
	А4	***ІАЛЦ.467200.007 Д4***	HRCRM – система	41		
			управління персоналом			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА		
Зм	*Лист*	*№ докум.*	*Підп*	*Дата*			
Розроб		Подаш А.М.			*Еволюційні алгоритми глобальної пошукової оптимізації* Опис альбому	Літ.	Аркуш 1 / Аркушів 1
Перев		Волокита І.М.					
						НТУУ "КПІ" ФІОТ ІП-73	

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*HRCRM – система управління персоналом*»

Київ – 2022 р.

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку HRCRM системи управління персоналом, а також на подальшу підтримку та вдосконалення розробленої системи.

Областю застосування цієї системи є використання її приватними компаніями на постійному рівні у вигляді єдиного інструмента для вирішення питань продуктивності, найму та ведення працівників компанії.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка HRCRM – системи управління персоналом, що дозволить керівникам компаній та окремих відділів, ефективно відслідковувати ефективність працівників та полешить процес пошуку нових працівників.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс системи, який відповідає сучасним стандартам дизайну користувацького інтерфейсу.
- Надати можливість працівникам компаії, яка буде використовувати дану систему, ставити та відслідковувати завдання та глобальні задачі.
- Полегшити процес підбору персоналу в компанію.
- Створити єдине місце для ведення усієї інформації працівників компанії.
- Бути максимально простою у використанні, без необхідного додаткового навчання користувачів.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Веб браузер Safari, Google Chrome або інший.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-5100.
- ROM не менше ніж 128 ГБ.
- RAM не менше ніж 2 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	*10.12.2021-15.12.2021*
Вивчення та аналіз завдання	*15.12.2021-15.02.2022*
Розробка архітектури та загальної структури системи	*15.02.2022-24.03.2022*
Розробка структур окремих частин системи	*01.04.2022-15.04.2022*
Програмна реалізація системи	*15.04.2022-25.04.2022*
Виправлення помилок	*25.04.2021-30.04.2021*
Оформлення пояснювальної записки	*30.04.2022-30.05.2022*

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*HRCRM – система управління персоналом*»

Київ – 2022 р.

ЗМІСТ

ВСТУП

Робота в сучасних компаніях пов’язана з використання великого обсягу різних інструментів, застосунків та систем. Вони є необхідним елементом в робочі на сьогоднішній день. Проте, часто керівники компаній стикаються з проблемою, що кількість інтрументів для виконання щоденних завдань є надно великою та коли в команді з’являєтсья новий працівник потрібен тривалий час, поки він зможе пристосуватись та навчитись використовувати технічні інструменти для роботи. До списку систем, які сьогодні використовують сучасні компанії можна віднести:

- Менеджери завдань;
- Проєктні менеджери;
- Календарі;
- Застосунки для відео конференцій;
- Системи для управління клієнтами компанії;
- Системи для управління персоналом;
- Месенджери;
- Електронні пошти;
- Текстові редактори;
- Системи ведення фінансів;

Для ефективного вирішення задач, звичайному працівнику потрібно тримати відкрити від 5 до 12 програм на робочому комп’ютері, що в свою чергу знижує продуктивність системи та працівника. Також не завжди можна пам’ятати яка задача де саме вирішуєтсья, що збільшує час на її виконання.

У даній роботі я розгляну найпопулярніші застосунки для управління персоналом компанії (найм нових працівників, відслідкування ефективності, ведення працівників). На основі їх аналізу буде зроблено порівняльну таблицю з плюсами та мінусами кожної з систем.

Дана робота являє собою детальний аналіз та розробку системи для управління персоналом, яка буде модульною, що дозволить налаштовувати її функціонал індивідуально та збільшувати його за потреби.

РОЗДІЛ 1 ОГЛЯД І ПОРІВНЯННЯ ІСНУЮЧИХ СИСТЕМ

1.1 Система Acsender

1.1.1 Можливості та особливостні функціоналу системи

Ця система надає досить значний функціонал, та розподілена на декілька модулів, які кожен керівник компанії може компонувати та об’єднувати на власний розсуд, що дає великий спектр можливостей у використанні даного застосунку. Кожен з цих модулів відповідає за окремі процеси в роботі з працівниками у вашій компанії, що дозволяє користуватись саме тим програмним забезпеченням, яке необхідне для виконня ваших завдань. Кожен з модулів має окремі особливості та індивідуальний функціонал, який буде розглянуто нижче.

Індивідуальний досвід для співробітників

Незалежно від того, чи вітаєте ви нового працівника чи наявного співробітника на новій посаді, ви можете допомогти їм підключитися та інтегруватися зі своєю новою командою за допомогою програмного забезпечення для адаптації. Це дає можливість працівникам познайомитися з культурою організації з самого початку та допоможе їм легко знайти, хто є хто і що є у вашій організації[2].

Можливість автоматичного заповнення форм і зберігання цифрових підписів

Заповнивши форму один раз, решту часу це буде відбуватись автоматично! Це зберігає цінний робочий час для співробітників, заповнюючи податкові форми, вступні контрольні списки, банківські форми, квитки на паркування тощо, не вводячи основну інформацію знову і знову. Вся інформація зберігається на захищеному сервері та використовується лише в разі потреби[4].

Онлайн та автоматизоване керування політикою роботи в компанії

Співробітники можуть підписувати всі свої обов'язкові правила в електронному вигляді. Це дає можливість створювати нові робочі процеси, які відповідають потребам компанії, щоб працівники могли відповідати необхідним вимогам та щоб ваша команда з працевлаштування не плуталась в нових правилах та їх актуальності[2].

Автоматизований курс навчання нових працівників

Новим працівникам більше не потрібно переходити до іншої системи, щоб закінчити необхідні вступні курси. Співробітники можуть пройти курси, створені вами, щоб допомогти їм розпочати роботу та отримати сертифікат про закінчення.

Чітке відстеження прогресу у вашій програмі адаптації

Багато компаній працюєте за програмою адаптації на три, шість, дванадцять місяців. Дає можливість керівникам та рекрутерам слідкувати за прогресом нового найму та переконайтися, що він добре відстежується в системі адаптації[2].

Прості у використанні інструменти для заповнення необхідних форм

Швидше, ніж будь-коли, надавши новим працівникам доступ до обов'язкових форм ще до першого робочого дня. Завдяки меншій кількості документів ваша команда з персоналу може зосередитися на тому, щоб дати вашим новим працівникам хороший досвід, щоб допомогти їм почати роботу.

1.1.2 Управління продуктивністю

Можливість легко створювати та налаштовувати розумні цілі та шкали оцінок

Цілі показники якості є обов'язковими для високоефективних співробітників, але написання значущих цілей/ключових показників ефективності може бути проблемою. Пошук програмного забезпечення, яке включає автора, може допомогти створити конкретні, вимірні, досяжні,

реалістичні та обмежені в часі цілі для ваших співробітників. Крім того, можливість налаштувати ваші рейтинги/системи вимірювання для кожної цілі також може допомогти співробітникам знати, що їм слід визначити пріоритетом для досягнення успіху.

Керування описами відкритих робочих позицій компанії та доступ до них в одному місці

Контроль над описом посади є основоположним для продуктивності. Зберігання описів ваших позицій в Інтернеті означає, що ваші співробітники можуть отримати до них доступ у будь-який час і в будь-якому місці, зменшуючи плутанину щодо своїх ролей, і дозволяє кожному легко призначити їх собі або своїм звітам. Більше не потрібно шукати інформацію про це в купі паперів чи найновішу версію опису[4].

Персональні перевірки ефективності зі статусами та примітками

Регулярні перевірки є необхідністю для успішного управління продуктивністю. Встановлення мети на початку року лише перевіряти прогрес через 12 місяців та призводить до втрати часу та зусиль. Для цього в "**Acsender**" створена система управління продуктивністю, яка може допомогти досягти цілей для всіх сторін. Спростивши цей процес, зустрічі віч на віч між працівниками, зі статусами та онлайн-нотатками, а також можливість відстежувати, що працює, а що потребує особливої уваги.

Узгодження цілей, щоб допомогти вашій організації досягти свого стратегічного плану

Усі цілі, завдання та показники ефективності на одній сторінці каскадного стилю. Це допомагає вашим керівникам узгоджувати співробітників із загальними бізнес-цілями, щоб кожен міг досягти однакових стратегічних результатів. Що дозволяє підтримувати продуктивність своїх співробітників, показуючи їм, що вони є важливою частиною загальної картини.

1.1.3 Програмне забезпечення для навчання та розвитку

Різні варіанти навчання, щоб покращити навички та продуктивність робочої сили

Можливість створити культуру навчання та розвитку, надаючи своїм співробітникам різноманітні можливості для навчання. Від навчання на робочому місці, коучингу та наставництва до різноманітних формальних навчальних заходів — можливість залучати свою робочу силу кількома способами, забезпечуючи максимальне збереження знань.

Доступ до аналітики та звітів

Вимірювання ефективності та цінності навчання та можливість отримати уявлення про показники, визначені вами в системі. Що надає уявлення про фактичні дані учнів, щоб ви могли приймати навчальні рішення на основі даних і переконатися, що ваші зацікавлені сторони також бачать загальну картину.

Бібліотека комплексного підвищення кваліфікації

Надає змогу підвищити кваліфікацію та утримати своїх співробітників, маючи доступ до різних навчальних тем, усе у них під рукою. Від лідерства, галузевих курсів відповідності вимогам, HR, комунікацій, обслуговування клієнтів, IT-тренінгів до навичок продажів, ваші співробітники можуть більше розвиватися у своїх поточних навичках і навіть вивчати інші функції, які вони можуть захотіти вивчити

Видимість прогресу навчання

Відстеження завершення курсів, надсилайте нагадування та відстежуйте прогрес тих, хто завершив свої етапи. Статистика навчання з одного центрального місця.

1.1.4 Програмне забезпечення для підбору персоналу

Спрощений процес подачі заявки та перевірки

Спрощує процес заявки для ваших кандидатів, зменшивши кількість кроків і час очікування. Дає рекрутерам поштовх, позбувшись ручної роботи з документами, а всю необхідну інформацію буде повною та доступною в одній централізованій системі. Зробивши процес набору персоналу ефективнішим, ви зможете отримати найкращі таланти раніше, ніж конкуренти.

Управління та відстеження заявок на роботу

Іноді здається, що існує нескінченний список ролей, які потрібно виконати, але не тоді, коли ваша команда з підбору персоналу має ефективний засіб відстеження, щоб переконатися, що вони на вершині. Використовуючи менеджер запитів на вакансії, ви маєте повний контроль над процесом підбору персоналу: від прийому резюме до подання заявки на оформлення.

Легка реклама вакансій

Створення рекламних оголошень на свої відкриті ролі там, де це важливо, щоб ви могли охопити цільових претендентів там, де вони є. Доступ до понад 600 дощок вакансій у всьому регіоні та легка публікація вакансій напозиції, які вам потрібні.

Доступ до більшого кадрового резерву з кількох джерел кандидатів

Облік усіх людей, які подали заявку до вашої організації на майбутні посади. Немає необхідності перебирати старі програми чи рекомендації співробітників. Відсортуйте та відфільтруйте наявний пул талантів, щоб знайти найкращі збіги, або створіть канал для ваших майбутніх відкритих вакансій.

1.1.5 Тестування та користувацький досвід

Перше знайомство користувача зданою системою відбувається з сайту компанії, на якому знаходиться детальна інформація про ресурс та його використання.

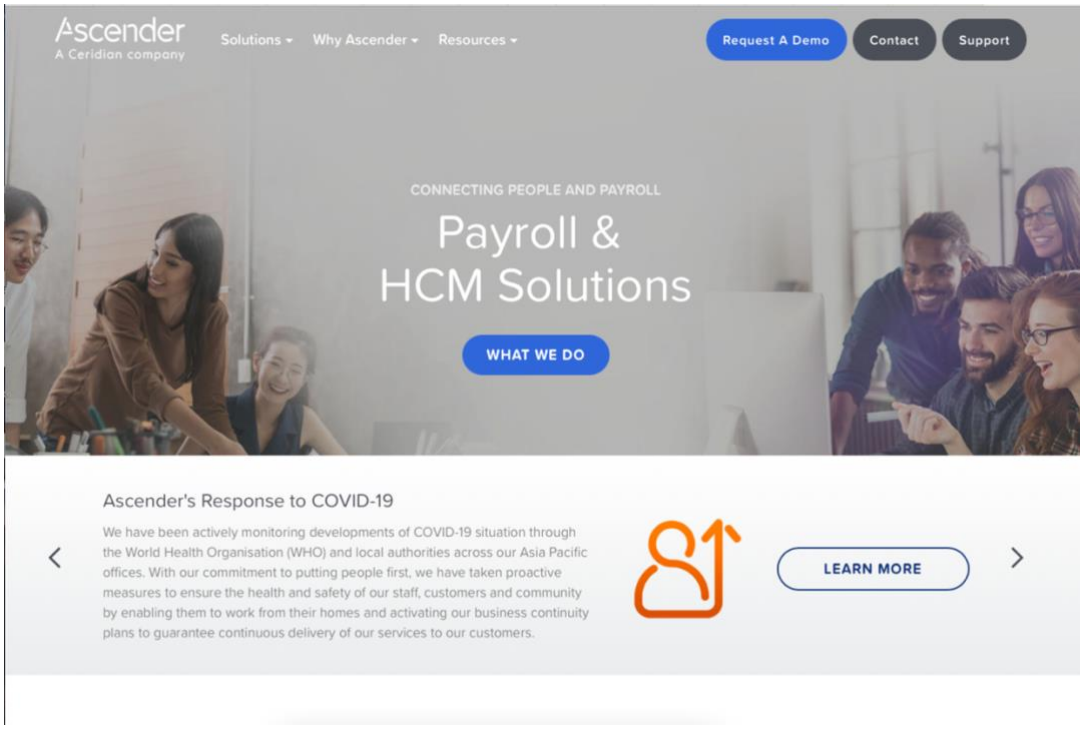


Рисунок 1.1 - Головна сторінка сайту систему Acsender

Пункти мені добре скомпоновані та чітко відображаються логіку користування сайтом. Проте немає пункта з цінами за користування послугами цієї системи. Поганим досвідом є те, що для реєстрації акаунта та початку користування необхідно залишати заявку з персональними данним та очікувати на дзвінок або повідомлення від менеджера компанії.

Набагато краще було б створити окрему сторінку для входу в додаток яка переспрямовувала користовача на урл адрессу “name.com/register”. На сторінці реєстрації повинна бути форма з полями для вводу данних, заповнивши які повинен прийти лист підвертдження на електронну адресу користувача.

1.1.6 Продуктивність та показники системи

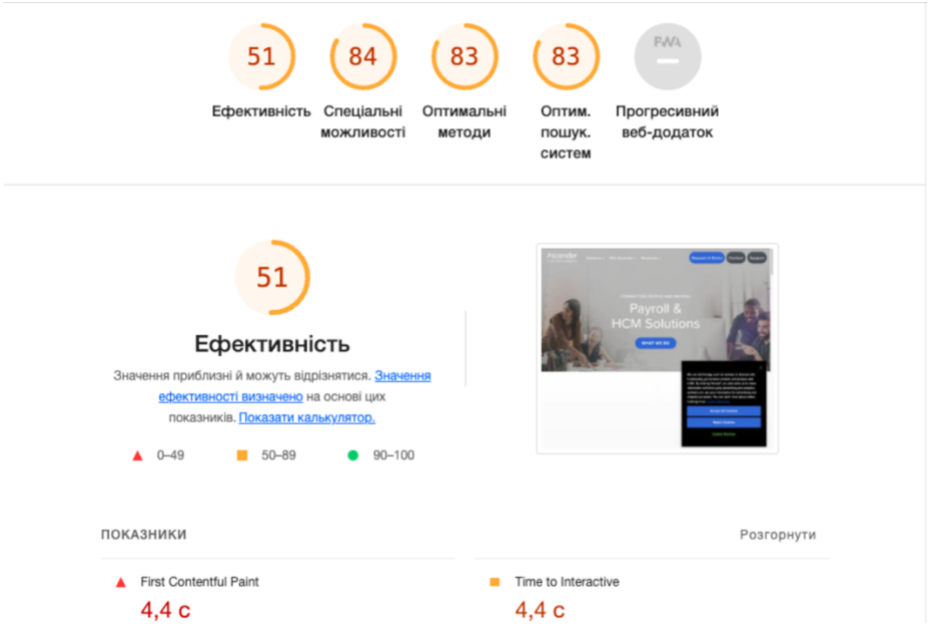


Рисунок 1.2 - Результати тестів системи Acsender в Lighthouse

Використовуючи застосунок для аналітики та оптимізації швидкості роботи сайту від компанії “Google” лайфтхауз, я протестував роботу цієї системи.

Швидкість роботи є досить малою, що не відповідає вимогам сучасних застосунків і в свою чергу призведе до того, що користувач закриє покине створінку не дочекавшись її завантаження.

Оптимальні методи є непоганими, проте оцінка не ідеальна через використання застарілої JavaScript бібліотеки Jquery.

Оптимізація для пошукових систем є нормальною, не вистачаєть тільки “alt” атрибутів для зображень на сайті, що дає змогу пошуковому двигуну гугл краще розпізнавати зображення з сайту та відображати їх в пошуку за відповідними запитами.

1.1.7 Інтерфейс системи

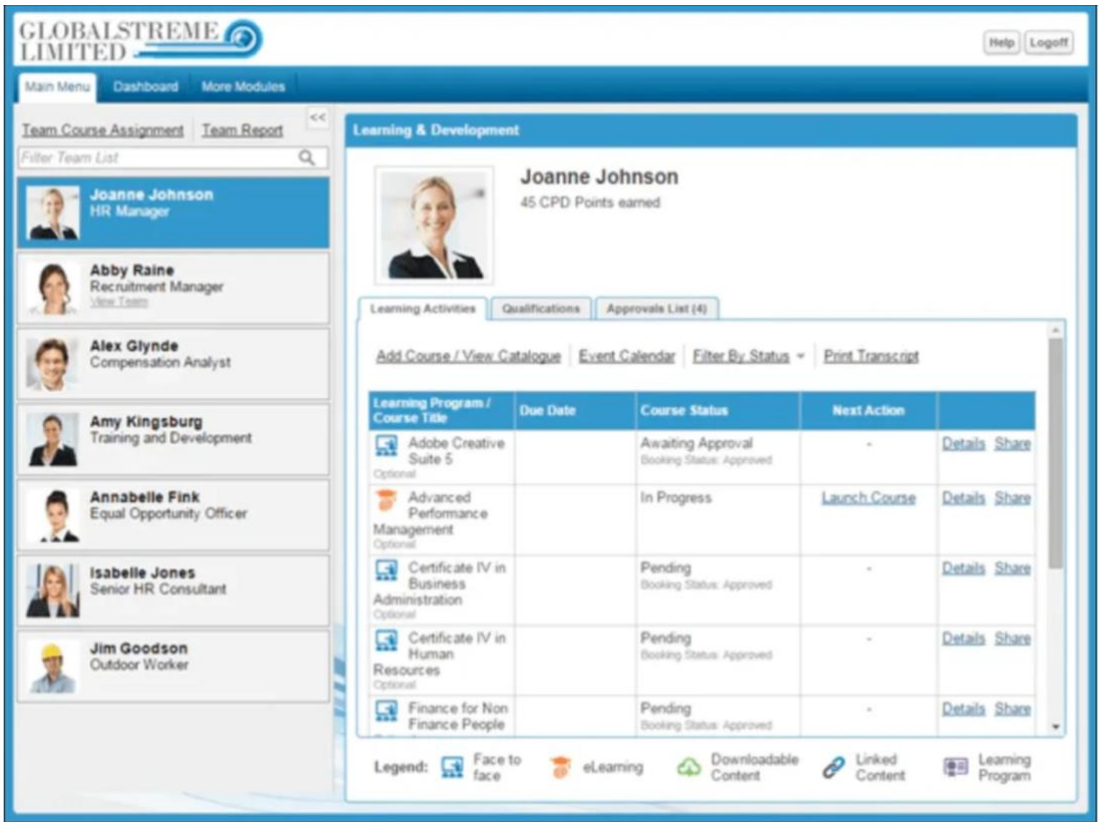


Рисунок 1.3 – Користувацький інтерфейс системи Acsender

На скриншоті з головного екрану застосунку, див. рис. 1.3, ми бачимо достатньо поганий інтерфейс, зразку 2007 року, який так і не оновився до цьогоденних стандартів дизайну. Всього є 3 вкладки які відповідають пунктам меню. Головне меню являє собою каскадний список з працівників компанії та основну інформацію про кожного. Іконки виконані не в єдиному стилі та мають різні кольори. Також тут присутньо більше двух шрифтів, що є поганим для сприйняття користувачу.

Таблиця 1.1. – Переваги та недоліки системи Ascender

Плюси	Мінуси
Модульна система роботи	Застарілий інтерфейс
Великий функціонал	Відсутність прямої реєстрації в системі
Хороша навчальна програма для користувачів	Низька швидкодія
Хороша підтримка зі сторони розробників	Відсутність мобільної оптимізації
Інтеграції з сайтами з пошуку роботи	Висока ціна
Зручний процес публікації вакансій	
Ефективний контроль виконання завдань	

У таблиці 1.1 ми бачемо, що плюси в даній системі переважають, тому її можна використовувати.

1.2 Система Sage

1.2.1 Можливості та особливостні функціоналу системи

Система управління людськими ресурсами Sage. Являє собою локальне рішення для управління людськими ресурсами, яке допоможе компаніям максимізувати кожну суму, яка інвестується в співробітників.

1.2.2 Програмне забезпечення для автоматизованого підбору кадрів

Набір та адаптація, мабуть, є найважливішою роллю в кадровому спектрі. Модулі набору та адаптації, доступні як доповнення до системи Sage, автоматизовуючи багато частин процесів набору, найму та адаптації. Програмне забезпечення для підбору персоналу, яке допомагає досягати успіху в пошуку нових працівників. Відділи кадрів продовжують боротися за

створення найкращої робочого простору, тому що успіх будь-якої організації залежить від її людей. Відмінні процеси набору та адаптації не відбуваються природно, вони потребують важкої роботи та правильної технології. Але переваги того варті[1].

З цією програмою нам відкриваються наступні можливості:

- Зменшення плинності кадрів, залучаючи та наймаючи найкращих талантів;
- Максимізувати рентабельність інвестицій вашої компанії;
- Доступ до співпраці з рекрутерами та менеджерами з найму для набору кандидатів, набору нових працівників та створення стратегічної робочої сили;
- Можливість зробити співробітників продуктивними та задоволеними, а також заощадити свою компанію на витратах, пов’язаних із плинністю кадрів;

Аналіз «що, якщо» за допомогою організаційних діаграм

Використання Sage, допоможе керівникам та співробітникам краще зрозуміти стратегію та структуру компанії, а також роль кожного в досягненні організаційних цілей.

Ефективне закриття відкритих позицій

За допомогою програмного забезпечення Sage ви можете швидше та ефективніше заповнювати відкриті вакансії. Цей власний інструмент, на 100% доступний в Інтернеті, легко налаштовується, безпаперовий і орієнтований на процес[1].

Створення безпаперової веб-форми

Спрощення збору і затвердження даних про співробітників за допомогою Інтернету. За допомогою Sage ви можете легко створювати безпаперові форми, використовуючи будь-які поля для отримання необхідних данних від працівника. Цим самим зменшаючу в рази паперову метушню в компанії та створюючи собою єдиний простір для ведення онлайн документів компанії.

Полегшення повторного введення даних

Дані співробітників часто дублюються в системі управління персоналом і сховищі Microsoft Active Directory. Надлишкова інформація для відділу кадрів та відділу розробки, щоб більше не було розбіжностей чи марних зусиль. Адже коли дані вже зберігаються в одній системі, не є проблемою за потреби їх отримати та внести у необхідні поля іншої програми, що в рази полегшує введення нових інструментів для роботи в компанії та покращує адаптацію персоналу до них[1].

1.2.3 Програмне забезпечення адміністрування пільгових співробітників

Управління виплатами працівникам стомлює і вимагає багато паперів. За допомогою служб управління виплатами співробітників — розширення для Sage HRMS — ви можете автоматизувати, оптимізувати та надати своїм співробітникам можливість використовувати інструменти самообслуговування. Гнучке програмне забезпечення допомагає гарантувати, що технологія розроблена відповідно до ваших конкретних вимог. У Sage є три основні послуги, на які може розраховувати ваш відділ кадрів.

Автоматизація кар'єрних переваг для пільгових груп працівників

Зменшення витрат та складність адміністрування пільг і безпечного автоматизування передачи даних про реєстрацію пільг співробітникам до страхових компаній за допомогою Sage HRMS Benefits Messenger. Це потужне рішення позбавить від необхідності подавати паперові реєстраційні форми або створювати та підтримувати індивідуальні електронні формати файлів.

Керування відкритою реєстрацією та життєвими подіями

Можливість перенести відкриту реєстрацію в Інтернет середовище і спроможність надати своїм співробітникам можливість вибирати пільги за допомогою Sage. Прості покрокові навчальні посібники проведуть вас через налаштування та допоможуть співробітникам пройти відкриту реєстрацію. Надання щедрого та конкурентоспроможного пакету пільг – це один із способів залучення та утримання талановитих працівників, а також суттєвий фактор зниження плинності робочої сили[1].

Відстеження та аналіз інформації про компанію

За допомогою аналізатору робочого простору, щоб відстежувати й аналізувати інформацію про вашу компанію, щоб ви могли краще керувати та приймати зважені рішення щодо медичних послуг та вимог ACA. Аналізатор робочого простору допомагає вам керувати медичними послугами, які спонсорує роботодавець, і дотримуватися державних правил.

1.2.4 Керування зарплатою в компанії

Sage пропонуціє цілий робочий модуль для управління та обробки заробітної плати працівників, для створення точної та своєчасної розрахунку заробітної плати всередині компанії, днів заробітної плати. Часовий модуль системи Sage усуває традиційний відлік часу та запроваджує самообслуговування менеджерів і співробітників, поставляється в хмарі як

автономне рішення і підключається до інших моделів системи Sage для нарахування заробітної плати.

Індивідуальний спосіб виплат

Можливість надати фінансовій команді потужний інструмент для повної, своєчасної обробки заробітної плати. І оскільки він настільки гнучкий, ви можете налаштувати його, щоб обробляти вашу зарплату по-своєму за допомогою програмного модуля для виплат заробітних плат.

Відстеження часу відпочинку та неявкок на роботу

Керування всіма типами відпусток, у тому числі відпустками на основі інцидентів, такими як обов’язки присяжних, медичні відпустки та втрати, і призначайте регулярні відпустки, керуйте та звітуйте про відпустки, відстежуйте медичну сертифікацію та повторну сертифікацію дати.

1.2.5 Створення власних зарплатних квитанцій

Подання звітів для сплати податків в кінці календарного року

Захист компанії від помилок у податковій декларації та дотримання усіх державних і федеральних вимог щодо звітності та платежів за допомогою податкових форм.

Оптимізація обчислень

Звільніть свою команду з нарахування заробітної плати від обчислення та визначення пріоритетів складних гарнірів. З Sage HRMS Garnishment Manager від Delphia Consulting легко розрахувати суми відрахувань.

1.2.6 Тестування та користувацький досвід

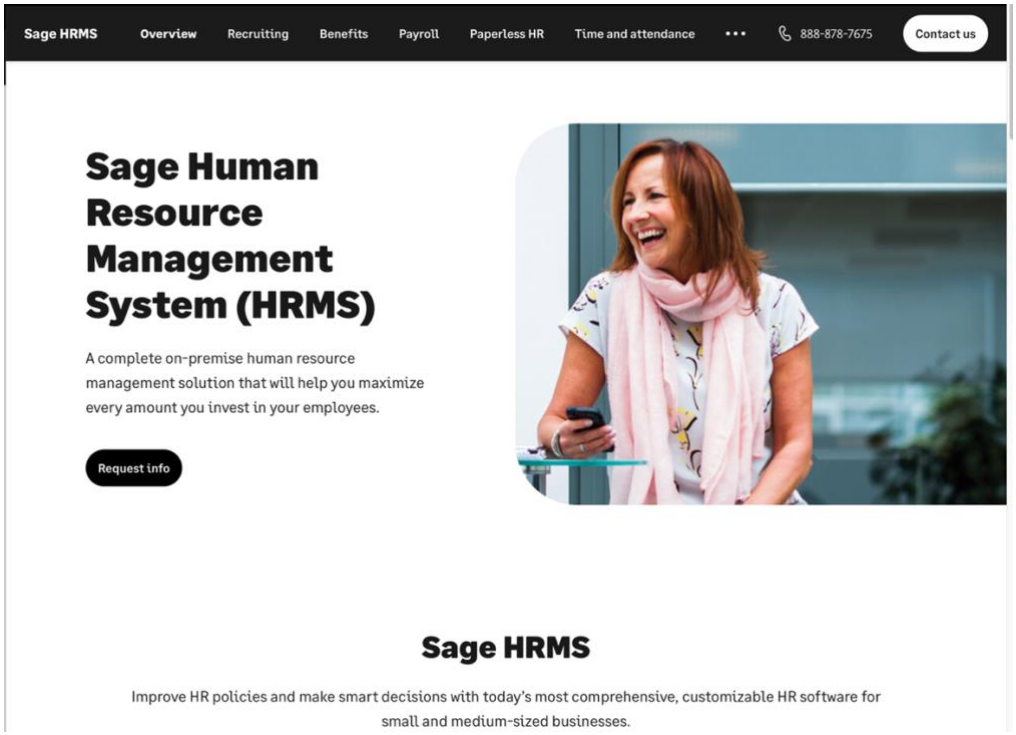


Рисунок 1.4 - Головна сторінка сайту системи Sage

Перше знайомство користувача зданою системою відбувається з сайту компанії див. рис. 1.4, на якому знаходиться детальна інформація про ресурс та його використання. Пункти мені добре скомпоновані та чітко відображаються логіку користування сайтом. Проте немає пункта з цінами за користування послугами цієї системи. Поганим досвідом є те, що для реєстрації акаунта та початку користування необхідно залишати заявку з персональними данним та очікувати на дзвінок або повідомлення від менеджера компанії. Набагато краще було б створити окрему сторінку для входу в додаток яка переспрямовувала користовача на урл адрессу “name.com/register”. На сторінці реєстрації повинна бути форма з полями для вводу данних, заповнивши які повинен прийти лист підвертдження на електронну адресу користувача. Бокове зображення не є інформативним, краще на його місце вставити скриншот роботи застосутку.

1.2.7 Продуктивність та показники системи

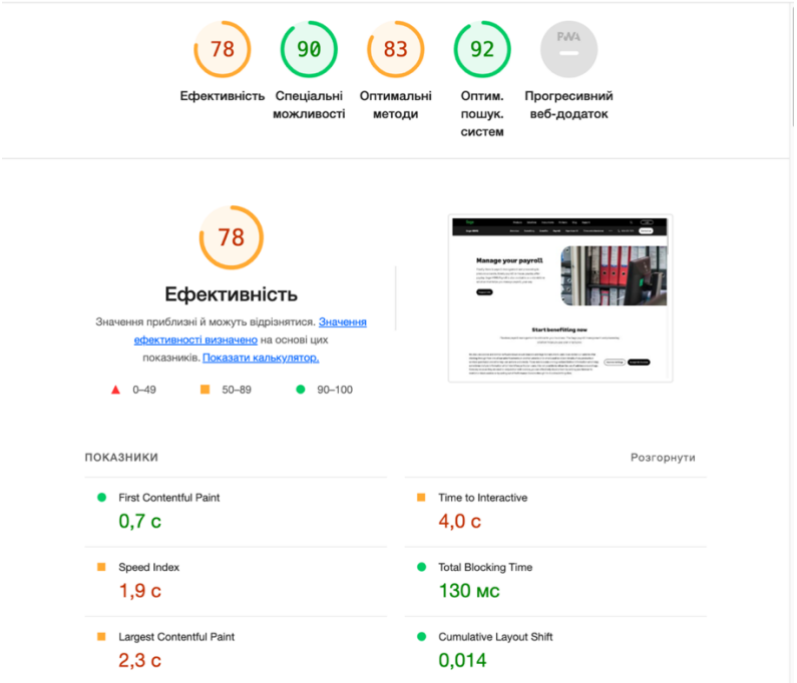


Рисунок 1.5 - Результати тестів системи Sage в Lighthouse

Використовуючи застосунок для аналітики та оптимізації швидкості роботи сайту від компанії “Google” лайфтхауз, я протестував роботу цієї системи.

Швидкість роботи є досить хорошою, що задовільняє повністю потребу користувачів в швидкодії, не змушуючи чекати відкриття сайту занадто довго

Дуже хорошими є спеціальні можливості, ці перевірки визначають можливості для покращення доступності користувачів да веб-додатку.

Оптимізація для пошукових систем є чудовою, сайт досконально відображається в органічному пошуку гугл. Зображення добре оптимізовані по розміру та мають альтернативний опис для пошукових систем.

1.2.8 Інтерфейс

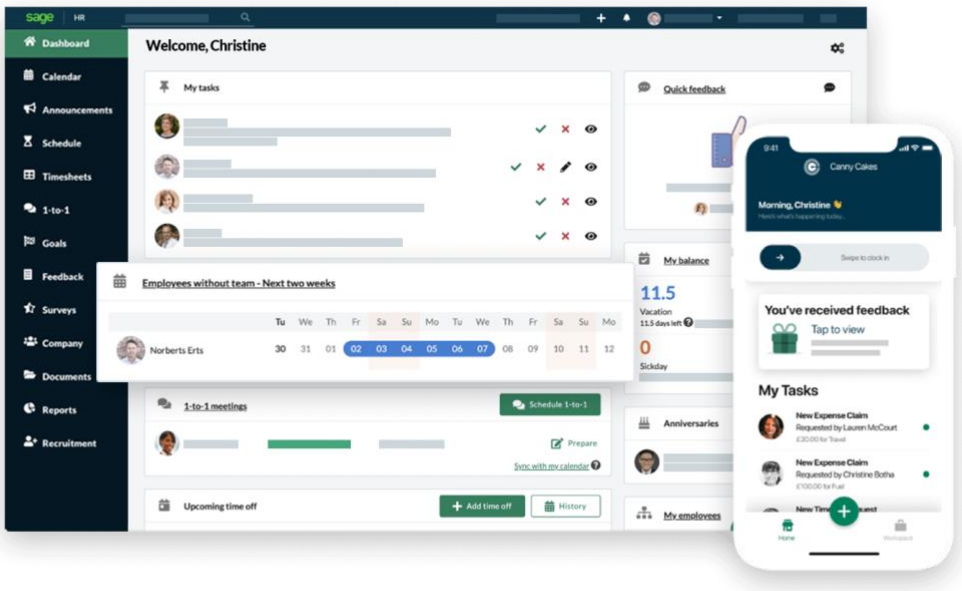


Рисунок 1.6 - Інтерфейс системи Sage

Зображено на рис 1.6 ми бачимо головну сторінку системи, яка являє собою список з задач. Відразу варто відмітити чудову адаптацію системи на мобільних пристроях. Списки меню добре скомпоновані та знаходяться в лівій частині екрану, за потреби ви ховаються. Інтерфейс сучасний зроблений з прорахуванням усіх необхідних деталей.

Таблиця 1.2. – Переваги та недоліки системи Sage

Плюси	**Мінуси**
База знань на головному сайті	Висока ціна
Можливість інтеграції	
Адаптація під різні пристрої	
База знань на головному сайті	

Отож, можна сказати, що в системи Sage майже відсутні мінуси і вона є досить хорошою, для використання в компаніях

1.3 Порівняльна таблиця

Таблиця 1.3. – Переваги та недоліки систем для управління персоналом Acsender та Sage

Функціонал	**Acsender**	**Sage**
Зарплатний модуль	-	+
Можливість підбору персоналу	+	+
Управління продуктивністю	+	+
Мобільна адаптація	-	+
База знань	+	+
Інтеграції з іншими сервісами	-	+
Підтримка	+	+
Хороший дизайн інтерфейсу	-	+

Після фінального огляду обох систем з певністю можна сказати, що система Sage має більше перевах над системою Acsender. Відповідно при розробці системи більшу увагу буде надато саме їй.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі було розглянуто системи для управління персоналом Ascender та Sage. Кожну з систем було розглянуто у декількох варіантах використання та виділено плюси та недоліки кожної.

Окреме місце в порівнянні мав користувацький інтерфейс систем, адже сьогодні – це одний з найголовніших факторів будь-якої програми, так як зручність користування напряму впливає на зацікавлення користувачів у ній. В результаті дослідження було виявлено поганий інтерфейс в обох програм, який не відповідає сучасним стандартам веб-дизайну, та являється головною їх проблемою.

Швидкодія є не менш головним факктором і відіграє важливу роль в користуванні. Кожну систему було протестовано на продуктивність за допомогую інструментів для розробників компанії Google. Lighthouse дослідив ефективність і вона була поганою. Це пов’язано через застарілі технології, які використовувались для програмування.

Головний підрозділ при аналізу систем був призначений їх функціоналу, тому що це є найголовнішим фактором користувачів при виборі системи для користування нею в робочих цілях. Тут варто зазначити, що можливості є досить великими, що дозволяє користуватись системою для великого спектру компаній та виконувати за вдяки ним різного характеру робочі завдання.

Таким чином, актуальною є задача розробки HRCRM системи з управління персоналом, яка дозволить покращити продуктивність працівників в коимпанії. За основу майбутньої системи буде прийнятий до уваги функціонал розглянутих систем.

РОЗДІЛ 2 ОГЛЯД АЛГОРИТМІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ

2.1 Прогресивні веб-програми

2.1.1 Огляд PWA

Простіше кажучи, прогресивний веб-додаток — це суміш мобільного та веб-додатка!

Рисунок 2.1 - Структура сучасних веб-програм

Але що це саме означає? І чим вони відрізняються від традиційних веб-програм і рідних програм?

- PWA є прогресивними — ці програми працюють для кожного користувача, незалежно від того, який браузер ви використовуєте або навіть де ви перебуваєте у світі! Тож, чи використовуєте ви Chrome чи Opera, чи живете ви в Україні, Франції чи навіть Японії, не має значення! Прогресивні веб-програми працюватимуть так само добре, оскільки вони створені з прогресивним удосконаленням як основним принципом.[5]

- Оптимізація відображення — прогресивні веб-програми підійдуть на будь-який пристрій! Будь то настільний комп’ютер, мобільний телефон, планшет або навіть щось, що ще не створено!
- Веб-програми, які не залежать від підключення — за допомогою сервіс воркерів прогресивні веб-програми можуть працювати навіть у низькоякісних мережах або навіть офлайн[5].
- Застосункова поведінка — прогресивні веб-програми відчувають себе як звичайні додатоки. Вони мають взаємодію та навігацію в стилі програми.
- Завжди актуальні — завдяки сервіс воркерам ваша прогремивний веб-програма завжди буде в курсі останніх оновлень.
- Безпечні — прогресивні веб-програми завжди обслуговуються через HTTPS, що гарантує, що ніхто без належної авторизації не зможе змінити вашу програму.
- Відкриті — відповідно до маніфесту W3C, прогресивні веб-програми класифікуються як програми. Прогресивну веб-програму легше знайти завдяки області реєстрації, що дозволяє пошуковим системам легко їх знаходити.
- Залучення користувачів — такі функції прогресивних веб-програм, як сплуваючі повідомлення, дуже полегшують залучення користувачів!
- Встановлення — користувачі можуть «зберігати» найкорисніші PWA на своєму головному екрані без клопоту з магазинами додатків.
- Можливість зв’язування — PWA можна легко надати за допомогою URL-адреси та не вимагати складної інсталяції.

2.1.2 Можливості виявлення в PWA

За даними Comscore Mobile App Report, понад 50% користувачів смартфонів Америки завантажують нуль програм на місяць! Що це означає, що час, коли наш телефон був заповнений додатками, повільно минає! За

статистикою, кожен крок із завантаження програми зменшує приблизно 20% потенційної кількості користувачів програми. Прогресивні веб-програми ефективно скоротили кроки між виявленням програми та відображенням її на головному екрані, тим самим усуваючи сумніви щодо встановлення програми на вашому пристрої. Ще одна перевага для прогресивних веб-програм полягає в тому, що власні програми доступні лише через Play Store/App Store. Причина успіху цих магазинів полягає в тому, що користувачі не мають іншого вибору, як вони можуть знаходити та встановлювати власні програми. Але коли справа доходить до веб-програм (як традиційних, так і прогресивних), не існує єдиного магазину! Тож ми можемо мати ретельно підібрані магазини додатків, які ідеально відповідають нашим потребам[11].

Загальний процес встановлення звичайних додатків виглядає приблизно так:

1. Відкрийте App Store і знайдіть програму.
2. Натисніть «Встановити».
3. Прийміть різні дозволи.
4. Відкрийте програму.
5. Запуск програми.
6. Увійдіть у додаток.
7. Приступайте до використання

Давайте порівняємо це з встановленням прогресивних веб-програм:

1. Відвідайте сайт.
2. Додати на головний екран.
3. Перейдіть на головний екран і відкрийте сайт.
4. Використовуйте додаток!

2.1.3 Маніфест прогресивної веб-програми

Маніфест — це простий файл JSON, який дає розробнику можливість керувати тим, як програма має виглядати користувачеві в тих областях, де вони очікують побачити програми (головний екран мобільного пристрою), керувати тим, що користувач може запускати, і визначати його зовнішній вигляд. при запуску[6].

Рисунок 2.2 - приклад файлу manifest.json програми PWA

2.1.4 Спливаючі сповіщення

Пуш-сповіщення – це повідомлення, яке з’являється на пристрої користувача. Спливаюче-повідомлення можуть ініціюватися локально відкритою програмою, або їх можна «пересилати» від сервера до користувача,

навіть коли програма не запущена. Push-сповіщення дозволяють користувачам підключатися до своєчасних оновлень, а також дають змогу ефективно повторно взаємодіяти з налаштованим вмістом[7].

Рисунок 2.3- Приклад спливаючого сповіщення

Push-повідомлення збираються за допомогою двох API: API сповіщень і Push API. API сповіщень дозволяє програмі відображати системні сповіщення користувачеві. API Push дозволяє працівнику служби обробляти Push-повідомлення із сервера, навіть коли програма не активна. API сповіщень і Push створено на основі API сервіс воркера, який реагує на події push-повідомлень у фоновому режимі та передає їх у вашу програму.

2.1.5 Сервіс воркери

Сервіс-воркери пропонують неймовірну корисність, але спочатку може бути складно працювати з ними. Workbox полегшує використання них. Однак, оскільки сервіс-воркери вирішують важкі проблеми, будь-яка абстракція цієї технології також буде складною без її розуміння.

Які можливості вони надають?

Сервіс-воркери — це спеціалізовані ресурси JavaScript, які діють як проксі між веб-браузерами та веб-серверами. Вони спрямовані на підвищення надійності, забезпечуючи офлайн-доступ, а також підвищуючи продуктивність сторінки.

Життєвий цикл, подібний до програми, що поступово розширюється

Сервіс воркери є доповненням до існуючих веб-сайтів. Це означає, що якщо користувачі веб-переглядачів, які не підтримують їх, відвідують веб-сайти, які їх використовують, базова функціональність не порушується. Сервісні працівники поступово покращують веб-сайти через життєвий цикл, подібний до програм для певної платформи[25]. Подумайте про те, що відбувається, коли нативну програму встановлено з магазину додатків:

- Зроблено запит на завантаження програми.
- Програма завантажується та встановлюється.
- Програма готова до використання та може бути запущена.
- Оновлення програми для нових версій.

Життєвий цикл сервіс-воркерів подібний, але з прогресивним підходом до покращення. Під час першого відвідування веб-сторінки, на якій встановлюється новий сервіс-воркер, початкове відвідування сторінки

забезпечує базову функціональність під час завантаження сервіс-воркера. Після встановлення та активації сервіс-воркера він контролює сторінку для підвищення надійності та швидкості.

2.1.6 Доступ доя API кешування за допомогою JavaScript

Незамінним аспектом технології сервіс-воркерів є інтерфейс кешу, який є механізмом кешування, повністю окремим від кешу HTTP. Доступ до інтерфейсу кешу можна отримати в межах області сервіс-воркерів і в межах основного потоку. Це відкриває безліч можливостей для керованої користувачем взаємодії з екземпляром кешу. У той час як на кеш HTTP впливають директиви кешування, зазначені в заголовках HTTP, інтерфейс кешу програмується через Javascript. Це означає, що кешування відповідей на мережеві запити може базуватися на будь-якій логіці, яка найкраще підходить для даного веб-сайту[11]. Наприклад:

- Збереження статичних активів в кеші під час першого запиту для них і обслуговування їх лише з кешу для кожного наступного запиту.
- Збереження розмітки сторінки в кеші, але показ розмітки з кешу лише в автономних сценаріях.
- Подавання застарілих відповіді для певних активів із кешу, але оновлювати його з мережі у фоновому режимі.
- Передача часткового вмісту із мережі та збір його за допомогою оболонки програми з кешу, щоб покращити продуктивність сприйняття.

Кожен із них є прикладом стратегії кешування. Стратегії кешування роблять можливим роботу в автономному режимі та можуть забезпечити кращу продуктивність, обмежуючи перевірку з високою затримкою, що перевіряє кеш HTTP. Перш ніж зануритися в Workbox, ми розглянемо кілька стратегій кешування та код, який змушує їх працювати.

2.1.7 Попереднє кешування під час виконання

Взаємодія між сервіс-воркером і екземпляром кешу включає дві різні концепції кешування: попереднє кешування та кешування під час виконання. Кожне з них є центральним для переваг, які може надати обслуговуючий працівник. Попереднє кешування — це процес завчасного кешування активів, як правило, під час встановлення сервісним працівником. За допомогою попереднього кешування ключові статичні активи та матеріали, необхідні для офлайн-доступу, можна завантажувати та зберігати в екземплярі кешу. Такий вид кешування також покращує швидкість сторінки для наступних сторінок, які потребують попередньо кешованих активів. Кешування під час виконання — це коли стратегія кешування застосовується до активів, оскільки вони запитуються з мережі під час виконання. Такий вид кешування корисний, оскільки гарантує офлайн-доступ до сторінок і ресурсів, які користувач уже відвідав. У поєднанні ці підходи до використання інтерфейсу кешу в сервіс-воркері забезпечують величезну перевагу для користувацького досвіду та забезпечують поведінку, подібну до програми, для звичайних веб-сторінок.

2.2 Мікросервіси

Мікросервіси – це архітектура, в якій створюються і розміщуються різні компоненти програмного забезпечення як окремі ізольовані служби. Кожен з них розгортається окремо, і вони спілкуються через чітко визначені мережеві інтерфейси. Мікросервіси призначені як «маленькі» (нечітко визначені) і зберігаються в єдиному обмеженому контексті. Мікросервіси мають багато переваг. Через свою ізольованість і сувору вимогу до спілкування через чітко визначені інтерфейси мікросервіси запобігають швидким і брудним рішенням, які часто зустрічаються в монолітах. Ці хаки всередині моноліту призводять до втрати зчеплення та збільшення зв’язку — двох основних причин складності.

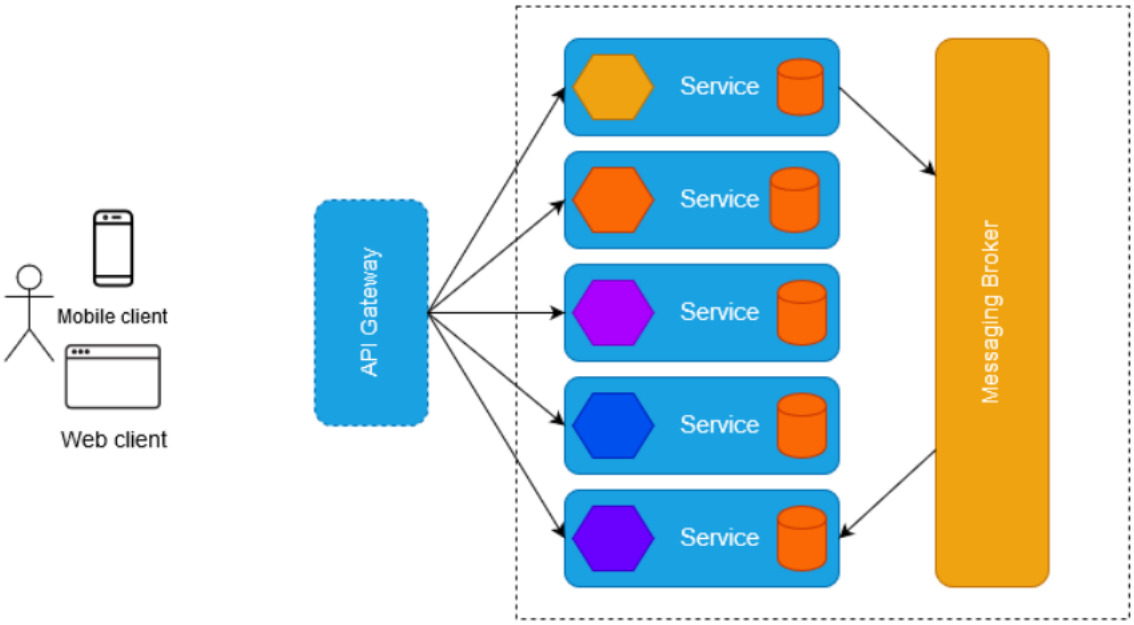


Рисунок 2.4 - Модель архітектури мікросервісів

Багато хто стверджує, що ви можете зберегти цю поведінку в моноліті. Насправді, оскільки це легко і тому що надто мало архітекторів, які працюють у наших базах коду, моноліти зазвичай падають через цю саму невдачу. Складність виникає через низьку когезію та високу зчеплення. Мікросервіси надають структуру, щоб уникнути цього. Цю перевагу неможливо переоцінити. Оскільки ми тримаємо монстра складності в страху, розробка систем десятирічної давності може продовжувати розвиватися зі швидкістю розробки, коли система була абсолютно новою. Знову й знову складність, викликана слабкою згуртованістю та тісним зв'язком, була причиною повільного розвитку старих проектів. Згуртованість і зв'язок – це традиційно технічний борг, який бере на нас ноги, сповільнюючи нас. Назбирайте його з роками, і ви будете боротися з цим. Якщо послуги написані з урахуванням них, а інфраструктура надає це, інші переваги можуть включати горизонтальну масштабованість, можливість тестування, надійність, спостережливість, замінюваність і незалежність від мови. Недоліком мікросервісів є те, що для досягнення цих переваг ви повинні забезпечити базову інфраструктуру, яка їх підтримує. Без

цієї підтримки ви можете легко опинитися з ненадійною та непрозорою системою — або ви зможете заново винаходити колесо надійності в кожній окремій службі. Це чудовий підхід до наступного розділу…

2.2.1 Вимоги високого рівня

Середовище, яке підтримує мікросервіси, потребує набору базових вимог, щоб забезпечити певний рівень розумності. Якщо ви збираєтеся запускати мікросервіси, ваша організація має бути готова нести накладні витрати на їх запуск та підтримку. Накладні витрати не будуть незначними. Для якісного виконання мікросервісів знадобиться час і гроші. Успішна архітектура мікросервісів повинна мати внутрішній комітет або групу, відповідальну за визначення макроархітектури — це визначатиме, яка інфраструктура буде надаватися для розробки та функціонування мікросервісів разом із політикою, якої повинні дотримуватися всі мікросервіси. Цей комітет має бути найсильнішим із ваших розробників, і це може бути навіть один або кілька людей, які ще навіть не працюють на вас. Макроархітектура - це одна частина наданої інфраструктури і одна частина вимог політики для всіх мікросервісів. Макроархітектура кожної організації буде унікальною. Кожна область, наведена нижче, повністю відкрита для переговорів щодо того, де провести лінію для вашої групи: ви можете надати командам фіксовану послугу або бібліотеку коду для забезпечення необхідної функціональності. Ви можете або дозволити його використання, або зробити його необов’язковим. Ви можете просто вказати критерії прийнятності, яким має відповідати мікросервіс, але не надати реалізовану бібліотеку чи службу, щоб допомогти виконати вимогу. Нарешті, ви можете вибрати, чи нічого не вимагати для будь-якої категорії. Розумно вибирайте те, що не виключаєте зі своєї макроархітектури[26]. Для кожного вибору, який ви дозволяєте окремим командам розробників, ви повинні бути готові жити з різними рішеннями, впровадженнями та

операційною поведінкою. Ви — комітет, і завжди найкраще, коли люди в організації приймають такі рішення — тому я не можу надати вам запечений маніфест. На початку роботи також важливо тримати цю документацію з макроархітектури відкритою для змін і сприйнятливою до потреб команд і бізнесу. Тепер звернемося до різних категорій, для яких необхідно приймати рішення щодо макроархітектури[9].

2.2.2 Безперервна інтеграція

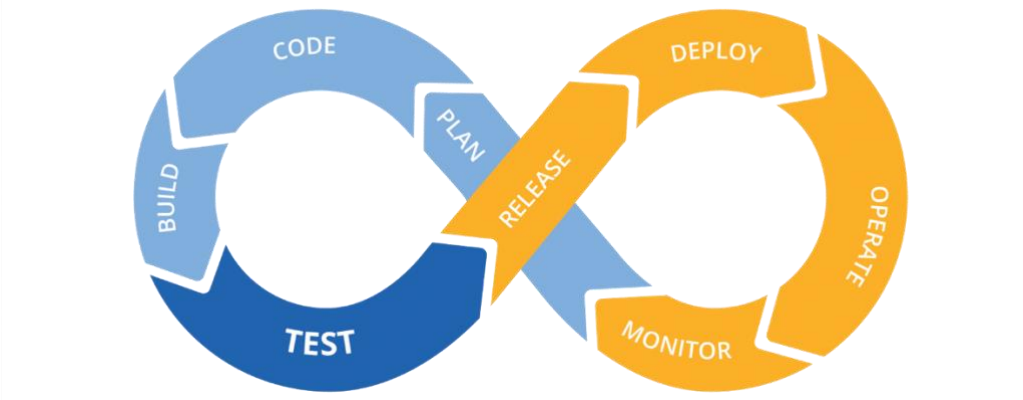


Рисунок 2.5 – План безперевної інтеграції

Основою концепції мікросервісів є здатність створювати та виконувати тести дуже швидко. Кожна фіксація мікросервісу має призвести до перевіреної збірки. Після того, як тести пройдуть і система збирання задоволена, наступним важливим аспектом є кнопка або автоматичне розгортання у виробництво. Скорочення часу на розгортання дозволяє швидко виконувати ітерації та дає змогу використовувати будь-яку кількість хороших практик кодування. Це легко виконати в наші дні. Існує будь-яка кількість систем збірки, які надають доступ до конвеєрних збірок. Команда Сіті. бамбук. Дженкінс з Блакитним океаном. Спробуйте їх і виберіть один. Здебільшого набори функцій є досить стандартними для лідерів пакету. Організація повинна прагнути до послідовності в тому, як створюються та розгортаються послуги. Отже, макроархітектура повинна визначати інструмент побудови та процеси

конвеєра. Команди повинні мати право голосу в розмові, що веде до вибору, але не можна допускати, щоб у цій бесіді було шахрайство[24].

2.2.3 Авторизація

Важливо стежити за своїми мікросервісами у виробництві. Щоб зробити це ефективно, вам потрібно ввімкнути швидке розташування різної інформації. Це означає, що макроархітектура повинна ретельно розглянути, включаючи наступне: Сервіс для централізованого ведення лісозаготівель. Це може бути Elastic Stack, Slack, Graylog та інші подібні. Вам потрібен стек журналів, який містить потужний синтаксичний аналізатор/візуалізатор, тому що ви збираєтеся мати справу з купою даних. Частиною вашої інфраструктури може бути одна з цих служб і гарантія того, що кожен хост у середовищі буде налаштований на передачу файлів журналів від імені кожної служби. Визначення ідентифікаторів трасування, щоб дозволити розташування всіх журналів у всіх мікросервісах, які обробляють один зовнішній запит. Концепція полягає в тому, що для кожного зовнішнього запиту, що надходить у ваші мікросервіси, ви створюєте унікальний ідентифікатор, і цей ідентифікатор передається будь-яким внутрішнім викликам мікросервісів, які використовуються для обробки цього запиту. Таким чином, за допомогою пошуку єдиного ідентифікатора трасування ви можете знайти всі виклики мікросервісів, отримані в результаті одного зовнішнього доступу. Базові вимоги до форматування для ідентифікаторів сервера, служби, екземпляра, позначки часу та відстеження.

2.2.4 Моніторинг

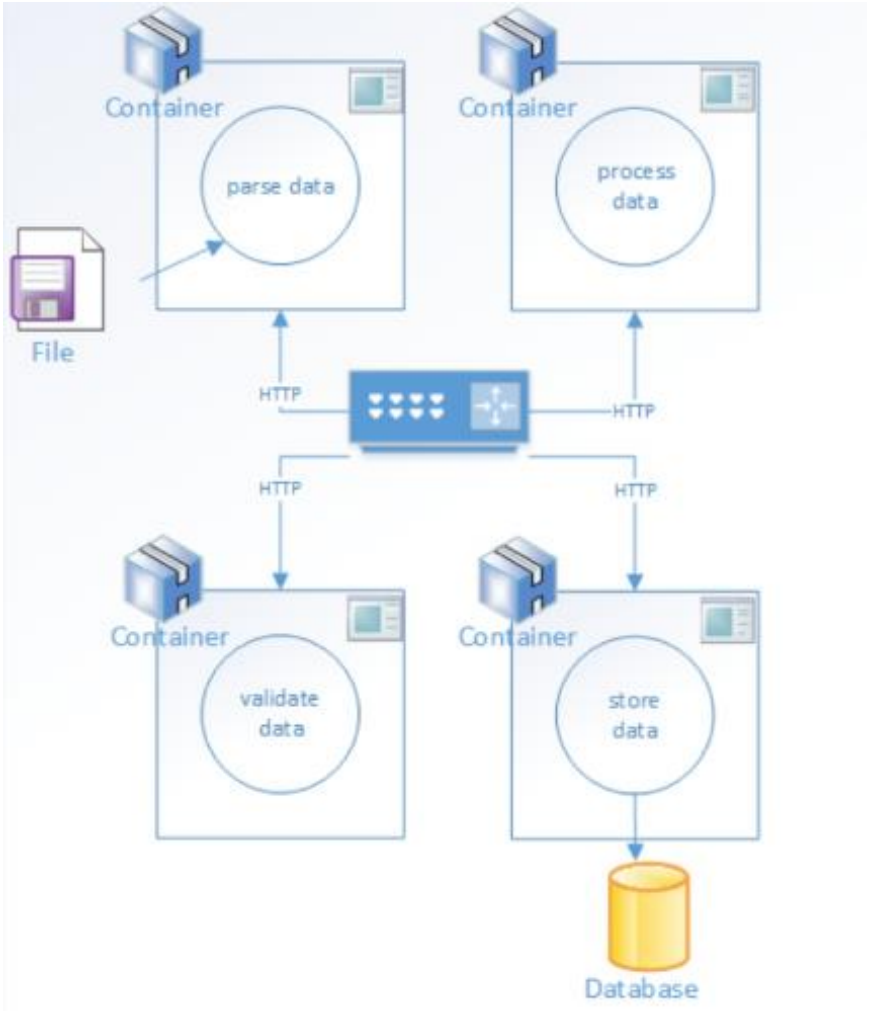


Рисунок 2.6 – Модель моніторингу в мікросервісах

Це ще одне «повинне забезпечити» для макроархітектури. Кожному з мікросервісів потрібно буде визначити найкращі показники для вимірювання та моніторингу, які забезпечать індивідуальний успіх, але макроархітектура матиме конкретні інструменти, які їй знадобляться для кожної служби, щоб забезпечити нагляд за загальним працездатністю системи. Деякі точки даних на макрорівні включають: Обсяг повідомлень, невдач, успіхів, повторних спроб і падіння. Затримка запитів. Співвідношення отриманих повідомлень до надісланих[31]. Стан вимикачів. І більше. Набагато, набагато більше. Інструментарій — це одна з областей, де «Дао мікросервісів» справді сяє, і я настійно рекомендую його для хорошого розуміння широти та глибини

моніторингу в мікросервісах. Реєстрація та місцезнаходження служби Це часто ігнорується, коли архітектура мікросервісів невелика, оскільки кілька мікросервісів завжди можуть знайти один одного відносно легко[30]. Проте з часом і збільшенням кількості мікросервісів конфігурація, необхідна для статичного з'єднання всіх разом, стає занадто обмеженою і в кінцевому підсумку схильною до помилок. Можна мати багато рішень, включаючи DNS та служби конфігурації (і т. д.) Макроархітектура вашого середовища мікросервісів повинна визначати, як це робиться — навіть якщо перша ітерація /etc/services.yaml розгорнута та синхронізована з усіма хостами. Це не те, що повинні встановлювати групи розробників окремих служб — це має бути повсюдно й керуватися на рівні макроархітектури. Існує багато існуючих проектів із відкритим кодом, які намагаються вирішити цю проблему, включаючи деякі сервісні сітки[28].

2.2.5 Комунікаційні механізми

Мікросервіси повинні мати певний рівень контролю над тим, як вони реалізують свої інтерфейси. Як протокол мережевого рівня, так і протокол прикладного рівня повинні забезпечувати певний рівень гнучкості. ВикористанняБуфери протоколу Google через необроблений TCP можуть бути так само доступні, як і використання JSON RPC через HTTPS. Тим не менш, макроархітектура повинна надавати певні вказівки, деякі обмеження і, можливо, навіть деяку інфраструктуру, щоб полегшити комунікацію. Якщо інфраструктура мікросервісів буде працювати разом у спільному просторі доменних імен за URI HTTPS, вам знадобиться стандартизація щодо іменування та маршрутизації. Запити повинні мати загальний і послідовний метод, за допомогою якого вхідні запити користувачів, а також запити між послугами аутентифікуються, авторизуються та маршрутизуються. Інфраструктура мікросервісів, яка хоче дозволити використання обміну

повідомленнями як комунікаційного пристрою, повинна розглянути можливість надання керованої операційною шиною обміну повідомленнями. Це дає змогу швидко розробляти й розгортати служби без того, щоб командам спочатку потрібно було зосередитися на запуску, а потім на довгостроковому управлінні службою обміну повідомленнями[27].

2.2.6 Безпека

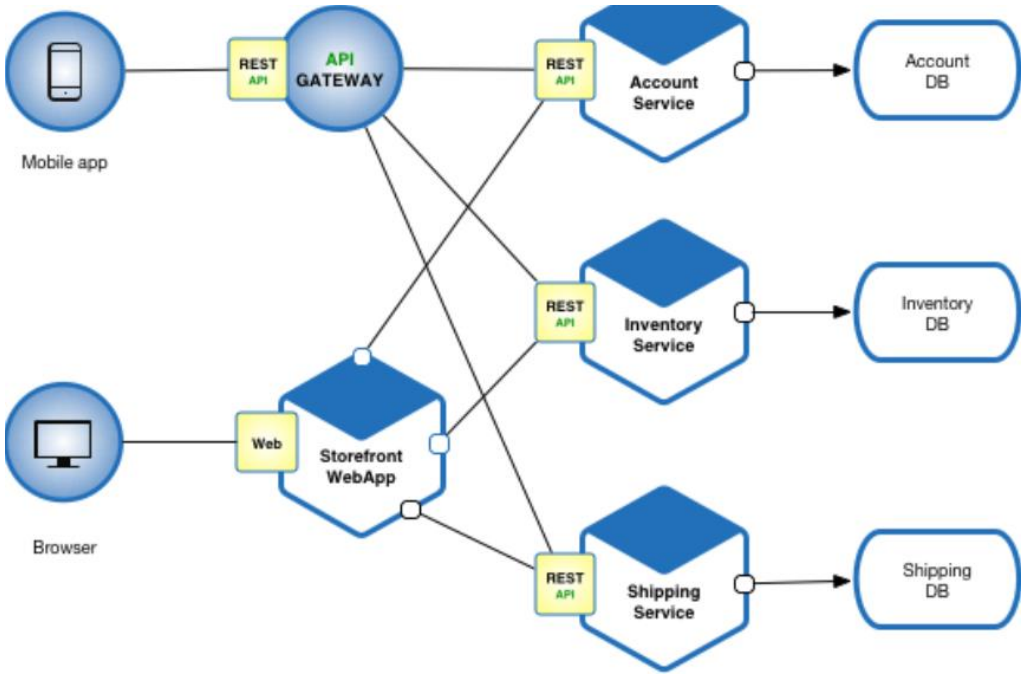


Рисунок 2.7 – Модель ізольованості шарів в мікросервісах

Ваші служби повинні знати, з ким вони спілкуються (аутентифікація) і які дані та операції дозволені (авторизація) для зазначеної особи. Тут є кілька потенційних концепцій: Дозвольте IP-мережі захищати послуги — якщо ви запускаєте всі свої мікросервіси в захищеній мережі і хочете передати довіру своїм розробникам, щоб вони не зловживали доступом, це може спрацювати для вас. Майте на увазі, що порушення однієї служби передбачає повний доступ до всіх інших служб[23]. Аутентифікація на рівні сервісу — спільні ключі або аутентифікація на основі сертифікатів дають змогу службі, що викликається, перевіряти службу, що викликає. Щоб забезпечити безпеку, вам знадобиться безпечний спосіб розповсюдження та оновлення ключів і сертифікатів[17].

Використовуйте службу керування ключами. Аутентифікація на рівні користувача — не тільки служби спілкуються зі службами, але вони досить часто спілкуються від імені користувача або навіть безпосередньо з ним. Повинні бути засоби аутентифікації та авторизації облікових даних на рівні користувача для ресурсу. Почніть з простого — це область, яка може вийти з воріт організації, і, мабуть, найкраще почати з простого. Ймовірно, у вас уже є кілька різних служб, які спілкуються один з одним, і ви, ймовірно, використовуєте деякі списки контролю доступу IP, щоб їх захистити. Почніть з простого, додайте складність у міру природної еволюції системи[8].

2.2.7 Балансування навантаження та стійкість

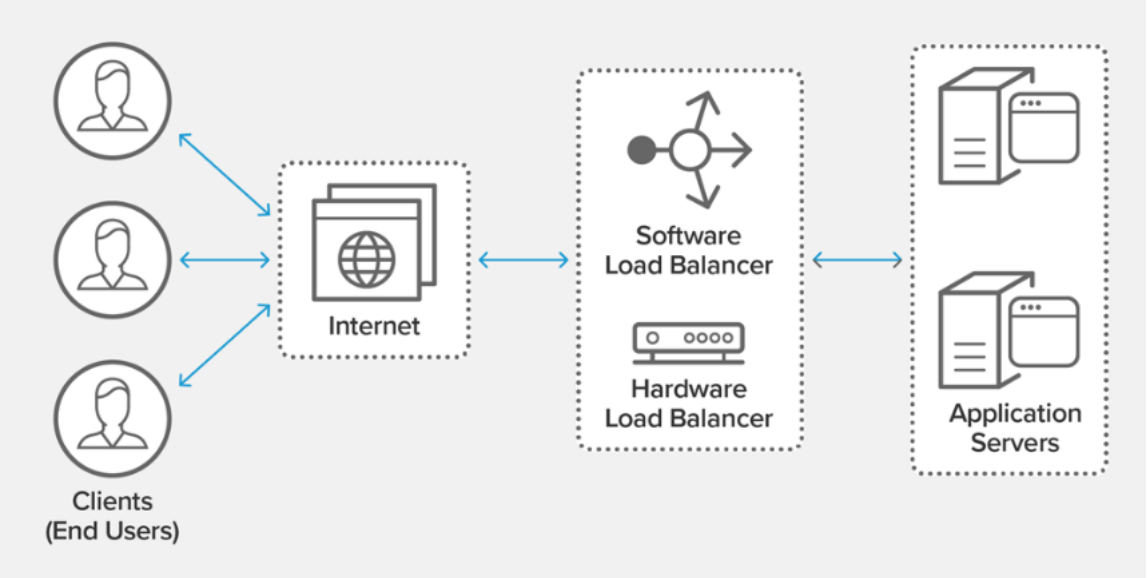


Рисунок 2.8 – Модель використання балансувальника навантаження

Мікросервіси часто використовуються в середовищах, де очікується масштабування та доступність. Традиційно мережеві пристрої забезпечують функціональність балансування навантаження. Але в середовищі мікросервісів більш типово бачити, як це переміщується на програмний рівень інфраструктури макроархітектури. Код, за допомогою якого послуги зв’язуються, може використовувати розташування служби для виявлення всіх

мережевих розташування даної служби, а потім може безпосередньо виконувати логіку балансування навантаження прямо на розподіленому краю.

Стійкість означає залишатися стабільним навіть перед обличчям помилок. Повторні спроби, кінцеві терміни, поведінка за замовчуванням, поведінка кешування та черги — це лише деякі з способів, якими мікросервіси забезпечують стійкість. Так само, як і балансування навантаження, деяка частина стійкості ідеально підходить для інфраструктури, яку обробляє на периферії — наприклад, повторні спроби та розриви ланцюга (автоматичні реакції на помилки для служб, які перевищують поріг збою в недавньому минулому). Однак окремій службі слід розглянути, яку роль стійкості вона повинна відігравати всередині себе. Наприклад, система реєстрації облікового запису, де втрата реєстрації прирівнюється до втрати грошей, повинна брати на себе відповідальність за те, щоб кожна реєстрація проходила, навіть якщо це означає відкладене створення, що призводить до отримання електронного листа власнику облікового запису після успішного завершення. Внутрішню чергу та керування незавершеними реєстраціями найкраще керувати безпосередньо цією критичною службою[3].

2.2.8 Де та як варто використовувати мікросервіси

Реальність полягає в тому, що чим менша ваша організація, тим менші ваші потреби в повноцінній архітектурі мікросервісів. Однак немає причин відмовлятися від усього руху і залишати позаду переваги, які усвідомили ці дуже розумні люди, навіть якщо ви невеликий магазин з невеликими послугами[19]. Ваші початкові розмови про макроархітектуру мікросервісів мають зосередитися саме на тому, що вам потрібно для початку, а потім з’ясувати, як це зробити. Створюйте деякі служби, спостерігайте за їх поведінкою та вчіться на основі того, що працює, а що не працює. Знову скликайте свій комітет макроархітектури мікросервісів і використовуйте свій

новий досвід разом із здоровим читанням і зростаючим розумінням галузевої екосистеми, щоб визначити, якою має бути наступна еволюція вашої макроархітектури.

2.3 Express.js

Express.js — це безкоштовна платформа веб-додатків з відкритим вихідним кодом для Node.js. Він використовується для швидкого та легкого проектування та створення веб-додатків. Веб-програми — це веб-програми, які можна запускати у веб-браузері. Оскільки для Express.js потрібен лише javascript, програмістам і розробникам стає легше створювати веб-додатки та API без будь-яких зусиль. Express.js — це фреймворк Node.js, що означає, що більшість коду вже написана для роботи програмістами. Ви можете створювати односторінкові, багатосторінкові або гібридні веб-програми за допомогою Express.js. Express.js є легким і допомагає організувати веб-додатки на стороні сервера в більш організовану архітектуру MVC. Важливо вивчити javascript і HTML, щоб мати можливість використовувати Express.js. Express.js полегшує керування веб-додатками. Це частина технології на основі javascript, яка називається програмним стеком MEAN, що означає MongoDB, ExpressJS, AngularJS і Node.js. Express.js є внутрішньою частиною MEAN і керує маршрутизацією, сеансами, HTTP-запитами, обробкою помилок тощо. Бібліотека JavaScript Express.js допомагає програмістам створювати ефективні та швидкі веб-програми. Express.js покращує функціональність node.js. Насправді, якщо ви не використовуєте Express.js, вам доведеться виконати багато складного програмування, щоб створити ефективний API. Це полегшило програмування на node.js і надало багато додаткових функцій.

2.3.1 Чому варто використовувати Express.js

Express.js підтримує JavaScript, який є широко використовуваною мовою, яку дуже легко вивчити і яка також підтримується всюди. Тому, якщо ви вже знаєте JavaScript, то вам буде дуже легко програмувати за допомогою Express.js. За допомогою Express.js ви можете легко створювати різні види веб-додатків за короткий проміжок часу. Express.js забезпечує просту маршрутизацію для запитів, зроблених клієнтами[21]. Він також забезпечує проміжне програмне забезпечення, яке відповідає за прийняття рішень щодо надання правильних відповідей на запити, зроблені клієнтом. Без Express.js вам доведеться написати свій власний код для створення компонента маршрутизації, що займає багато часу та виснажливе завдання. Express.js пропонує програмістам простоту, гнучкість, ефективність, мінімалізм і масштабованість. Він також має перевагу потужної продуктивності, оскільки є основою Node.js. Node.js виконує всі виконання дуже швидко за допомогою циклу подій, що дозволяє уникнути будь-якої неефективності[18]. Потужна продуктивність Node.js і простота кодування за допомогою Express.js є найпопулярнішими функціями, які люблять розробники веб-додатків. Оскільки Express.js написаний на Javascript, ви можете створювати веб-сайти, веб-додатки або навіть мобільні програми, використовуючи його.

2.3.2 Де використовувати Express.js?

Найцінніший актив у будь-якій справі – це час. Крім того, багато програмістів змушені створювати ефективні веб-програми за короткий період часу. Але кодування веб-програм і їх тестування вимагає часу. Саме тут Express.js стає паличкою-виручалочкою для програмістів. Express.js може скоротити час кодування вдвічі і при цьому допомогти нам створювати ефективні веб-додатки. Це не тільки скорочує час, але й зменшує зусилля, необхідні для створення веб-програм за допомогою різних функцій. Іншою

причиною використання Express.js є JavaScript. Express.js дозволяє навіть новачкам увійти в сферу розробки веб-додатків, оскільки він підтримує JavaScript. Javascript дуже легко вивчити будь-кому, навіть якщо він не має жодних попередніх знань з інших мов. Тому Express.js дозволяє молодим талантам увійти в сферу розробки веб-додатків і досягти успіху. Node.js керується подіями і, таким чином, має можливість одночасно обробляти тисячі клієнтських запитів, що неможливо з PHP. У сучасному світі веб-додатки та послуги в режимі реального часу стають все популярнішими. Node.js розроблено виключно для підтримки реальних веб-додатків. Найпоширенішим прикладом веб-програми в режимі реального часу є чат. Він включає тисячі користувачів і взаємодію в реальному часі, яку легко підтримує Node.js. Ще одним активом будь-якого бізнесу є гроші. Важливо ефективно використовувати гроші для максимізації прибутку. Оскільки Express.js є безкоштовним веб-додатком з відкритим вихідним кодом, який надає багато чудових функцій, немає причин не використовувати його[29].

2.3.3 Особливості Express.js

1. Швидшка серверна розробка Express.js надає багато часто використовуваних функцій Node.js у вигляді функцій, які можна легко використовувати в будь-якому місці програми. Це знімає необхідність кодування на кілька годин і, таким чином, економить час.
2. Проміжне програмне забезпечення Проміжне програмне забезпечення — це частина програми, яка має доступ до бази даних, клієнтського запиту та інших проміжних програм. В основному він відповідає за систематичну організацію різних функцій Express.js[16].
3. Маршрутизація ExpressJS забезпечує високорозвинений механізм маршрутизації, який допомагає зберегти стан веб-сторінки за допомогою URL-адрес.

4. Шаблонування ExpressJS надає механізми шаблонів, які дозволяють розробникам створювати динамічний вміст на веб-сторінках, створюючи шаблони HTML на стороні сервера[15].
5. Налагодження Налагодження має вирішальне значення для успішної розробки веб-додатків. ExpressJS спрощує налагодження, надаючи механізм налагодження, який має можливість точно визначити частину веб-програми, яка має помилки[15].

2.3.4 Переваги Express.js

Express.js є популярним інструментом завдяки наступним перевагам:

- Легко навчитися, оскільки багато користувачів інтерфейсу вже знайомі з JavaScript. Тому їм не потрібно вивчати нову мову, щоб вивчити Express.js
- Це значно полегшує розробку бекенда для розробників інтерфейсу, які використовують Express.js
- Веб-розробник може використовувати JavaScript як єдину мову для розробки інтерфейсу та сервера. Розробнику не потрібно вивчати або використовувати будь-яку іншу мову для розробки на стороні сервера.
- Код JavaScript інтерпретується за допомогою Google V8 JavaScript Engine за допомогою Node.js. Таким чином, код реалізується швидко і легко в ефективний спосіб[30].
- Фреймворк Express.js дуже простий у налаштуванні та використанні відповідно до потреб.

2.4 Next.js

Однією з головних переваг вивчення Next.js є знання того, наскільки гнучкими ви можете стати у створенні та адаптації до онлайнової реальності.

Внутрішньо ми вважаємо, що це єдина найважливіша перевага в гіпер-зростаючому цифровому світі, оскільки ми можемо швидко спробувати і перевірити наші ідеї. Якщо нам це вдасться, ми зможемо легко додавати нові функції та реагувати на швидкі зміни набагато швидше, ніж будь-коли раніше, щоб залишатися конкурентоспроможними. Якщо ні, легше перебудувати всю стратегію та адаптуватися відповідно.

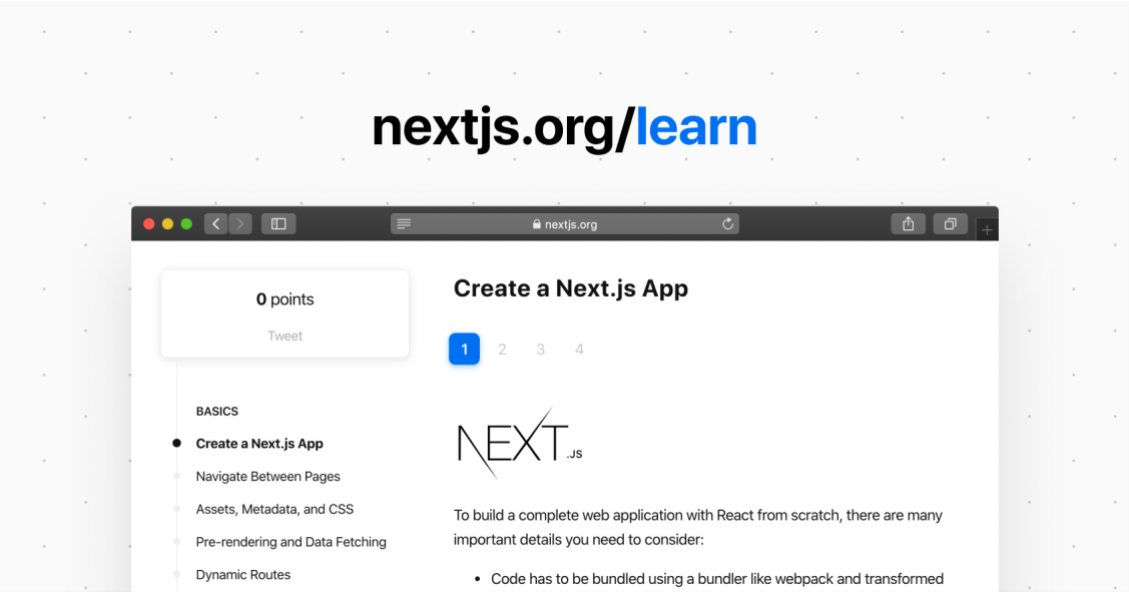


Рисунок 2.9 – Головна сторінка next.js

Ми стали набагато більш вимогливими до швидкості завантаження сторінок (у мілісекундах!) і досвіду користувачів від використання веб-сайтів чи веб-магазинів.

Це створює чудову можливість для компаній, які вирішили довіритися сучасним технологіям, таким як React.js, або вибрали підхід Jamstack.

Це дозволяє створювати як прості, так і складні веб-додатки набагато швидше, простіше, а завдяки багатьом чудовим фреймворкам, які виросли на ньому, тепер ви можете створювати надзвичайно швидкі веб-сайти, щоб досягти набагато кращої ефективності UX та SEO.

Давайте подивимося на один з цих фреймворків – Next.js, який користується все більшою популярністю і швидко став першим вибором для багатьох великих імен і компаній.

2.4.1 Що таке Next.js

Next.js — це фреймворк JavaScript, який дозволяє створювати надшвидкі та надзвичайно зручні статичні веб-сайти, а також веб-додатки за допомогою React. Насправді, завдяки автоматичній статичній оптимізації «статичні» та «динамічні» тепер стають одним цілим. Ця функція дозволяє Next.js створювати гібридні програми, які містять як відтворені на стороні сервера, так і статично згенеровані сторінки[32]. Статично згенеровані сторінки все ще реагують: Next.js зволожує вашу програму на стороні клієнта, щоб надати їй повну інтерактивність. Це відкриває нам багато переваг, таких як:

Розширений досвід користувача (простіше та швидше) Чудова продуктивність (також простіше та швидше) Швидкий розвиток функцій Next.js широко використовується найбільшими та найпопулярнішими компаніями по всьому світу, такими як Netflix, Uber, Starbucks або Twitch. Вона також вважається однією з найбільш швидкозростаючих фреймворків React, ідеально підходить для роботи зі статичними сайтами – що було найактуальнішою темою у світі веб-розробки останнім часом[20].

2.4.2 Що можна створювати за допомогою Next.js

За допомогою Next.js ви можете створити ряд цифрових продуктів та інтерфейсів, таких як:

- MVP (мінімально життєздатний продукт)
- Веб-сайти Jamstack
- Веб-портали

- Окремі веб-сторінки
- Статичні веб-сайти
- Продукти SaaS
- Веб-сайти електронної комерції та роздрібної торгівлі
- Інформаційні панелі
- Складні та вимогливі веб-додатки
- Інтерактивні інтерфейси користувача

2.4.3 Користувацький досвід в Next.js

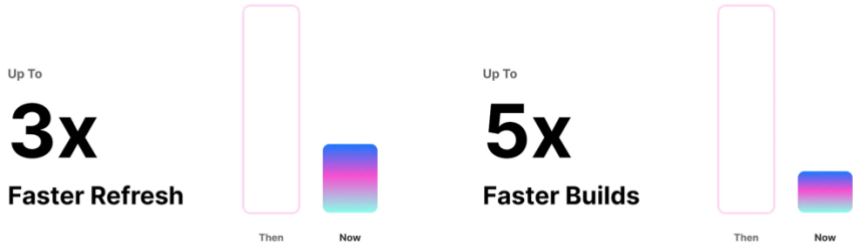


Рисунок 2.10 – Офіційна статистика по швидкодії Next.js

Наприклад, якщо у вас є інтернет-магазин і ви не дбаєте про користувацький досвід (UX) належним чином, це призведе до:

- Втрати клієнтів та відвідувачів
- Пустий кошик для замовлень
- Високий показник відмов на сайті

Дизайн також важливий – якщо ви використовуєте теми чи шаблони, швидше за все, хтось має схожий макет. Це також означає, що ви не можете створити унікальний досвід клієнта та покращити його з часом. Навіть якщо це означає змінити одну просту річ, наприклад додати кнопку на сторінку продукту або видалити її[10].

На щастя – завдяки Next.js – ви можете створити повністю налаштований користувацький досвід. Давайте подивимося, що це насправді означає.

- Свобода UX – вам не потрібно обмежувати себе жодними плагінами, шаблонами чи будь-якими іншими обмеженнями, продиктованими платформами електронної комерції чи CMS. Це дає вам повну свободу налаштовувати інтерфейс так, як вам потрібно або хочете. Це також дозволяє вносити творчі зміни без будь-яких обмежень[14].
- Адаптивність та швидкість реагування – веб-сайти та веб-додатки, створені за допомогою Next.js, працюють на будь-якому пристрої та адаптуються до будь-якого розміру та роздільної здатності екрана. Таким чином, користувачі можуть отримати доступ до вашого веб-сайту або веб-програми за допомогою свого улюбленого пристрою[13].
- Короткий час завантаження сторінки – веб-сайти Next.js надзвичайно швидкі, оскільки вони статичні, тому відвідувачі будуть більш ніж задоволені продуктивністю.
- Безпека даних – у випадку статичних веб-сайтів немає прямого підключення до бази даних, залежностей, даних користувачів або іншої конфіденційної інформації, що робить їх абсолютно безпечними.

Усі ці речі, згадані вище, роблять користувацький досвід настільки чудовим, наскільки це можливо.

Але переваги використання Next.js на цьому не закінчуються.

2.4.4 Переваги в пошуковій оптимізації в Next.js

В обох випадках це дуже допоможе вам:

- Швидше зростання органічного трафіку
- Вищий рейтинг ваших ключових слів із високим наміром

- Перевершити конкурентів легше
- Будьте більш помітними для потенційних клієнтів

Веб-сайти Next.js надзвичайно швидкі, їх легко сканувати та забезпечують чудовий досвід роботи з користувачем, тому Google віддасть їм перевагу над іншими та оцінить їх вище[12].

2.4.5 Переваги Next.js для розробників

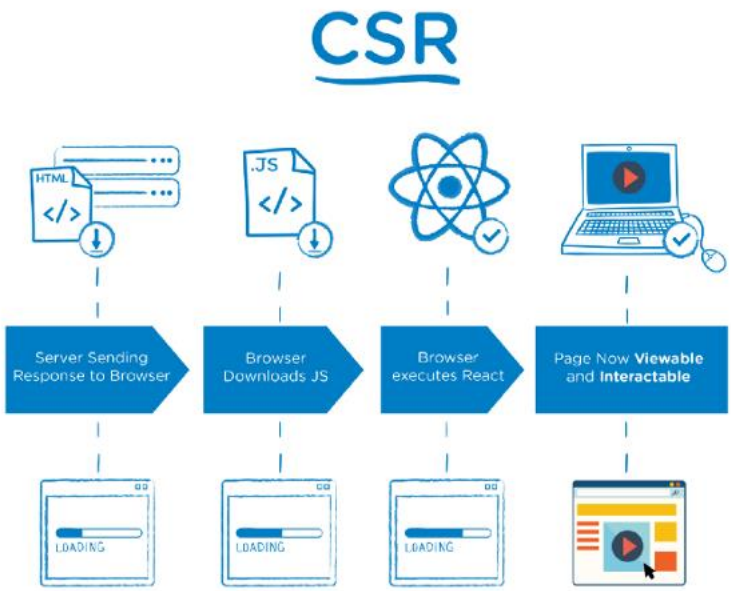


Рисунок 2.11 – Вигляд серверного рендерингу, який використовується в Next.js

Незалежно від того, чи шукаєте ви переваги з точки зору бізнесу чи технічної, ви знайдете кілька причин серйозно подумати про вибір Next.js.

Якщо ви хочете створити складний і вимогливий додаток, природа розробки Next.js React дозволяє заощадити багато часу. Розробники особливо віддають перевагу таким функціям, як:

- Zero Config – Next дозволяє зосередитися на бізнес-логіці вашої програми, а не на логіці програми. І щоб допомогти вам, він забезпечує автоматичну компіляцію та групування. Іншими словами, Next оптимізовано для виробництва з самого початку[33].
- Інкрементальна статична регенерація – вона дозволяє оновлювати сторінки, повторно відтворюючи їх у фоновому режимі в міру надходження трафіку. Іншими словами, статичний вміст може стати динамічним.
- Гібрид SSR рендеринга на стороні сервера та SSG для створення статичного сайту – попереднє відтворення сторінок під час створення або запиту в одному проекті.
- Підтримка TypeScript – автоматична конфігурація та компіляція TypeScript.
- Швидке оновлення – швидкий досвід редагування в реальному часі – зміни, зроблені на компонентах React, опубліковані за лічені секунди. Він працює аналогічно гарячій заміні модуля (HMR).
- Парсери CSS – можливість імпортувати файли CSS з файлу JavaScript. Нові аналізи покращили роботу з CSS[22].
- Вбудований компонент зображення та автоматична оптимізація зображення – ця функція автоматично оптимізує зображення[35].
- Автоматичне розділення коду – автоматично зменшує розмір сторінки, розділяючи код і обслуговуючи компоненти лише за потреби. Модулі також можна автоматично імпортувати завдяки опції динамічного імпорту.
- Вибір даних – ця опція дозволяє відображати вміст різними способами, відповідно до випадку використання програми. Це можна зробити шляхом попереднього рендеринга за допомогою SSR відображення на стороні сервера або створення статичного сайту, а також шляхом оновлення або створення вмісту за допомогою ISR.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі було розглянуті осносні технології, мови програмування та фреймворки, які будуть використані, для проектування та розробки системи. Кожну технологію було обгрунтовано та наведені плюси її використання.

Головним параметром вибору засобів для розробки було їх сучасність та можливість програмування схожих систем, які були розглянуті в першому розділі. Першим було розглянуто прогресивні веб-програми – що являють собою основу системи, структуру, методику роботи та можливість використання.

Далі було розглянуто тип мікросервісної системної архітектури, що стане головною при проектуванні архітектури системи. Головним її плюс є ізольованість модулів програми, що дає додаткову безпеку та простоту у маштабованні.

Для бекенд розробки був обраний ExpressJS, який є фреймворком техтології NodeJS, що дозволяє будувати швидке та сучасне серверне програмне забеспечення та демонструє чудову продуктивність

Для реалізації фронтенд розробки використовується NextJS що є фреймворком для технології ReactJS та надає фантастичні показники в оптимізації сайту.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ

3.1 Архітектура системи

Фінальна структура системи складається з чотирьох модулів, які пов’язані один з одним але можуть працювати автономно

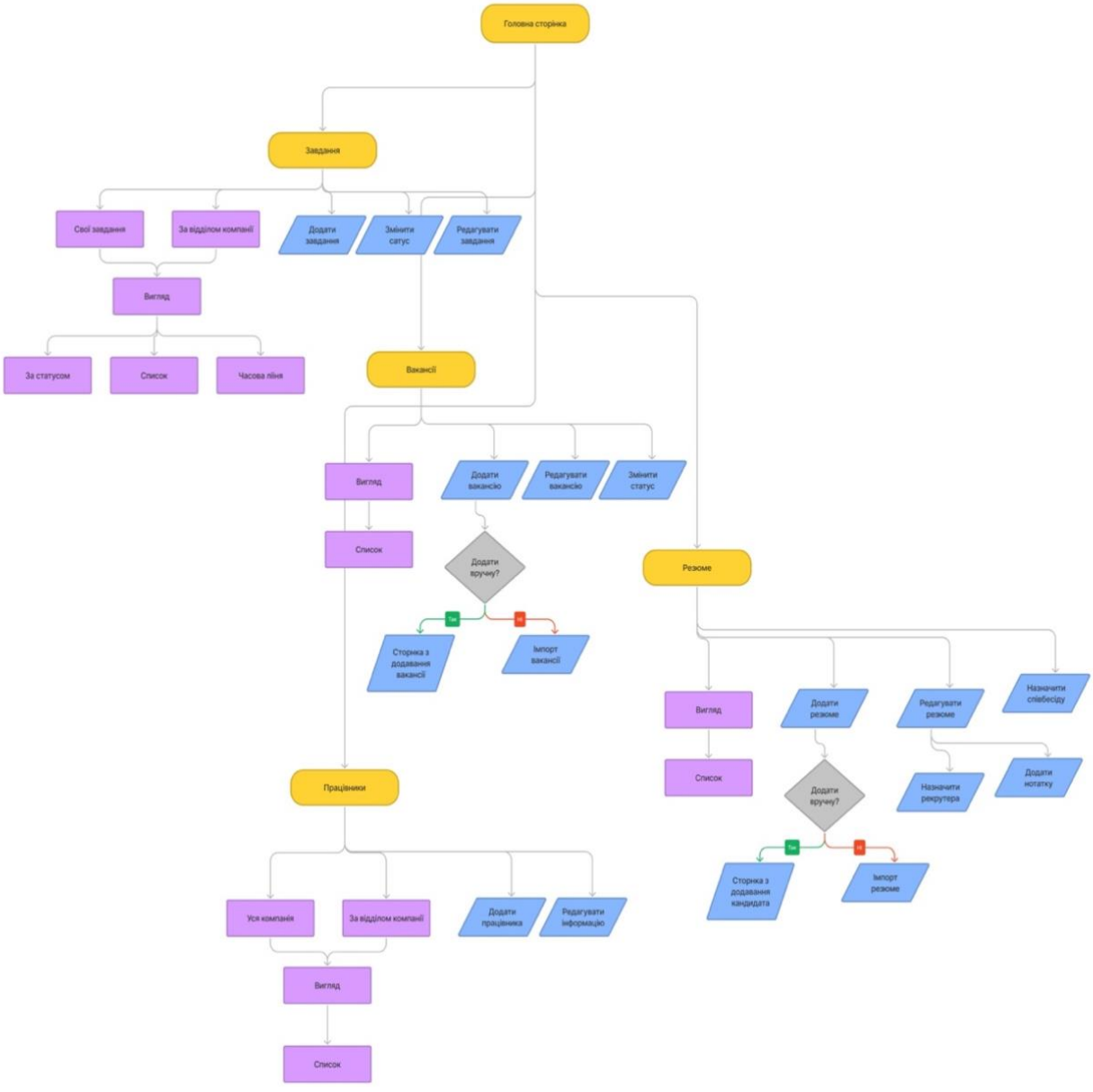


Рисунок 3.1 – Стуктура системи

Усі сервіси пов’язує головна сторінка - це перше, що бачить користувач після входу у систему.

3.2 Юзер сторіс

3.1.1 Сервіс зберігання резюме кандидатів

Сервіс зберігання резюме кандидатів створений, для полегшення найму працівників в середину компанії так і для рекрутингових агенцій, чиїми основними послугами є підбор персоналу іншим компаніям.

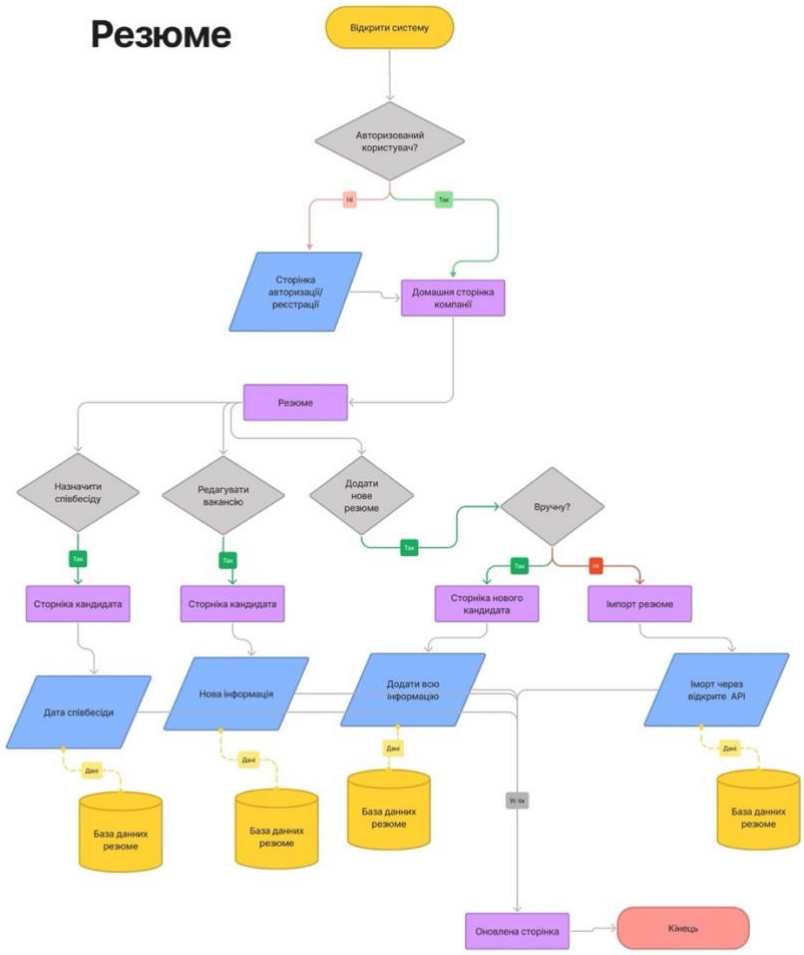


Рисунок 3.2 – Можливості у використання модуля резюме

Головне завдання полегшити цей процес. Резюме можна створити вручну – заповнивши всю необхідну інформацію або можна імпортувати готові резюве з сайтів з пошуку роботи.

3.1.2 Сервіс управління завданнями

Сервіс для управління завданнями, дозволяє користовачеві створювати нові завдання для себе, підлеглого або цілої команди.

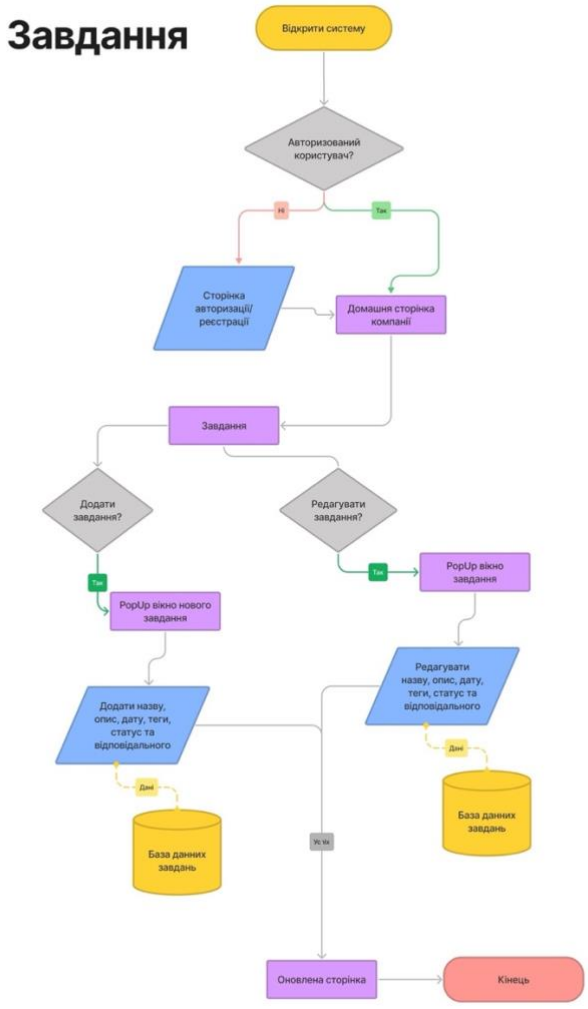


Рисунок 3.3 – Схема можливостей у використанні модуля завдання

У кожного завдання є крайній термін виконання. У кожного завдання є індивідуальні теги, які відображають тип та відділ компанії, до якого воно

відноситься. Загальний вигляд сторінки з завданнями за замовчуванням у стилі канбан дошки (колонки з карточками). Також стиль відображення є індивідальним тому кожен користувач може налаштувати його під себе.

3.1.3 Сервіс для ведення вакансій

Сервіс для управління вакансіями компанії, для рекрутерів та керівників.

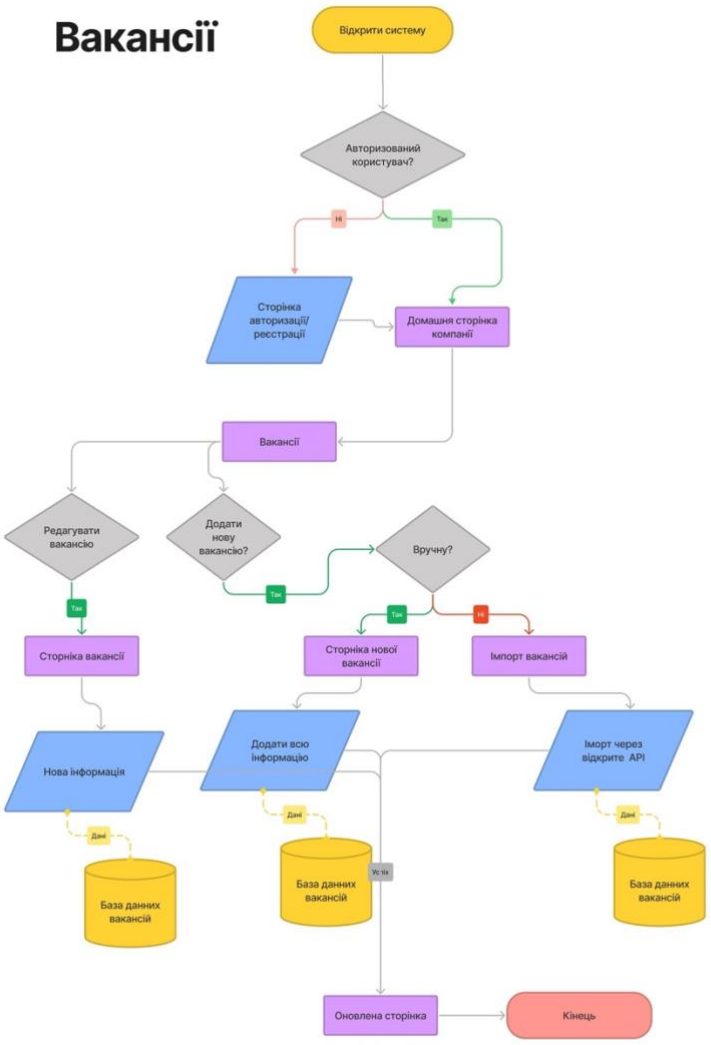


Рисунок 3.4 – Можливості модуля вакансії

За допомогою нього можна створювати сторінки вакансій, назначити для кожної відкритої вакансії рекрутера, який буде відповідальний по ній та буде

займатись усіма кандидатами на неї. Вакансію можна додати вручну заповнивши усі необхідні дані або імортувати відразу декілька вакансій.

3.1.4 Сервіс для ведення працівників компанії

Сервіс для управління внутрішнім персоналом компанії.

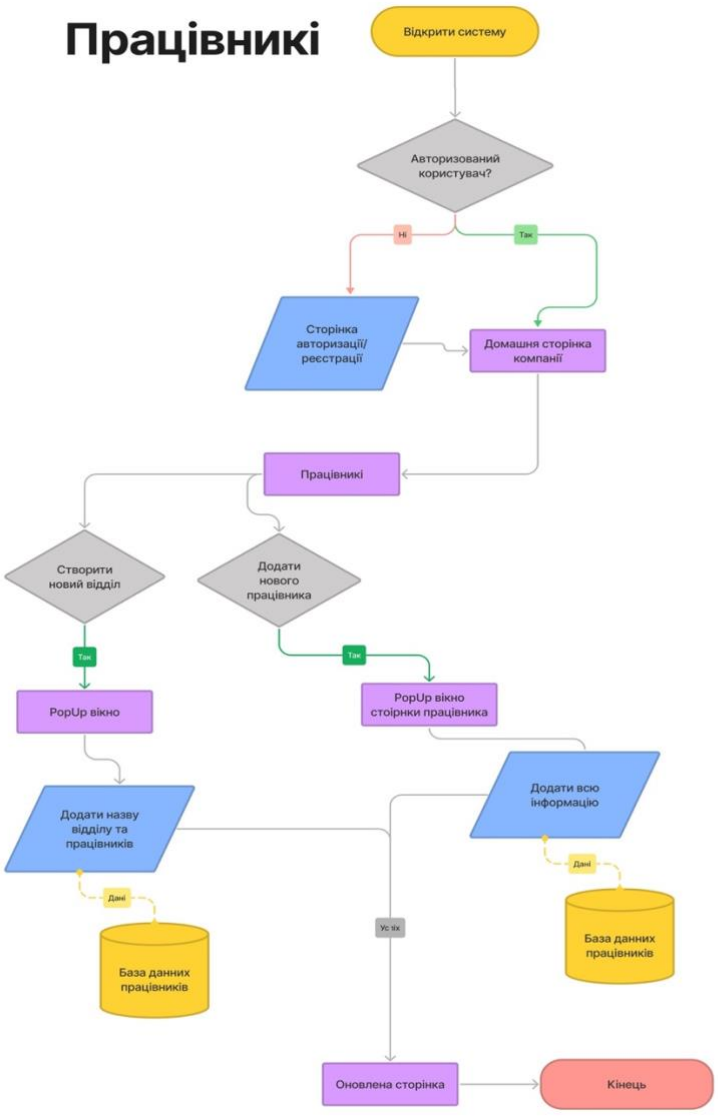


Рисунок 3.5 – Можливості модуля працівникі

Можливість додавання нових працівників є тільки в користувача з правами адміністратора. Перегляд усіх працівників відображений у ви.

3.2 Проектування та розробка бази данних

3.2.1 Мікросервіс управління вакансіями

База данних мікросервісу вакансій складється з шести пов’язаних між собою таблиць.

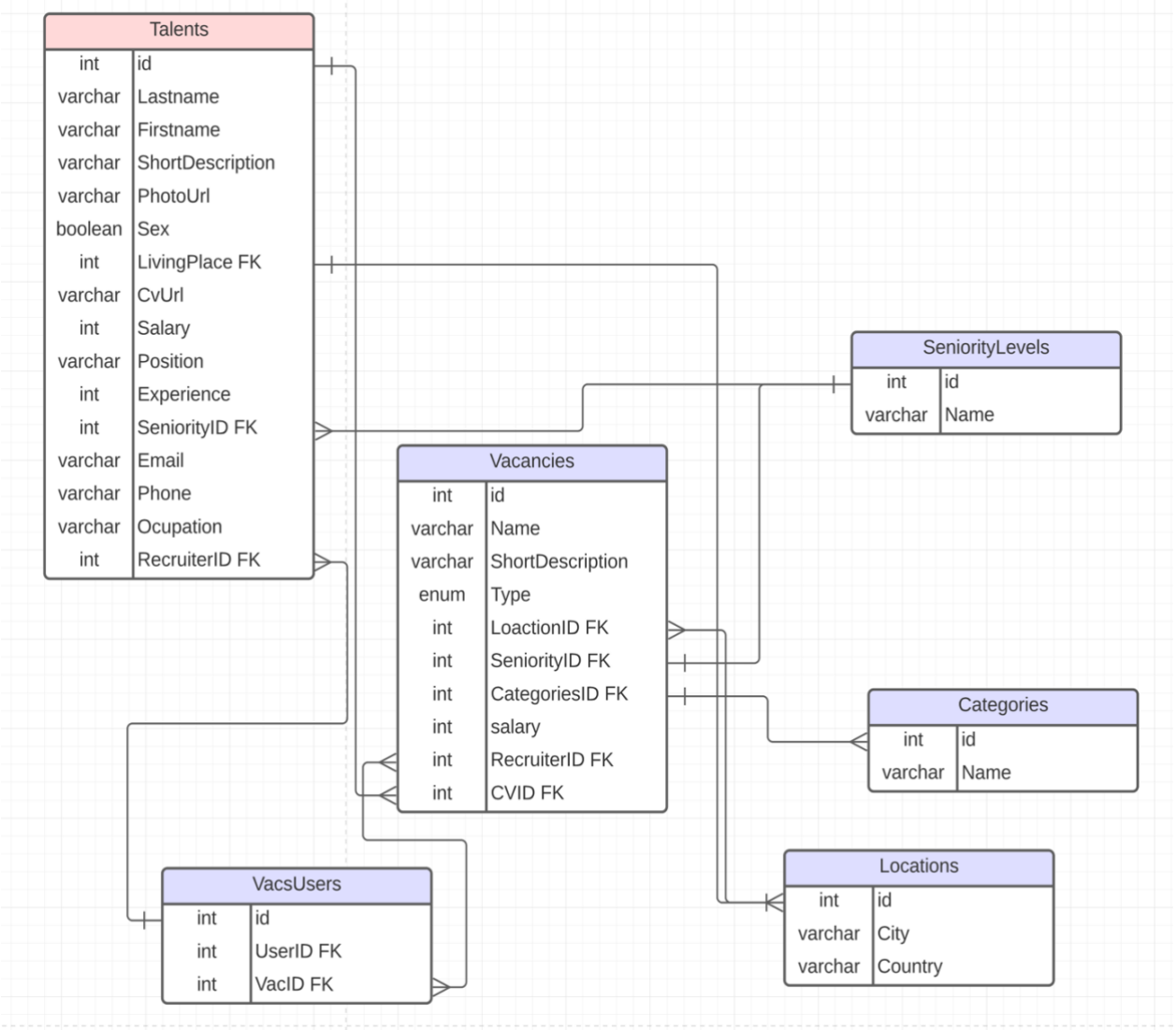


Рисунок 3.6 – База данних мікросервісу вакансії

Основними функціями які повинна виконувати дана база данних це – відображення списку вакансій, з відповідальними рекретерами та з резюме кандидатів, які його залишили.

3.2.2 Мікросервіс управління завданнями

База данних мікросервісу завдань складється з восьми пов’язаних між собою таблиць.

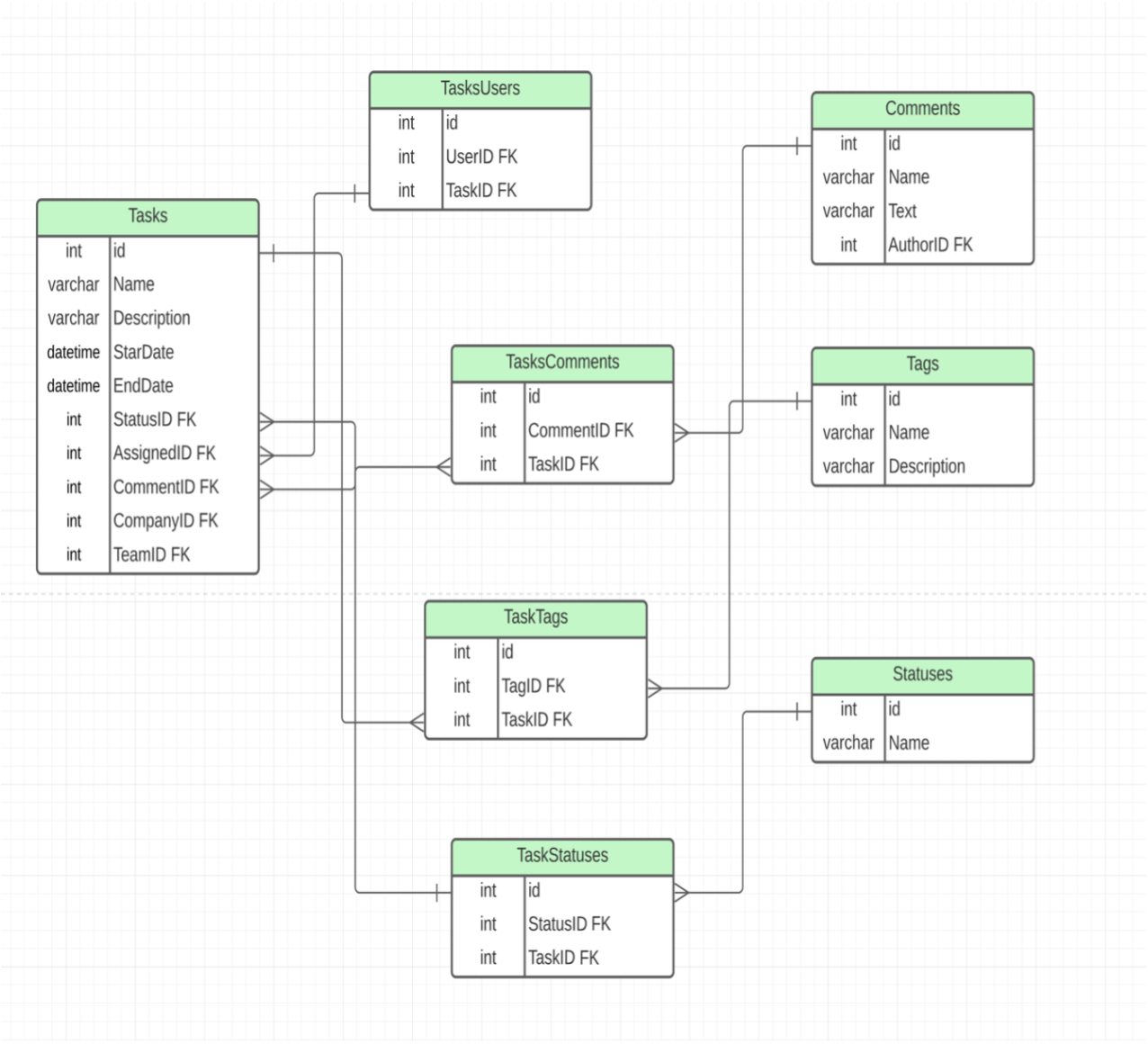


Рисунок 3.7 – База данних мікросервісу завдань

Головною задачою завдань є – моніторинг продуктивності працівників компанії. Відповідно, кожне завдання має крайній термін виконання, який встановлюється постановником та відповідального, який виконує.

3.2.3 Мікросервіс користувачів

База данних мікросервісу користувачів складється з шести пов’язаних між собою таблиць.

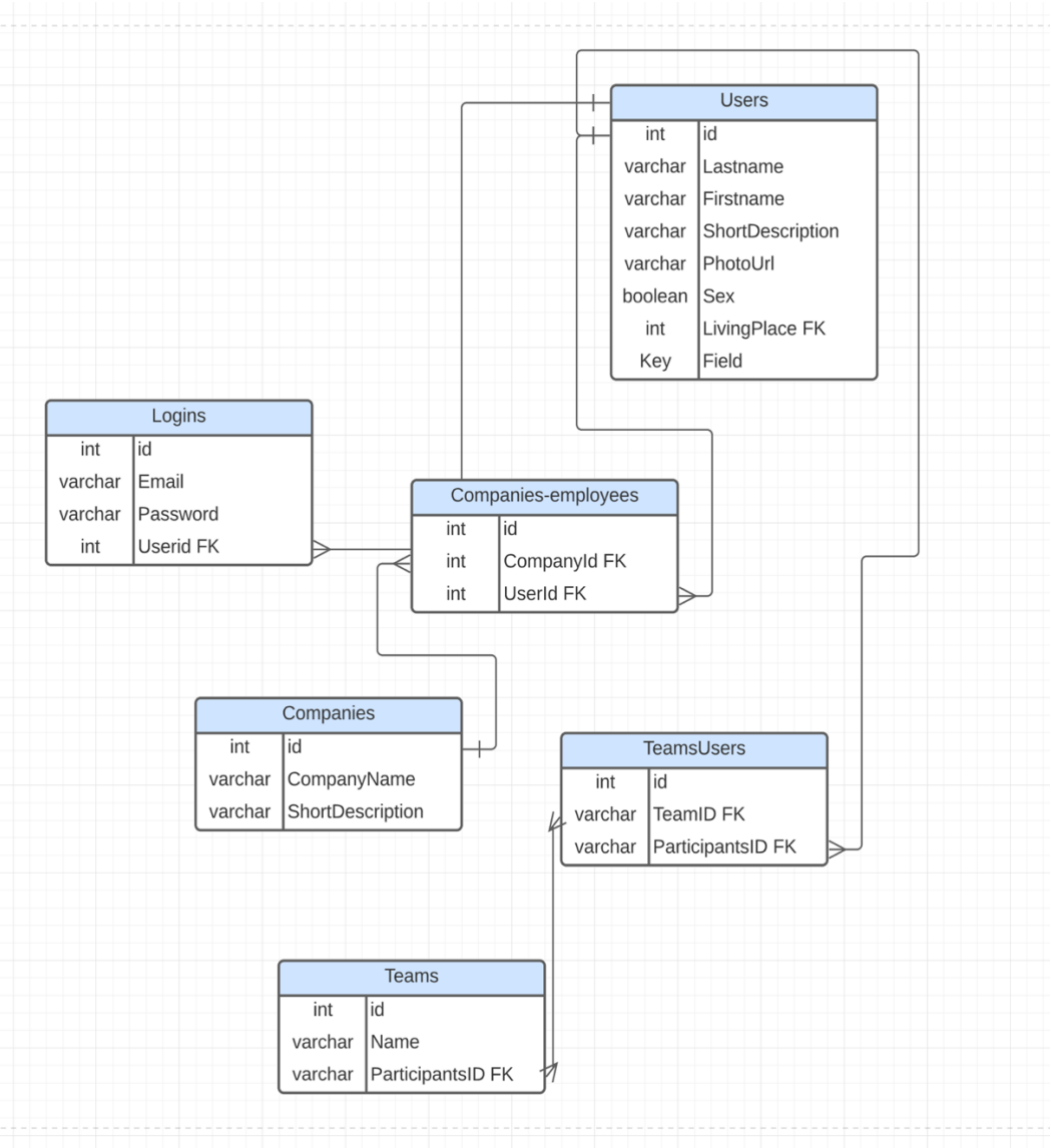


Рисунок 3.8 – База данних мікросервісу користувачів

Метою відокремлення таблиць з інформацією про користувачів є – безпека та ізольованість її від інших процесів. За її допомогою можна формувати відділи в середині компанії. Та переглядати список усіх працівників.

3.2.3 Патерн база данних під кожен сервіс

Служби повинні бути слабо пов’язані, щоб їх можна було розробляти, розгортати та масштабувати незалежно

- Деякі бізнес-операції повинні застосовувати інваріанти, які охоплюють декілька послуг. Наприклад, варіант використання "Розмістити замовлення" повинен підтвердити, що нове замовлення не перевищуватиме кредитний ліміт клієнта. Інші бізнес-операції повинні оновлювати дані, що належать кільком службам.
- Деякі бізнес-операції мають запитувати дані, які належать кільком службам. Наприклад, використання "Перегляд доступного кредиту" має запитати Клієнта, щоб знайти кредитний ліміт і замовлення для обчислення загальної суми відкритих замовлень.
- Деякі запити мають об’єднувати дані, які належать кільком службам. Наприклад, для пошуку клієнтів у певному регіоні та їхніх останніх замовлень потрібне об’єднання клієнтів і замовлень.
- Для масштабування бази даних іноді потрібно реплікувати та розділяти.
- Різні служби мають різні вимоги до зберігання даних. Для деяких служб найкращим вибором є реляційна база даних. Іншим службам може знадобитися база даних NoSQL, така як MongoDB, яка добре зберігає складні, неструктуровані дані, або Neo4J, яка розроблена для ефективного зберігання та запитів даних графіка.

Зберігайте постійні дані кожного мікросервісу конфіденційними для цієї служби та доступними лише через його API. Транзакції служби включають лише її базу даних.

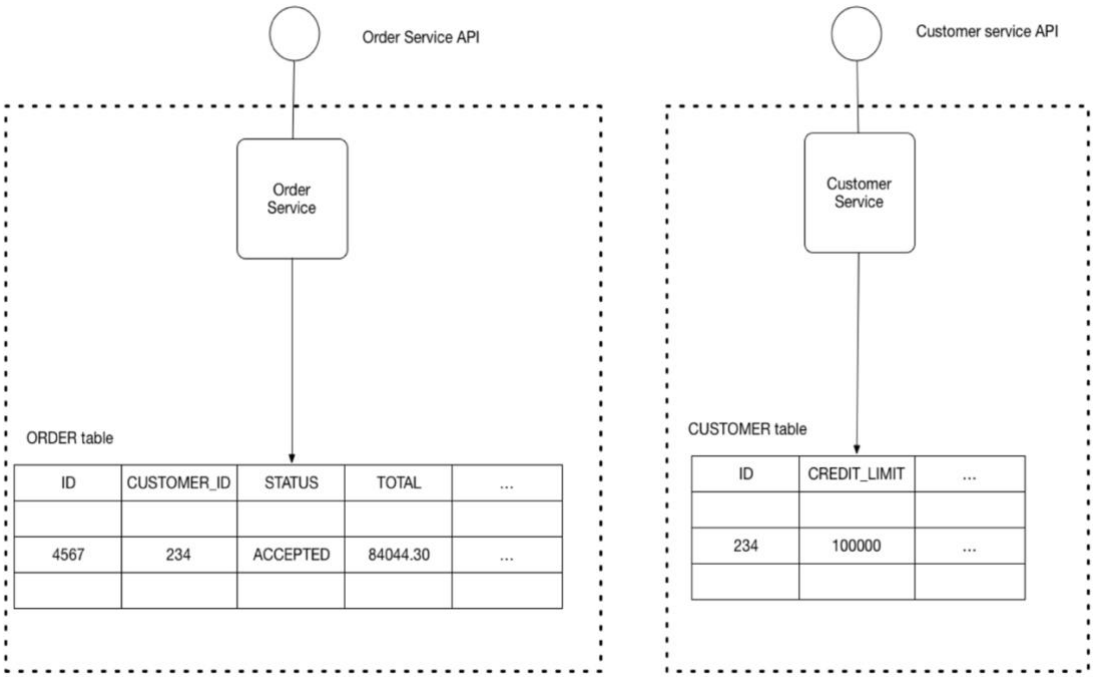


Рисунок 3.9 – Модель баз данних в мікросервісах

База даних служби фактично є частиною реалізації цієї служби. Інші служби не можуть отримати прямий доступ до нього.

Є кілька різних способів зберегти конфіденційність постійних даних служби. Вам не потрібно надавати сервер бази даних для кожної служби. Наприклад, якщо ви використовуєте реляційну базу даних, можливі такі варіанти:

Private-tables-per-service – кожна служба володіє набором таблиць, доступ до яких повинен мати лише цей сервіс Схема на службу – кожна служба має схему бази даних, яка є приватною для цієї служби Database-server-per-service – кожна служба має власний сервер бази даних. Приватні таблиці на сервіс і схема на послугу мають найменші накладні витрати. Використання схеми для кожної служби є привабливим, оскільки робить право власності більш зрозумілим. Деякі високопродуктивні служби можуть потребувати власного сервера бази даних.

Це гарна ідея створити бар’єри, які забезпечують цю модульність. Ви можете, наприклад, призначити різний ідентифікатор користувача бази даних кожній службі та використовувати механізм контролю доступу до бази даних, наприклад гранти. Без певного бар’єру для забезпечення інкапсуляції у розробників завжди виникне спокуса обійти API служби та отримати прямий доступ до її даних.

Використання бази даних для кожної служби має такі переваги: Допомагає гарантувати, що послуги слабо пов’язані. Зміни в базі даних одного сервісу не впливають на інші служби. Кожна служба може використовувати той тип бази даних, який найкраще відповідає її потребам. Наприклад, служба, яка виконує текстовий пошук, може використовувати ElasticSearch. Служба, яка маніпулює соціальним графіком, може використовувати Neo4j.

Використання бази даних для кожної служби має такі недоліки:

- Реалізація бізнес-транзакцій, які охоплюють декілька послуг, не є простою. Через теорему CAP краще уникати розподілених транзакцій. Більше того, багато сучасних (NoSQL) баз даних не підтримують їх.
- Реалізація запитів, які об’єднують дані, які зараз знаходяться в кількох базах даних, є складною задачею.
- Складність управління кількома базами даних SQL і NoSQL

Реалізація запитів, які охоплюють послуги:

- Композиція API – програма виконує об’єднання, а не базу даних. Наприклад, служба (або шлюз API) може отримати клієнта та його замовлення, спочатку витягуючи клієнта зі служби обслуговування клієнтів, а потім запитуючи службу замовлень, щоб повернути останні замовлення клієнта.

- Відокремлення відповідальності за командні запити (CQRS) – підтримувати одне або кілька матеріалізованих представлень, які містять дані з кількох служб. Перегляди зберігаються службами, які підписалися на події, які кожна служба публікує, коли оновлює свої дані. Наприклад, інтернет-магазин може реалізувати запит, який знаходить клієнтів у певному регіоні та їхні останні замовлення, підтримуючи представлення, яке об'єднує клієнтів і замовлення. Подання оновлюється службою, яка підписується на події клієнтів і замовлень.

3.3 Авторизація користувачів

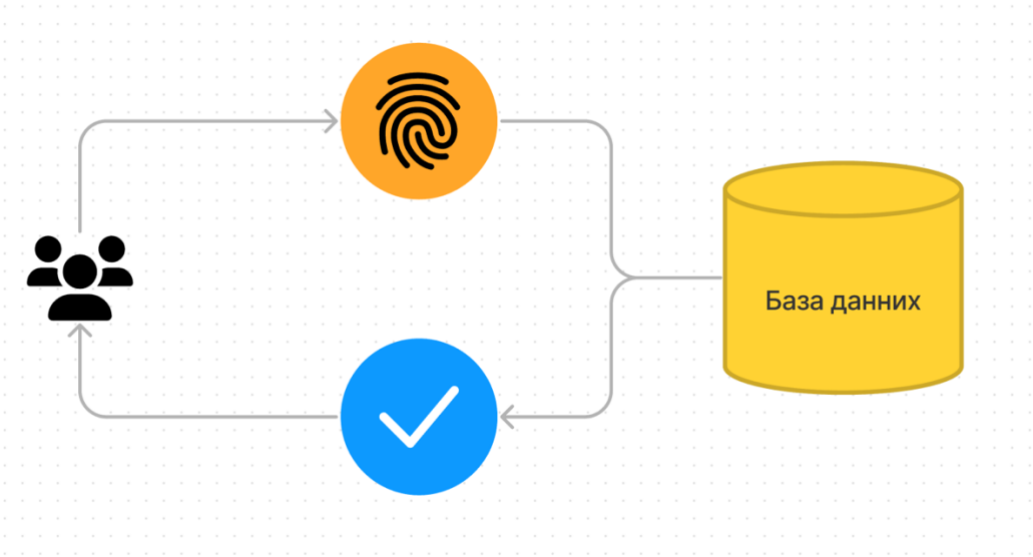


Рисунок 3.10 – Модель авторизації користувачів

Архітектура мікросервісів дає багато переваг програмним додаткам, включаючи невеликі групи розробників, коротші цикли розробки, гнучкість вибору мови та покращену масштабованість сервісу. У той же час також було впроваджено багато складних проблем розподілених систем. Однією з проблем є те, як реалізувати гнучку, безпечну та ефективну схему аутентифікації та

авторизації в архітектурі мікросервісів. У цій статті спробуємо провести більш повне обговорення цього питання.

3.3.1 Проблеми авторизації користувачів

Відповідно до архітектури мікросервісів, додаток розбивається на кілька мікросервісних процесів, і кожен мікросервіс реалізує бізнес-логіку одного модуля в оригінальній єдиній програмі. Після розділення програми запит на доступ до кожної мікросервіси має бути аутентифікований та авторизований. Якщо ви посилаєтеся на реалізацію програми Monolithic, ви зіткнетеся з такими проблемами:

- Логіку аутентифікації та авторизації потрібно обробляти в кожній мікросервісі, і цю частину глобальної логіки необхідно повторно реалізовувати в кожній мікросервісі. Хоча ми можемо використовувати базу коду для повторного використання частини коду, це, у свою чергу, призведе до того, що всі мікросервіси будуть залежати від певної бази коду та її версії, що вплине на гнучкість вибору мови/фреймворку мікросервісів.
- Мікросервіси повинні дотримуватися принципу єдиної відповідальності. Мікросервіс обробляє лише одну бізнес-логіку. Глобальну логіку аутентифікації та авторизації не слід розміщувати в реалізації мікросервісу.
- HTTP — це протокол без стану. Для сервера кожен раз, коли HTTP-запит користувача є незалежним. Без стану означає, що сервер може надсилати клієнтські запити будь-якому вузлу в кластері за потреби. Конструкція HTTP без стану має очевидні переваги для балансування навантаження. Оскільки немає стану, запити користувачів можуть бути розподілені на будь-який сервер. Для служб, які не потребують аутентифікації, наприклад перегляд сторінок новин, проблем немає. Однак багатьом

сервісам, таким як інтернет-магазини та системи управління підприємством, потрібно автентифікувати особу користувача. Таким чином, необхідно зберегти статус входу користувача у спосіб, що ґрунтується на протоколі HTTP, щоб запобігти потребі користувача виконувати перевірку для кожного запиту. Традиційним способом є використання сеансу на стороні сервера для збереження стану користувача. Оскільки сервер зберігає стан, це впливає на горизонтальне розширення сервера.

- Аутентифікація та авторизація в архітектурі мікросервісів включають сценарії, які є більш складними, за участю користувачів, які отримують доступ до мікросервісних програм, сторонніх додатків, які отримують доступ до програм мікросервісів, і кількох мікросервісних програм, які мають доступ один до одного, і в кожному сценарії необхідно виконати наступні схеми автентифікації та авторизації. розглядатися для забезпечення безпеки програми.

3.2.4 Технічні рішення

Щоб повністю використовувати переваги архітектури мікросервісів і досягти масштабованості та стійкості мікросервісів, бажано, щоб мікросервіси були без стану. Це рішення можна застосувати різними способами, наприклад:

- Ліпкий сеанс. Це гарантує, що всі запити від конкретного користувача будуть надіслані на той самий сервер, який обробив перший запит, що відповідає цьому користувачу, таким чином гарантуючи, що дані сеансу завжди правильні для певного користувача. Однак це рішення залежить від балансувальника навантаження, і воно може відповідати сценарію горизонтально розширеного кластера, але коли балансувальник навантаження змушений раптово з будь-якої причини перемістити

користувачів на інший сервер, усі дані сеансу користувача будуть втрачені.

- Реплікація сеансу. Означає, що кожен екземпляр зберігає всі дані сеансу та синхронізується через мережу. Синхронізація даних сеансу спричиняє накладні витрати на пропускну здатність мережі. Поки дані сеансу змінюються, дані потрібно синхронізувати з усіма іншими машинами. Чим більше екземплярів, тим більшу пропускну здатність мережі забезпечує синхронізація.
- Централізоване зберігання сесій. Означає, що коли користувач звертається до мікросервісу, дані користувача можуть бути отримані зі спільного сховища сеансу, гарантуючи, що всі мікросервіси можуть читати одні й ті самі дані сеансу.

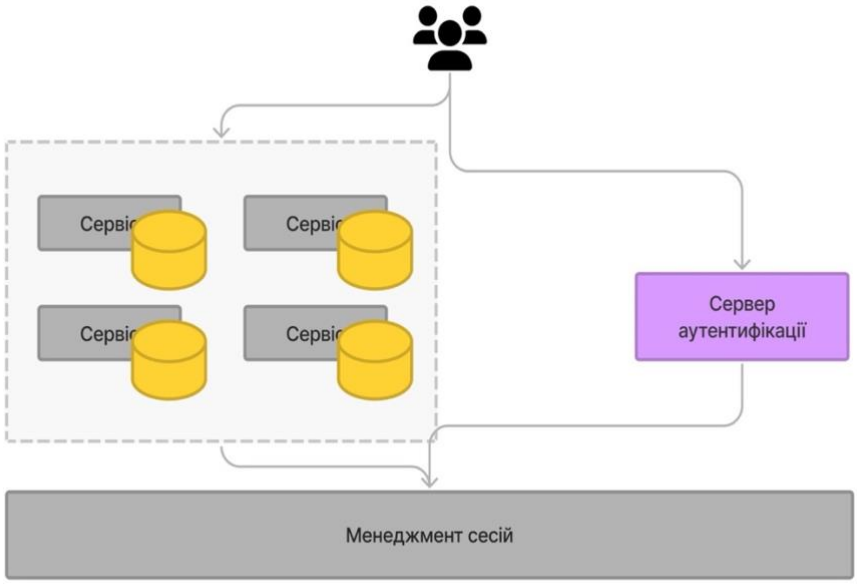


Рисунок 3.11 – Модель доступа користувачів до системи

У деяких сценаріях ця схема дуже хороша, а статус входу користувача непрозорий. Це також високодоступне і масштабоване рішення.

Клієнтський токен

Традиційним способом є використання сеансу на стороні сервера для збереження стану користувача. Оскільки сервер зберігає стан, це впливає на горизонтальне розширення сервера. Рекомендується використовувати Token для запису статусу входу користувача в архітектуру мікросервісів. Основна відмінність між Token і Session полягає в тому, де сховище відрізняється. Сесії зберігаються централізовано на сервері; Токени зберігаються самим користувачем і зазвичай зберігаються у браузері у вигляді файлів cookie. Токен містить ідентифікаційну інформацію користувача, і щоразу, коли запит надсилається на сервер, сервер може визначити ідентичність відвідувача та визначити, чи має він доступ до запитуваного ресурсу. Маркер використовується для позначення особи користувача. Тому вміст токена необхідно зашифрувати, щоб уникнути фальсифікації запитувачем або третьою стороною. JWT (Json Web Token) — це відкритий стандарт (RFC 7519), який визначає формат маркера, визначає вміст маркера, шифрує його та надає lib для різних мов.

Структура токена JWT дуже проста і складається з трьох частин:

```
{
"typ": "JWT",
"alg": "HS256"
}
```

Рисунок 3.12 – Заголовок JWT токена

- Заголовок заголовок містить тип, фіксоване значення JWT. Потім алгоритм хешування, використовуваний JWT.

- Корисне навантаження містить стандартну інформацію, таку як ідентифікатор користувача, термін дії та ім’я користувача. Він також може додавати ролі користувачів і визначену користувачем інформацію.

```
{
"id": 123,
"name": "Anton Podash",
"is_admin": true,
"expire": 1558213420
}
```

Рисунок 3.13 – Інформація токена

- Підпис Підпис токена використовується клієнтом для перевірки ідентичності токена, а також для перевірки того, що повідомлення не було змінено в дорозі.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
)
```

Рисунок 3.14 – Підпис токена

Ці три частини об’єднуються за допомогою кодування Base64 і стають рядками Token, які в кінцевому підсумку повертаються клієнту, розділені символом «.», Токен, сформований у наведеному вище прикладі, буде таким:

```
1   eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MTIzLC
    JuYW1lIjoiTWVuYSBNZXNlaGEiLCJpc19hZG1pbiI6dHJ1ZSwiZ
    XhwaXJlIjoxNTU4MjEzNDIwfQ.Kmy_2WCPbpg-
    aKQmiLaKFLxb5d3rOC71DHexncH_AcQ
```

Рисунок 3.15 – Кінцевий вигляд токену

Використовуючи маркер для аутентифікації користувача, сервер не зберігає статус користувача. Клієнт повинен надсилати маркер на сервер для аутентифікації щоразу, коли клієнт запитує його.

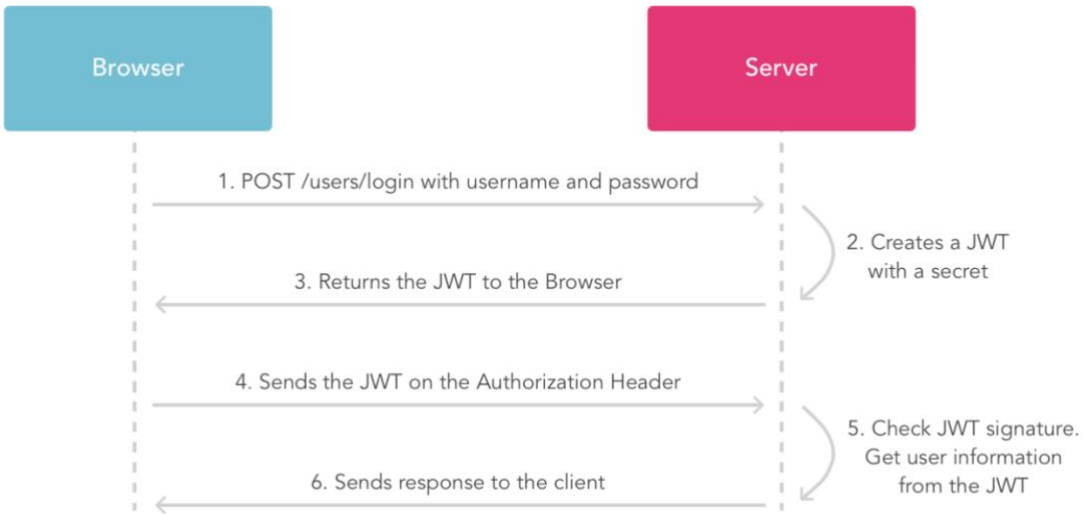


Рисунок 3.16 – Зв’язок користувача з сервером

Єдиний вхід

Ідея єдиного входу проста, тобто користувачам потрібно лише один раз увійти в програму, після чого вони зможуть отримати доступ до всіх мікросервісів програми. Це рішення означає, що кожна орієнтована на

користувача служба повинна взаємодіяти зі службою аутентифікації, як показано на наступній діаграмі:

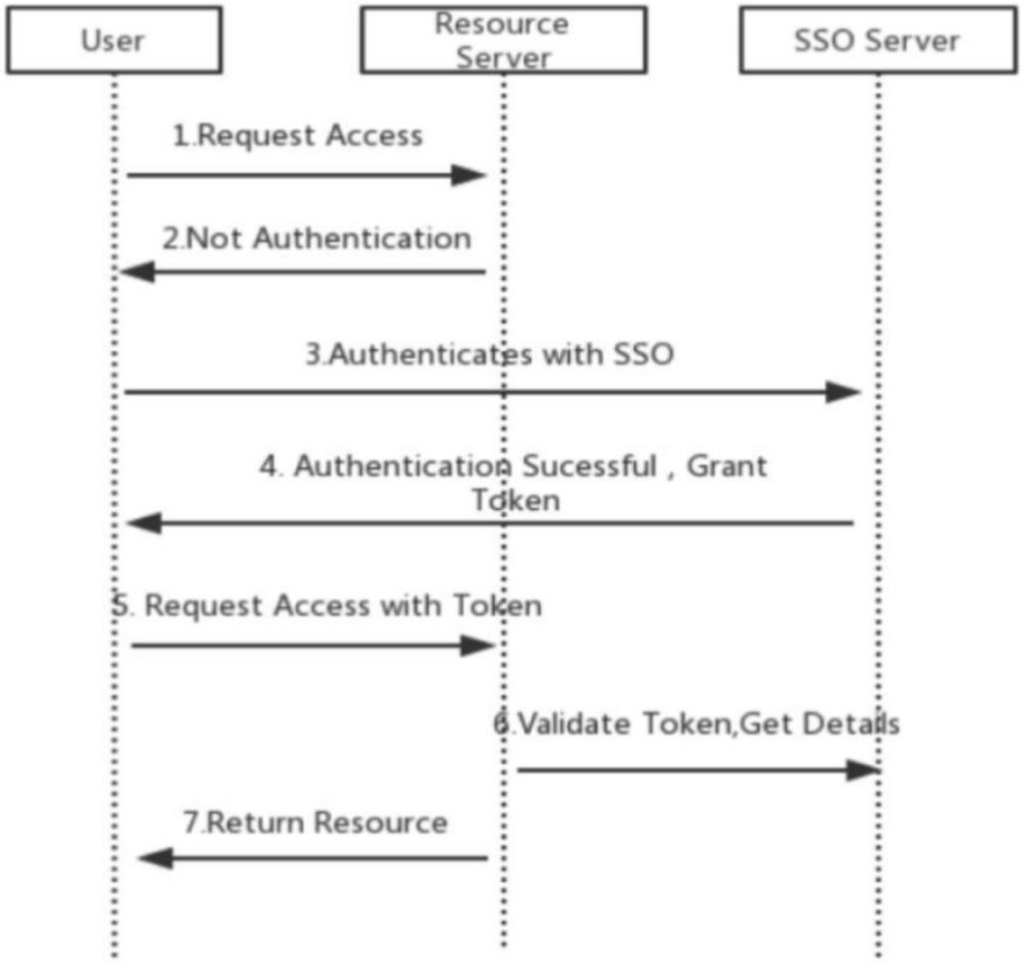


Рисунок 3.17 – Робота єдиного входа авторизації

Це може призвести до великої кількості дуже тривіального мережевого трафіку, повторної роботи, і це може призвести до однієї точки збою. Коли є десятки мікропрограм, недоліки цього рішення стануть більш очевидними.

Клієнтський токен із шлюзом API

Процес аутентифікації користувача подібний до основного процесу аутентифікації токена. Різниця полягає в тому, що шлюз API додається як вхід для зовнішнього запиту. Цей сценарій означає, що всі запити проходять через

шлюз API, фактично приховуючи мікросервіси. За запитом шлюз API перетворює вихідний маркер користувача в непрозорий маркер, який тільки сам може розв’язувати.

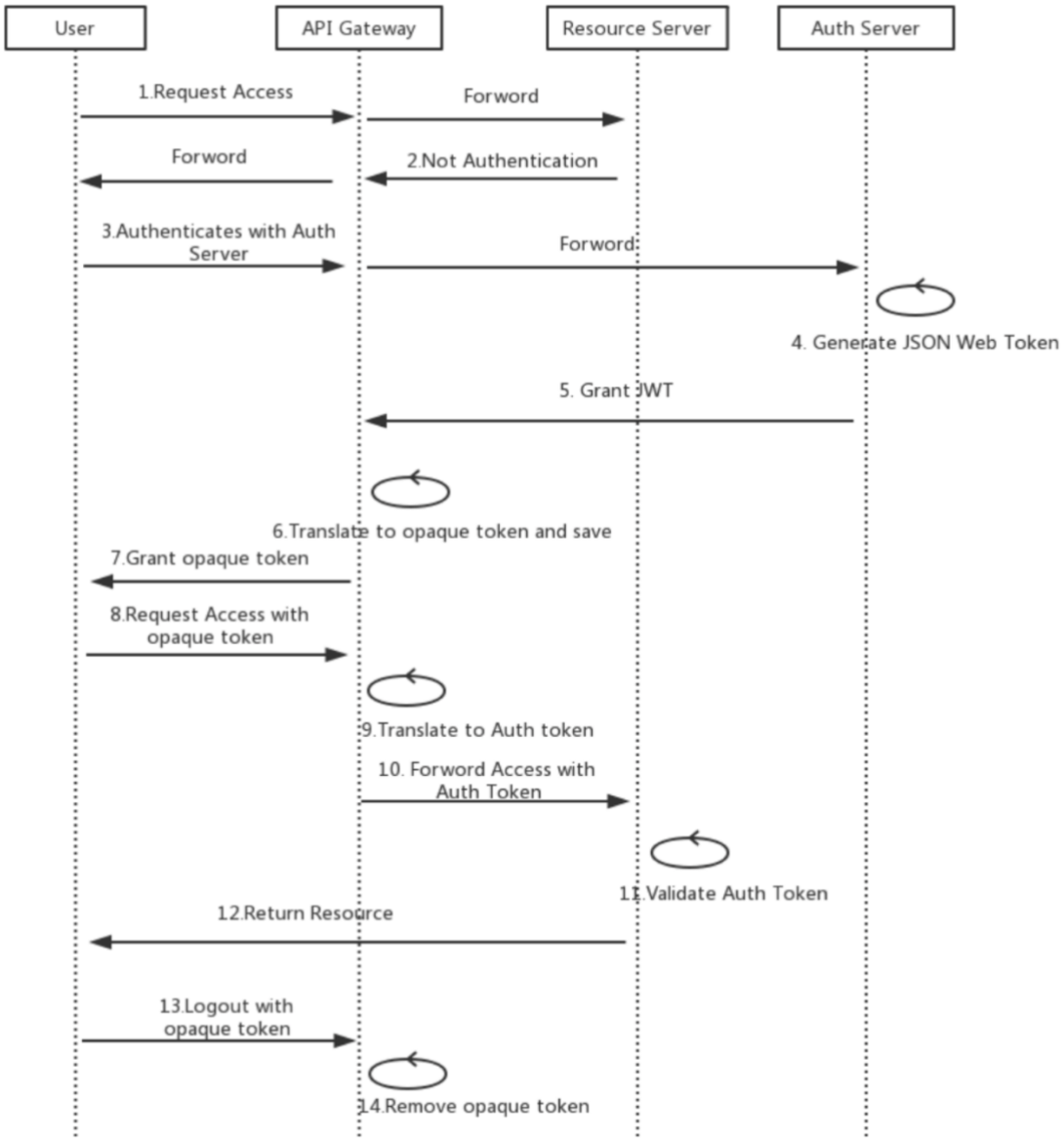


Рисунок 3.18 – Шлюз API клієнтського токену

У цьому випадку вихід із системи не є проблемою, оскільки шлюз API може відкликати маркер користувача, коли він виходить із системи, а також додає додатковий захист маркеру Auth від розшифровки, приховуючи його від клієнта.

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі була розглянута спроектована архітектура системи, юзер сторіс користувачів, функціональні особливості.

Були представлена структура бази данних, яка розподілена та спроектована океремо під кожен модуль системи. Детально оглянута та описана кожна діаграма окремої бази данних

Технології були використані, описані в 2 розділі. Окреме місце було надато процесу авторизації користувачів в системі.

РОЗДІЛ 4 ОГЛЯД РЕАЛІЗОВАНОЇ СИСТЕМИ

4.1 Огляд кожної сторінки системи

4.1.1 Головна сторінка

Початкова сторінка, яку бачить користувач при першому знайомстві з системою.

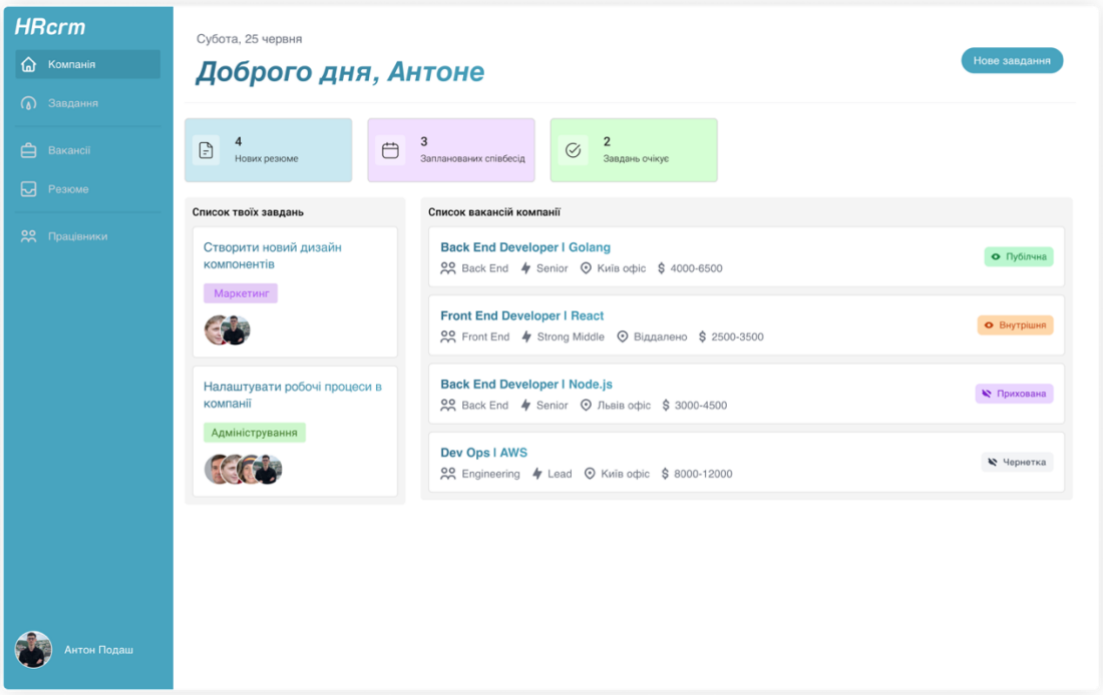


Рисунок 4.1 – Скриншот головної сторінки системи

У кожного користувача видно на домашній сторінці його завданні, які чекають виконання, також список актуальних вакансій компанії.

4.1.2 Сторінка резюме

Сторінка з детальним оглядом кожного резюме кандидата.

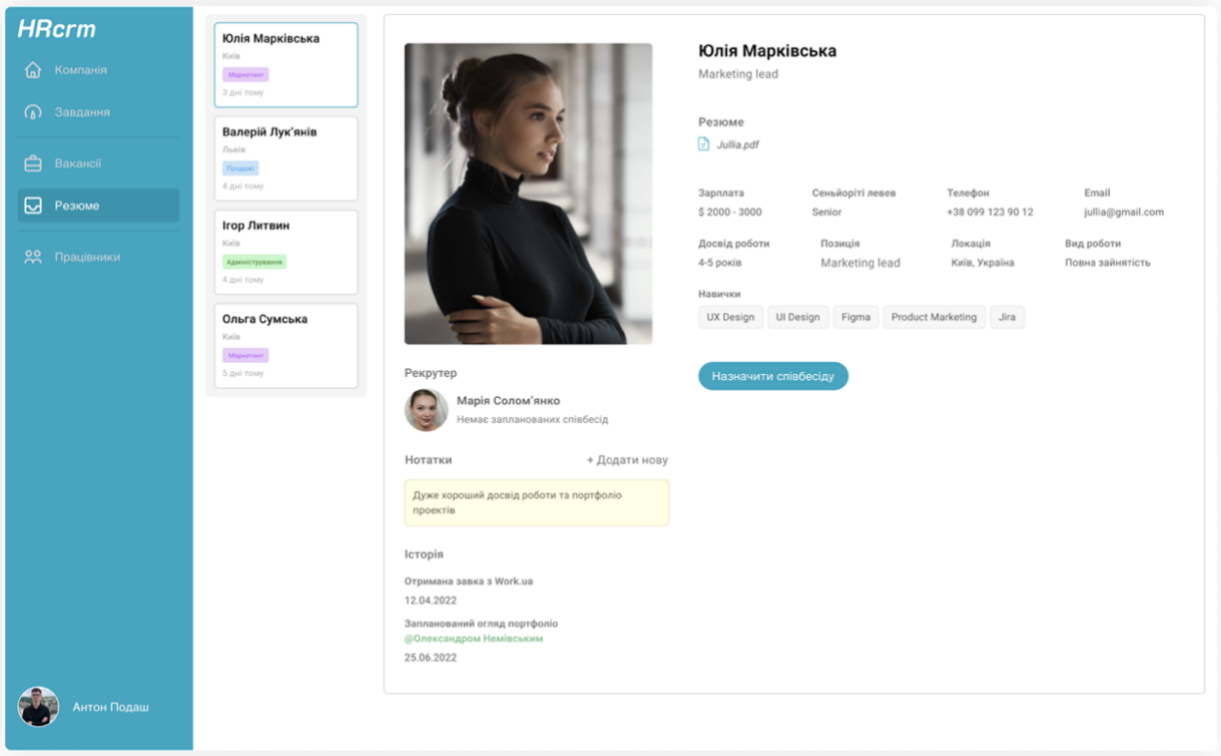


Рисунок 4.2 – Скриншот столрінки резюме

Користувач може подивитись усі доступні резюме компанії, назначити співбесіду конкретному кандидатові та написати замітки до резюме. Під кожним резюме закріплений відповідний рекрутер компанії.

4.1.3 Сторінка додавання вакансії

Сторінка для додавання нової вакансії компані вручну працівниками.

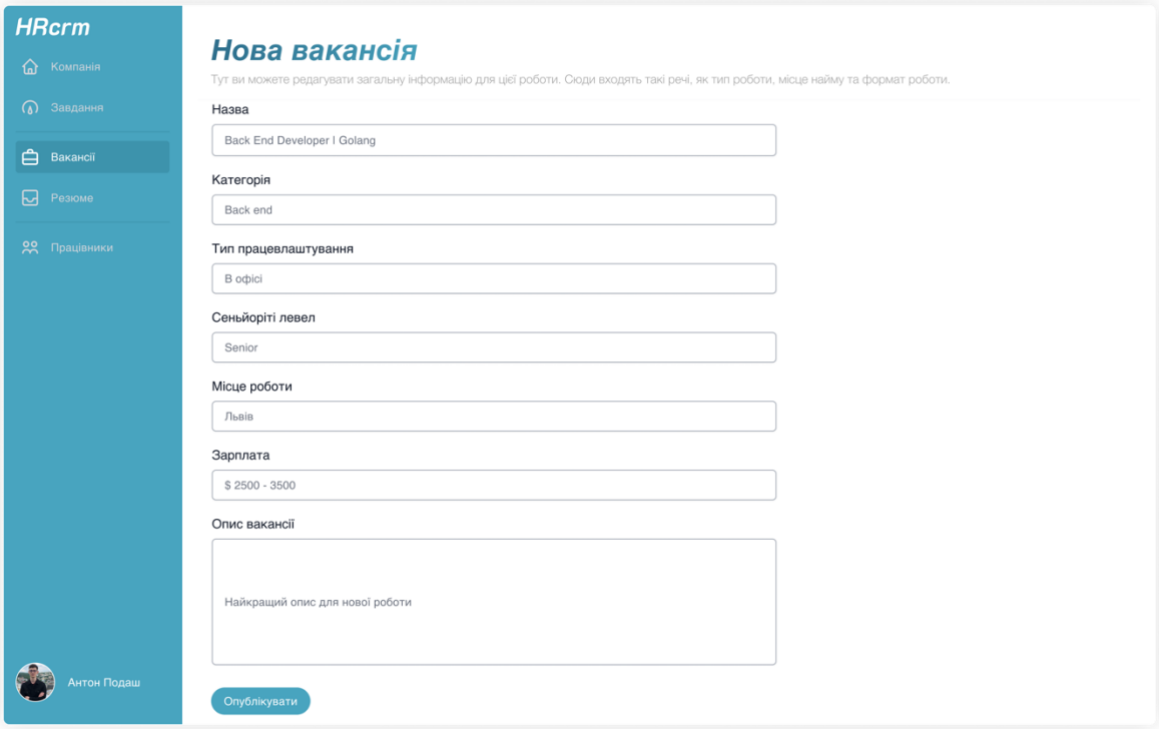


Рисунок 4.3 – Скриншот сторнки додавання вакансії

Основна інформація заповнюється в окремих полях, для подальшої можливості відфільтрувати необхідні вакансії за певними параметрами. Загальний опис вакансії вноситься в найбільше поле.

4.1.4 Сторінка списку вакансій

Загальний список вакансій компанії у списковому відображені.

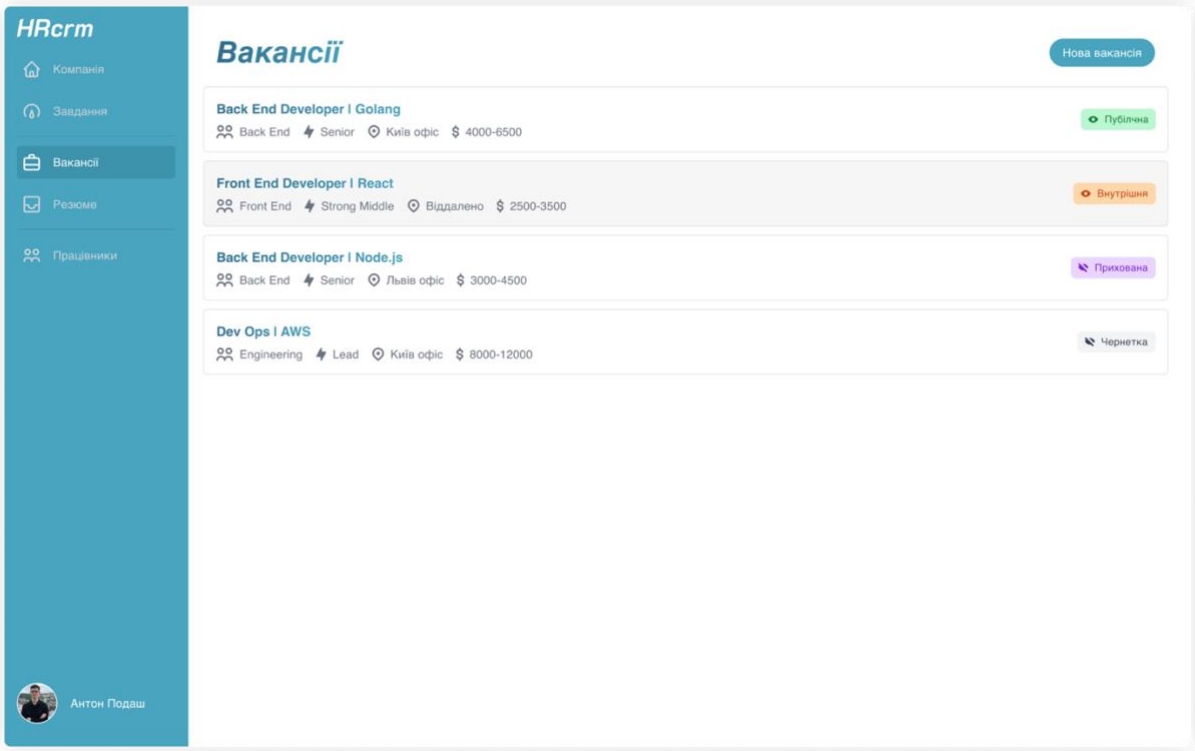


Рисунок 4.4 – Скриншот сторнки вакансій

Список вакансій представлений у вигляді простих карточок з основною інформацією про вакансію. Навпроти кожної вакансії є її статус, який відповідає за її видимість та актуальність.

4.1.5 Сторінка створення нового завдання

Модальна сторінка для створення нового завдання в системі.

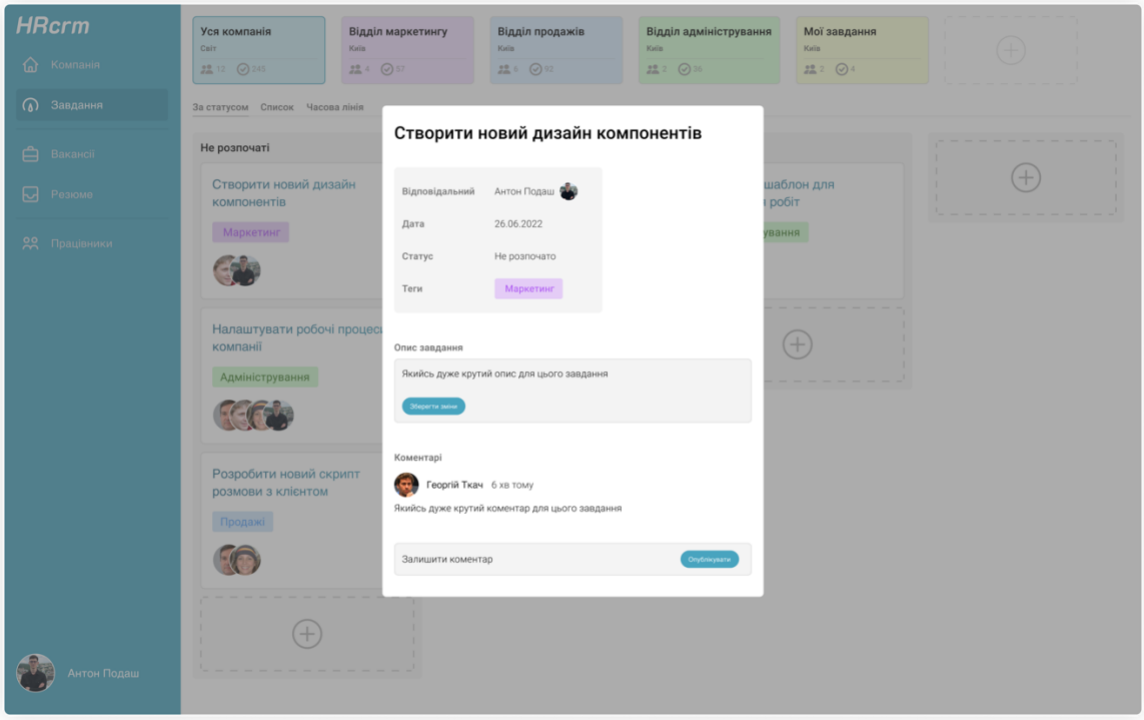


Рисунок 4.5 – Скриншот сторінки нового завдання

Нове завдання створюється в один клік, після чого необхідно заповнити обов’язкову інформацію про завдання, призначити відповідального, поставити крайній термін виконання завдання та опублікувати.

4.1.6 Сторінка списку всіх завдань

Загальна стоірнка з завданнями у канбан вигляді. Відображення завдань на цій сторінці є індивідаульним.

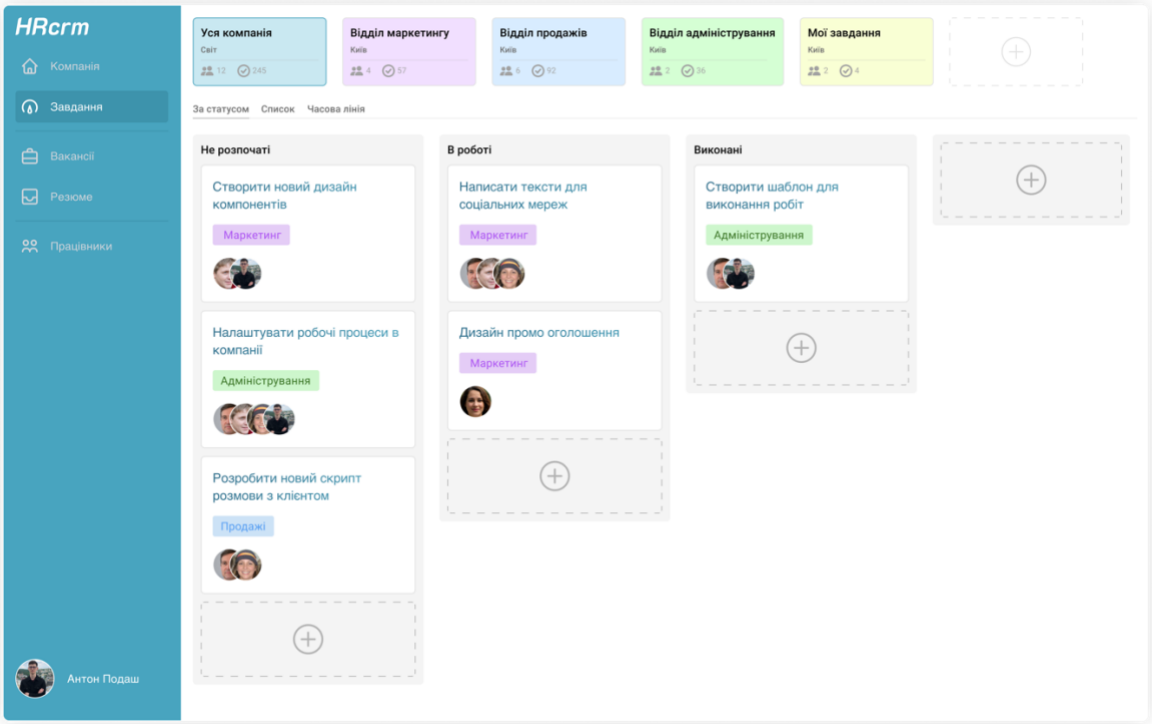


Рисунок 4.6 – Скриншот сторінки завдань

Користувач можу змінити стиль відображення на список або вибрати завдання конкретного відділу, якщо він є його членом. Адміністратори компанії, мають доступ до завдань всіх працівників компанії.

4.1.7 Сторінка працівників

Загальний вигляд та структура компанії, яка відображена на сторінці з працівниками.

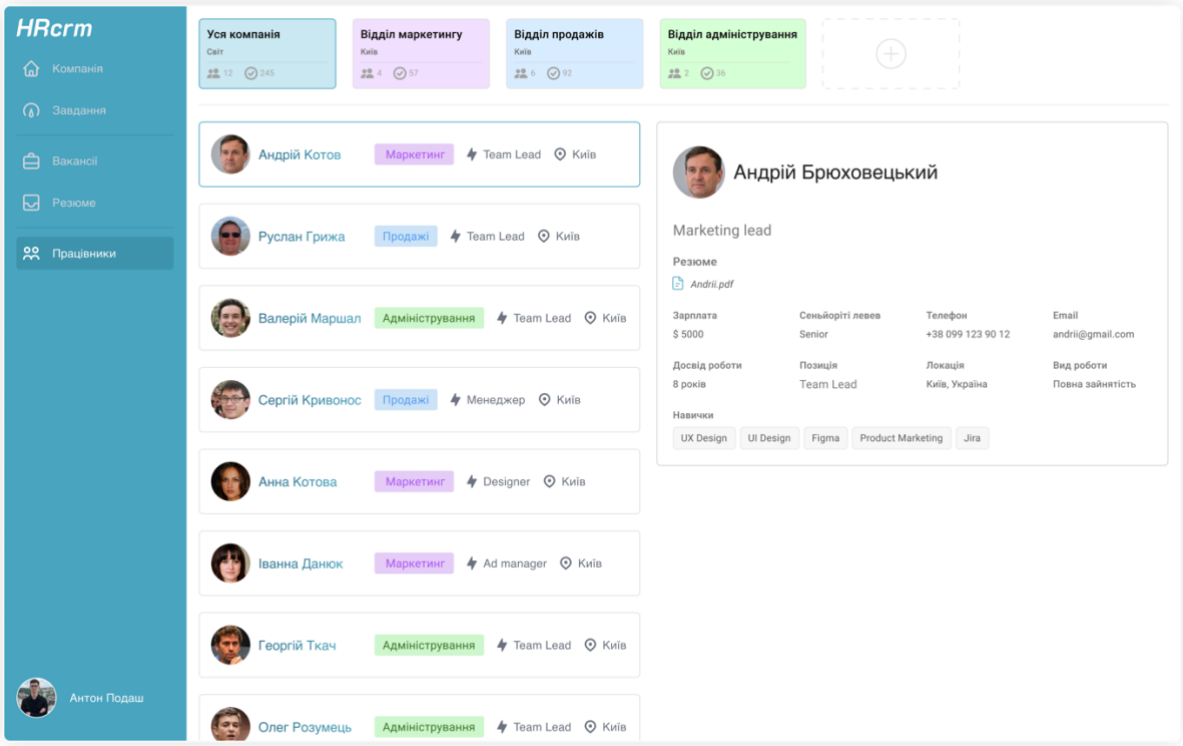


Рисунок 4.7 – Скриншот сторінки працівників

Кожна карточка працівника являє собою його резюме з необхідною інформацією. Також можна поділити праівників на відділи та команди в середині компанії, що дає змогу полегшити роботу керівникам.

4.2 Тестування системи

4.2.1 Основний мотив

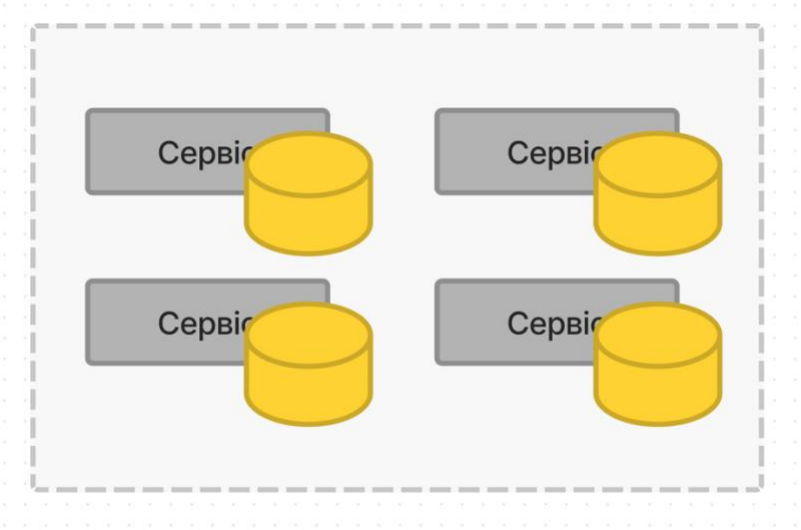


Рисунок 4.8 – Модель мікросервісів

Можливість незалежно розробляти, розгортати та масштабувати різні бізнес-функціональні можливості є однією з найбільш рекламованих переваг прийняття архітектури мікросервісів. Хоча населення все ще не знає, чи справді ці переваги реалізуються чи ні, мікросервіси дуже в моді, аж до такої міри, коли ця архітектура стала стандартною навіть для багатьох стартапів. Але коли справа доходить до тестування (або, що ще гірше, при розробці) мікросервісів, більшість організацій, здається, досить прив'язані до допотопної моделі тестування всіх компонентів унісон. Розроблені інфраструктури конвеєрів тестування вважаються обов'язковими, щоб забезпечити цю форму наскрізного тестування, коли набір тестів кожної служби виконується, щоб підтвердити, що немає жодних регресій або аварійних змін.

4.2.2 Спектр тестування

Історично склалося так, що тестування включало в себе діяльність перед виробництвом або перед випуском. Деякі компанії наймали — і продовжують використовувати — спеціальні групи тестувальників або інженерів з контролю якості, єдиною відповідальністю яких було виконання ручних або автоматизованих тестів програмного забезпечення, створеного командами розробників. Після того, як частина програмного забезпечення пройшла контроль якості, вона була передана команді операцій для запуску (у випадку послуг) або відправлена як випуск продукту (у випадку програмного забезпечення для настільних комп'ютерів чи ігор і що у вас є). Команди розробників тепер відповідають за тестування, а також за роботу служб, які вони створюють. Ця нова модель — те, що я вважаю неймовірно потужною, оскільки вона дійсно дозволяє командам розробників думати про масштаби, цілі, компроміси та виграші всього спектру тестування у реалістичний і стійкий спосіб. Для того, щоб мати можливість розробити цілісну стратегію для розуміння того, як функціонують наші послуги та отримати впевненість у їх правильності, важливо мати можливість вибрати правильний набір методів тестування, враховуючи вимоги щодо доступності, надійності та коректності служби. . Загалом, «тестування» можна використовувати для охоплення різноманітних видів діяльності, включаючи багато практик, які традиційно підпадали під парасольку «інженерних релізів» або операцій або забезпечення якості. Деякі з методів, перерахованих на ілюстрації нижче, не розглядаються строго як форми тестування в деяких колах — наприклад, офіційне визначення інженерії хаосу класифікує її як форму експерименту — і я також не маю на меті вдавати, що це всеосяжний. Список — тестування безпеки (тестування вразливості, тестування на проникнення, моделювання загроз тощо) взагалі ніде не представлено, для початку — але він охоплює деякі з найпоширеніших форм тестування, які зустрічаються в дикій природі.

4.2.3 Типи тестування

Рисунок 4.9 – Піраміда тестування

Під час розробки архітектури мікросервісів можна запустити кілька тестів для перевірки своєї програми, включаючи модульне тестування, тестування контрактів, інтеграційне тестування, наскрізне тестування та UI/функціональне тестування.

4.2.4 Модульне тестування

Цей тип тестування допомагає перевірити продуктивність кожного компонента (або блоку) вашого програмного забезпечення. Одиницею може бути об'єкт, модуль, процедура, функція або метод. Ви можете визначити розмір одиниці, але майте на увазі, що менші одиниці найпростіше перевірити. Крім того, якщо вам важко визначити модульний тест, вважайте це ознакою того, що ваш модуль слід розбити на менші частини.

В ідеалі розробники повинні проводити модульне тестування на етапі розробки, щоб переконатися, що всі компоненти коду працюють належним чином і як розроблено. Однак під час швидких розробок або коли команди працюють на окремих етапах, команді забезпечення якості (QA) може знадобитися провести модульне тестування — після того, як розробники завершили роботу над проектом.

4.2.5 Тестування контракту

Щоб забезпечити взаємодію служб, контракт визначає очікувані результати або результати для конкретних вхідних даних. Тестування договору споживача гарантує, що кожен виклик служби відповідає цьому контракту, і навіть якщо сама послуга змінюється всередині, споживачі продовжуватимуть отримувати ті самі результати від цієї служби.

Таким чином, сама послуга залишається чорною скринькою. Для перевірки контракту служби викликаються незалежно, а відповіді перевіряються. Якщо існують залежності служби, вони повинні працювати як заглушки, які дозволяють службі функціонувати, але без взаємодії з іншими службами. Це також запобіжить непостійну поведінку, спричинену зовнішніми викликами, зосередившись на тестуванні однієї служби.

4.2.6 Ітеграційне тестування

Інтеграційний тест збирає мікросервіси та тестує їх разом як підсистему. Мета полягає в тому, щоб переконатися, що мікросервіси співпрацюють для досягнення спільної операції. Під час тесту відпрацьовуються шляхи зв’язку, що дає змогу виявити неправильні припущення в кожній мікросервісі щодо взаємодії з однолітками.

Тестування інтеграції також може допомогти вам перевірити взаємодію між мікросервісами та зовнішніми компонентами. Наприклад, ви можете використовувати цей тип тесту, щоб переконатися, що зовнішні сховища даних і кеші належним чином інтегровані, відповідають і працюють належним чином.

4.2.7 Наскрізне тестування

Основна мета наскрізного тестування полягає в тому, щоб переконатися, що вся програма працює для досягнення своїх бізнес-цілей, незалежно від компонентів, що утворюють всю архітектуру. Під час наскрізного тестування програма розглядається як чорний ящик, а тести тренують систему, оскільки вона повністю розгорнута.

Наскрізне тестування особливо важливе для архітектури мікросервісів, яка складається з багатьох частин, які намагаються досягти подібної поведінки. Наскрізне тестування може допомогти вам переконатися, що між мікросервісами немає розривів і всі вони працюють над досягненням своєї мети. В ідеалі тести повинні підтверджувати, що повідомлення передаються правильно між службами, і що мережева інфраструктура, як-от брандмауери та проксі, правильно налаштована.

ВИСНОВОК ДО РОЗДІЛУ 4

В даному розділі було повністю оглянуто всю систему. Зроблений детальний огляд інтерфейсу кожної сторінки системи.

Реалізована система виконує наступні функції:

- Управління завданнями
- Аналіз продуктивності працівників
- Ведення резюме нових кандидатів
- Зручний перегляд актуальних вакансій компанії
- Відображення всіх працівників компанії та окремих відділів
- Можливість планування співбесіди з кандидатом

ВИСНОВКИ

В процесі даної роботи було зроблено детальний аналіз існуючих систем на ринку, які б задовільняли усіх технічні вимоги та потребності користувачів у цій сфері. Були виявлені головний спектр функціоналу подібних систем, який був взятий до уваги при розробці.

Наступною частиною роботи був детальний огляд та вибір технологічного стеку та платформ для розробки системи. Головними параметрами при їх підборі були сучасність та продуктивність. В результаті система отримала мікросервісну архітектуру, серверну розробку на Express.js та фронтенд розробку на Next.js.

Слідуючим етапом роботи є безпосередня розробка системи. Спочатку було сформовано архітектуру, прописані усі юзер сторіс майбутніх користувачів та спроектована база данних під мікросервіси.

Розроблена система повністю відповідає сучасним стандартам розробики прогресивних веб-програм та має обширний функціонал для використання з можливістю його розширення за потреби

Під час виконання роботи були використані знання та вміння в галузі розробки програмного забеспечення

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sage human resource management system (HRMS) [Electronic resource] // sage.com. – Режим доступу: https://www.sage.com/en-ca/products/sage-hrms/?r=pmp-hrms (дата доступу: 13.06.2022). – Заголовок з екрану.
2. Ascender CONNECTING PEOPLE AND PAYROLL Payroll & HCM Solutions [Electronic resource] // ascenderhcm.com. – Mode of access: https://www.ascenderhcm.com. – Заголовок з екрану.
3. Autili M. A hybrid approach to microservices load balancing [Electronic resource] / Marco Autili, Alexander Perucci, Lorenzo De Lauretis // Microservices. – Cham, 2019. – P. 249–269. – Режим доступу: https://doi.org/10.1007/978-3-030-31646-4_10(дата доступу: 13.06.2022). – Заголовок з екрану.
4. Garneau W. Human resources [Electronic resource] / Wallace Garneau // The way forward. – [S. l.], 2021. – P. 117–124. – Режим доступу: https://doi.org/10.4324/9781003165910-11(дата доступу: 13.06.2022). – Заголовок з екрану.
5. Hajian M. Debugging and measurement tools [Electronic resource] / Majid Hajian // Progressive web apps with angular. – Berkeley, CA, 2019. – P. 255–282. – Режим доступу: https://doi.org/10.1007/978-1-4842-4448-7_10(дата доступу: 13.06.2022). – Заголовок з екрану.
6. Hajian M. Modern Web APIs [Electronic resource] / Majid Hajian // Progressive web apps with angular. – Berkeley, CA, 2019. – P. 289–330. – Режим доступу: https://doi.org/10.1007/978-1-4842-4448-7_12(дата доступу: 13.06.2022). – Заголовок з екрану.
7. Hajian M. Push notifications [Electronic resource] / Majid Hajian // Progressive web apps with angular. – Berkeley, CA, 2019. – P. 201–227. –

Режим доступу: https://doi.org/10.1007/978-1-4842-4448-7_8(дата доступу: 13.06.2022). – Заголовок з екрану.

8. Hajian M. Safety service worker [Electronic resource] / Majid Hajian // Progressive web apps with angular. – Berkeley, CA, 2019. – P. 283–288. – Режим доступу: https://doi.org/10.1007/978-1-4842-4448-7_11(дата доступу: 13.06.2022). – Заголовок з екрану.
9. Hajian M. Setup requirements [Electronic resource] / Majid Hajian // Progressive web apps with angular. – Berkeley, CA, 2019. – P. 1–8. – Режим доступу: https://doi.org/10.1007/978-1-4842-4448-7_1(дата доступу: 13.06.2022). – Заголовок з екрану.
10. Hudson W. User stories don't help users [Electronic resource] / William Hudson // Interactions. – 2013. – Vol. 20, no. 6. – P. 50–53. – Режим доступу: https://doi.org/10.1145/2517668(дата доступу: 13.06.2022). – Заголовок з екрану.
11. Hume D. A. Progressive web apps / Dean Alan Hume. – [S. l.] : Manning Publications, 2017. – 200 p.
12. Inc B. P. Flow charts / Bergwall Productions Inc. – [S. l.] : Delmar Pub, 1988.
13. Kokol P. User interface metrics [Electronic resource] / Peter Kokol, Ivan Rozman, Vlado Venuti // ACM SIGPLAN Notices. – 1995. – Vol. 30, no. 4. – P. 36–38. – Режим доступу: https://doi.org/10.1145/202176.202180(дата доступу: 13.06.2022). – Заголовок з екрану.
14. Larson J. A. Principles for mujltimodal user inteface desgin [Electronic resource] / James A. Larson, Sharon Oviatt // CHI '03 extended abstracts, Ft. Lauderdale, Florida, USA, 5–10 April 2003 – New York, New York, USA, 2003. – Режим доступу: https://doi.org/10.1145/765891.766147(дата доступу: 13.06.2022). – Заголовок з екрану.
15. Mardan A. Abstraction [Electronic resource] / Azat Mardan // Pro express.js. – Berkeley, CA, 2014. – P. 155–160. – Режим

доступу: https://doi.org/10.1007/978-1-4842-0037-7_10(дата доступу: 13.06.2022). – Заголовок з екрану.

16. Mardan A. Database, keys and stream tips [Electronic resource] / Azat Mardan // Pro express.js. – Berkeley, CA, 2014. – P. 161–170. – Режим доступу: https://doi.org/10.1007/978-1-4842-0037-7_11(дата доступу: 13.06.2022). – Заголовок з екрану.
17. Mardan A. Security tips [Electronic resource] / Azat Mardan // Pro express.js. – Berkeley, CA, 2014. – P. 185–192. – Режим доступу: https://doi.org/10.1007/978-1-4842-0037-7_15(дата доступу: 13.06.2022). – Заголовок з екрану.
18. Mardan A. Starting with express.js [Electronic resource] / Azat Mardan // Pro express.js. – Berkeley, CA, 2014. – P. 3–14. – Режим доступу: https://doi.org/10.1007/978-1-4842-0037-7_1(дата доступу: 13.06.2022). – Заголовок з екрану.
19. Miranda E. Sizing user stories using paired comparisons [Electronic resource] / Eduardo Miranda, Pierre Bourque, Alain Abran // Information and software technology. – 2009. – Vol. 51, no. 9. – P. 1327–1337. – Режим доступу: https://doi.org/10.1016/j.infsof.2009.04.003(дата доступу: 13.06.2022). – Заголовок з екрану.
20. M V. A. ReactJS by example - building modern web applications with react / Vipul A. M, Prathamesh Sonpatki. – [S. l.] : Packt Publishing, 2016. – 280 p.
21. Professional NodeJs. – [S. l.] : Wrox Press, 2012.
22. Pub U. I. a. U. E. N. Mobile UX/UI design notebook: app mobile wireframe sketchpad user interface experience application development note book developers app mock ups 6 X 9 inches with 100 pages / User Interface and User Experience Notebook Pub. – [S. l.] : Independently Published, 2020. – 100 p.
23. Securing microservices [Electronic resource] / Antonio Nehme [et al.] // IT professional. – 2019. – Vol. 21, no. 1. – P. 42–49. – Режим

доступу: https://doi.org/10.1109/mitp.2018.2876987(дата доступу: 13.06.2022). – Заголовок з екрану.

24. Shaik B. User management and securing databases [Electronic resource] / Baji Shaik // PostgreSQL configuration. – Berkeley, CA, 2020. – P. 61–92. – Режим доступу: https://doi.org/10.1007/978-1-4842-5663-3_3(дата доступу: 13.06.2022). – Заголовок з екрану.
25. Sheppard D. Introduction to progressive web apps [Electronic resource] / Dennis Sheppard // Beginning progressive web app development. – Berkeley, CA, 2017. – P. 3–10. – Режим доступу: https://doi.org/10.1007/978-1-4842-3090-9_1(дата доступу: 13.06.2022). – Заголовок з екрану.
26. Sinnig D. Mastering use cases [Electronic resource] / Daniel Sinnig, Homa Javahery // The 2nd ACM SIGCHI symposium, Berlin, Germany, 19–23 June 2010 – New York, New York, USA, 2010. – Режим доступу: https://doi.org/10.1145/1822018.1822089(дата доступу: 13.06.2022). – Заголовок з екрану.
27. System test cases from use cases [Electronic resource] // 1st international conference on software and data technologies, Setúbal, Portugal, 11–14 September 2006 – [S. l.], 2006. – Режим доступу: https://doi.org/10.5220/0001309302830286(дата доступу: 13.06.2022). – Заголовок з екрану.
28. Teixeira P. Professional node. js: building javascript based scalable software / Pedro Teixeira. – [S. l.] : Wiley & Sons, Incorporated, John, 2012. – 408 p.
29. Teixeira P. Professional node. js: building javascript based scalable software / Pedro Teixeira. – [S. l.] : Wiley & Sons, Incorporated, John, 2012. – 408 p.
30. Testing nodejs web uis. – [S. l.] : Packt Publishing Limited, 2013.
31. Thones J. Microservices [Electronic resource] / Johannes Thones // IEEE software. – 2015. – Vol. 32, no. 1. – P. 116. – Режим доступу: https://doi.org/10.1109/ms.2015.11(дата доступу: 13.06.2022). – Заголовок з екрану.

32. Visan A. React JS authentication login [Electronic resource] / Andrei Visan // Front-End Development with iPad Pro and Raspberry Pi 4. – Berkeley, CA, 2022. – Режим доступу: https://doi.org/10.1007/978-1-4842-8061-4_2(дата доступу: 13.06.2022). – Заголовок з екрану.
33. Visan A. React JS authentication registration [Electronic resource] / Andrei Visan // Front-End Development with iPad Pro and Raspberry Pi 4. – Berkeley, CA, 2022. – Режим доступу: https://doi.org/10.1007/978-1-4842-8061-4_3(дата доступу: 13.06.2022). – Заголовок з екрану.
34. Wadhwa B. Converting epics into user stories in Agile [Electronic resource] / Bhawna Wadhwa, Pawan Bhadana // Global Sci-Tech. – 2019. – Vol. 11, no. 2. – P. 75. – Режим доступу: https://doi.org/10.5958/2455-7110.2019.00011.9(дата доступу: 13.06.2022). – Заголовок з екрану.
35. Walker A. Principles of beautiful web design / Alex Walker, Jason Beaird. – [S. l.] : SitePoint Pty, Limited, 2020. – 230 p.

ДОДАТОК 1

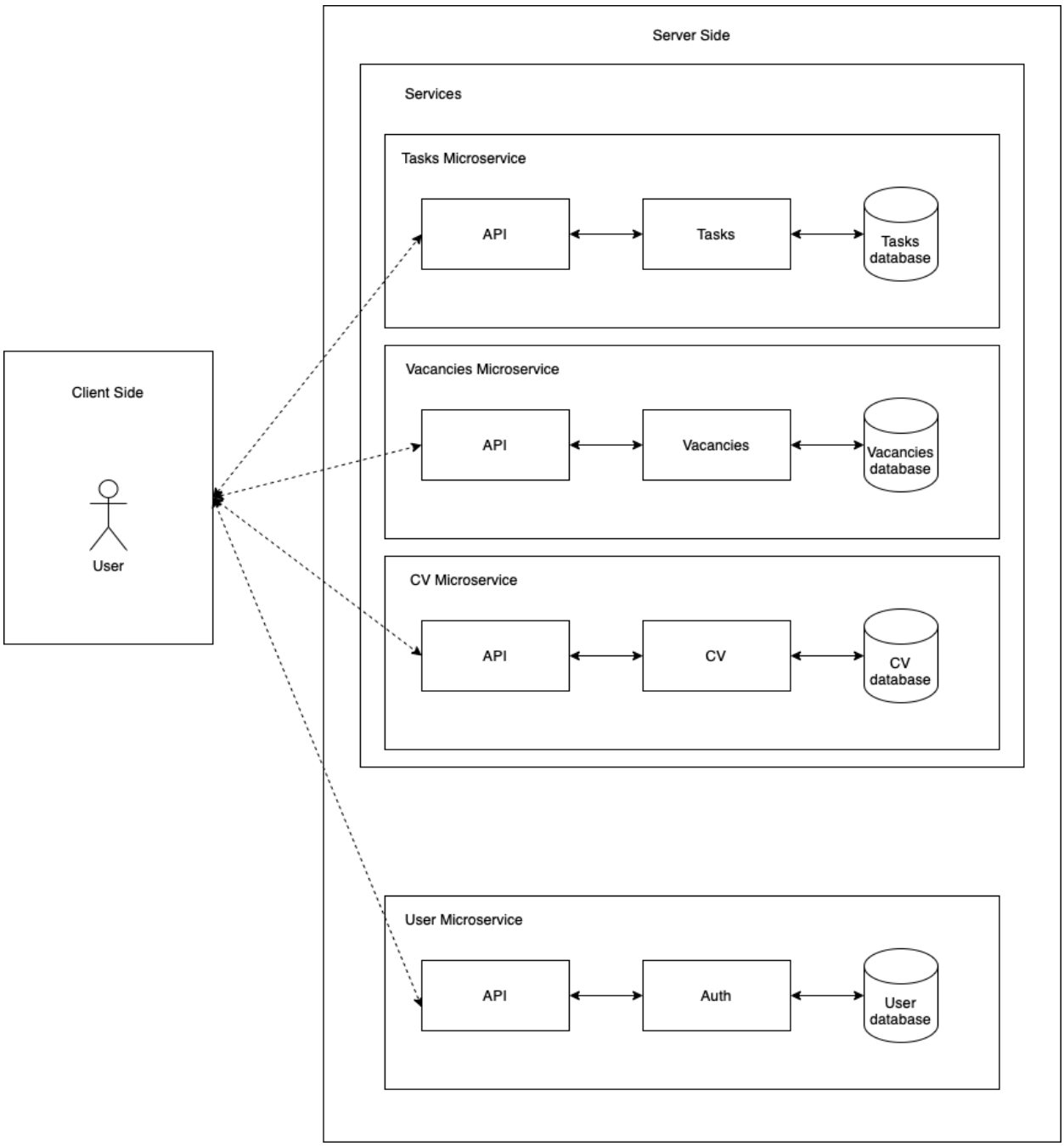
Еволюційні алгоритми глобальної пошукової оптимізації

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2022 р.



					ІАЛЦ.467200.004 Д1	
		№ докум.	Підпис	Дата		
Розробив		Подаш А.М,			**HRCRM – система для управління персоналом** Структурна схема даних	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Волокита І. М.				
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІП-73
Н. Контр.		Сімоненко В. П.				
Затвердив						

ДОДАТОК 2

HRCRM – система управління персоналом

Алгоритм дій програмного забезпечення

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2022 р.

					ІАЛЦ.467200.005 Д2	
		№ докум.	Підпис	Дата		
Розробив		Подаш А.М.			**HRCRM – система для управління персоналом** Алгоритм дії ПЗ	Літ. / Аркуш / Аркушів: 1 / 1
Перевірив		Волокита І. М.				
Н. Контр.		Сімоненко В. П.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІП-73
Затвердив						

ДОДАТОК 3

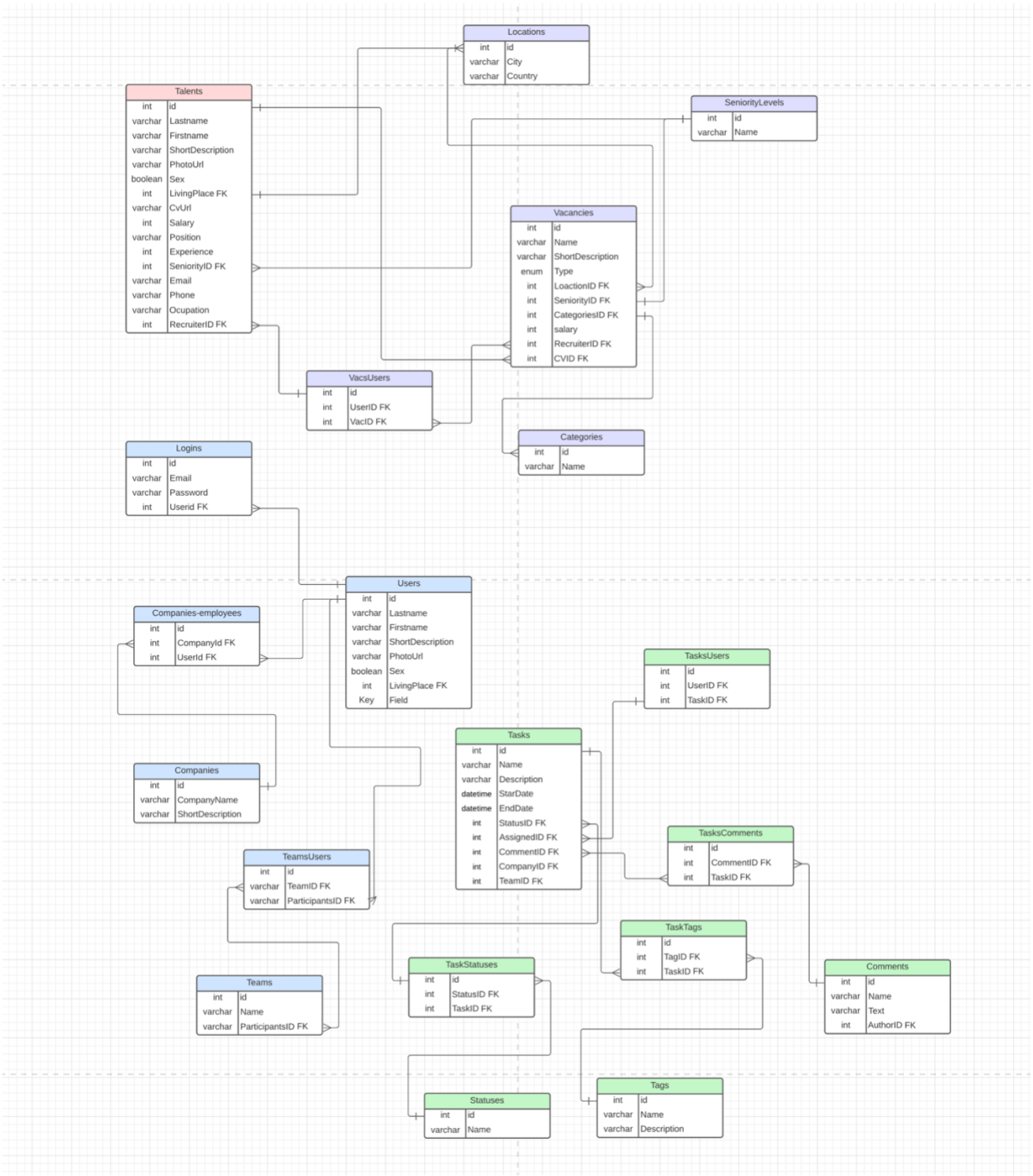
HRCRM – система управління персоналом

Функціональна схема

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2022 р.



					ІАЛЦ.467200.006 Д3	
		№ докум.	Підпис	Дата		
Розробив		Подаш А.М.			**HRCRM – система для управління персоналом** Функціональна схема	Літ. / Аркуш / Аркушів: 1 / 1
Перевірив		Волокита І. М.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІП-73
Н. Контр.		Сімоненко В. П.				
Затвердив						

ДОДАТОК 4

HRCRM – система управління персоналом

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 41

Київ 2022 р.