

**ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА”**

Навчально - науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

**Козирев Андрій Валерійович**

**ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ ЛЮДСЬКИМИ  
РЕСУРСАМИ**

кваліфікаційна робота  
здобувача вищої освіти першого (бакалаврського) рівня  
освітньої програми «Комп'ютерні науки та інформаційні технології»  
за спеціальністю 122 „Комп'ютерні науки ”

Особистий підпис

Науковий керівник

Владислав КОЗУБ, д-р філософії

В.о.завідувача кафедри

Ліна БОНДАРЕНКО, к.пед.н.

Полтава – 2024

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

## 6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» лютого 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи )	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	до 1 лютого	
2.	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	другий тиждень лютого	
3.	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	до 1 квітня	
4.	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	до 15 квітня	
5.	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	перший тиждень квітня	
6.	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	до 20 травня	
7.	Попередній захист роботи на кафедрі	Травень	
8.	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
9.	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Студент

Керівник роботи

\_\_\_\_\_

А.В.Козирев

В.Ю.Козуб

\_\_\_\_\_

## АНОТАЦІЯ

В останні роки завдяки розвитку нових технологій, збільшення кількості компаній, спостерігається зростання потреби в автоматизованих системах управління співробітниками, а також оцифрування документації та автоматизація управлінських операцій.

Метою цієї роботи є аналіз, проектування та реалізація веб-системи управління людськими ресурсами.

Ця система спрямована на управління співробітниками і автоматизація процесів організації. Спочатку проводиться аналіз концепції системи управління персоналом (Human Resource система управління, HRMS) і потреби, які вона повинна задовольнити. Важливим чинником являється використання інструментарію розробки таких систем, зокрема, вибір мов програмування, бібліотек і технологій, які будуть застосовані для його реалізації. При створенні системи управління застосований підхід на основі проектування системи за допомогою діаграм UML. Проведено аналіз можливостей та функцій розробленої систем, а також можливості реалізації різних сценаріїв використання розробленого додатку.

Ключові слова: система управління людськими ресурсами, HRMS, організації, співробітники, процедури

## ABSTRACT

In recent years, thanks to the development of new technologies, the increase in the number of companies, there has been a growing need for automated employee management systems, as well as digitization of documentation and automation of management operations.

The purpose of this work is to analyze, design and implement a web-based human resources management system.

This system is aimed at managing employees and automating organizational processes. First, an analysis of the concept of the personnel management system (Human Resource Management System, HRMS) and the needs it must satisfy is carried out. An important factor is the use of tools for the development of such systems, in particular, the choice of programming languages, libraries and technologies that will be used for its implementation. When creating the management system, an approach based on system design using UML diagrams was applied. An analysis of the capabilities and functions of the developed system was carried out, as well as the possibility of implementing various scenarios of the use of the developed application.

Keywords: human resources management system, HRMS, organizations, employees, procedures

## СКОРОЧЕННЯ

HRMS – Система управління людськими ресурсами

HRIS – Human Resource Information System

JS – JavaScript

UML – Уніфікована мова моделювання

ERD – Entity Relationship Diagram

HTTP – Протокол передачі гіпертексту

API – Програмований інтерфейс прикладної програми

JSON – Об'єкт JavaScript Notation

ORM – Object Relational Mapping

## Зміст

ВСТУП .....	8
РОЗДІЛ 1 Теоретичні основи розробки .....	9
РОЗДІЛ 2 Аналіз сценарію .....	12
2.1 Аналіз і дизайн .....	12
2.1.1 Діаграма варіантів використання .....	12
2.1.2 Діаграма класів .....	15
2.1.3 Функціональні пакети .....	18
2.2.4 Діаграма розгортання .....	20
2.2 API .....	23
2.3 Функціональні та нефункціональні вимоги .....	23
РОЗДІЛ 3 Реалізація програми .....	25
3.1 Node.js .....	25
3.2 React.js .....	27
3.3 Express.js .....	28
3.4 MySQL .....	28
3.5 Sequelize .....	29
3.6 Bootstrap .....	29
3.7 React Material Table .....	30
3.8 Веб-токени JSON .....	30
3.9 Додаткові модулі .....	32
РОЗДІЛ 4: Сценарії використання .....	33
4.1 Загальні положення .....	33
4.2 Адміністратор (Admin) .....	35
4.3 Менеджер .....	39
4.4 Співробітник .....	40
ВИСНОВКИ .....	41
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	42

## ВСТУП

Об'єктом роботи є створення веб - додатку, який використовує сучасні технології програмування, такі як мова Javascript. Ідея програми полягає в тому, щоб розробити веб-сайт для менеджерів, який автоматизує управління персоналом для працівників відділу кадрів та секретаріату. Крім того, надасть можливість для інших членів компанії/організації входити в систему виконувати різні функції та мати доступ до різних інформацію залежно від рівня доступу. Ці категорії рівні доступу поділяються на Адміністратор, Директор (Менеджер) і простий Працівник (Працівник).

Під «керівництвом» працівників компанії/організації мається на увазі управління інформацією, якою володіє компанія про них, надаючи простий спосіб перегляду, додавання, змінювання та зберігання даних.

Також система працює як інструмент управління заробітною платою та витратами є можливість зберігати дані про витрати компанії та платежі відображення місячних/річних витрат/платежів.



## РОЗДІЛ 1 Теоретичні основи розробки

HRMS (системи управління людськими ресурсами) або HRIS (інформаційна система про людські ресурси) — це програмні системи, які керують частиною або всіма операціями з людськими ресурсами організації/компанії. Для забезпечення працездатності HRMS та здатності підприємства підтримувати правильну, повну і оновлену базу даних, необхідно враховувати загальні елементи системи управління: демографію персоналу, управління оплатою праці, премії, відпустки та співбесіди, навчання, нарахування заробітної плати та звітність про результати діяльності [4].

У будь-якому випадку метою є забезпечення інформації до відділу кадрів та підтримки рішень, які вони пов'язані з людськими ресурсами [1,4,5].

Інформаційні системи управління людськими ресурсами або HRIS (Human Resource Information Systems) — це системи, на основі яких можна збирати, зберігати, використовувати, аналізувати та розповсюджувати інформацію, що стосується своїх людських ресурсів; організації [6].

Інформаційні системи людських ресурсів — це синтез даних та ІТ-додатків, апаратне та програмне забезпечення, необхідне для збирання, запису, зберігання, управління, розповсюдження, презентації та використання даних, що стосуються людських ресурсів [3].

Відповідно до Kettley and Reilly (2003) потенціал інформаційної системи людини складається з повністю інтегрованої мережі в діапазоні однієї організації з даних, інформації, послуг, баз даних, інструментів і операцій з людськими ресурсами. Якщо ці транзакції здійснюються за допомогою Інтернета ми говоримо про «електронний HR» або «e-HR», який часто включає застосування звичайних, мережових і голосових технологій, порталів і веб-додатків [7,8].

Стрімкий розвиток інформаційних систем за останні п'ять десятиліть неминуче вплинув на підприємства, які використовують цю технологію як засіб, що підвищить їх дієвість та ефективність. Так, все більше і більше компаній також застосовують інформаційні системи в менеджменті відділу людських ресурсів.

Нижче наведено деякі переваги, які компанія отримає від використання інформаційних систем людських ресурсів:

- Легкий доступ до персональних даних та інформації.
- Автоматизація процедур і скорочення бюрократії. Таким чином, операції здійснюються горизонтально і стають більше гнучкими шляхом усунення вертикальної структури, яка викликає бюрократію.[9]
- Більш точні та конкретні дані з меншою ймовірністю помилки.
- Швидший доступ і обробка за допомогою передових технологій у зручний спосіб для працівника, який подає заяву, наприклад, для інформації, а також у більш швидке прийняття адміністративних рішень.
- Створення «організаційної пам'яті» в бізнесі, яка буде зберігати інформація про всіх співробітників (навіть тих, що звільнилися) і зможе передбачити майбутні кар'єрні зміни співробітників, виділити виняткові серед «середніх» працівників, сильні сторони і слабкі сторони керівників і потреби в навчанні або їх потреби компанія для комплектування кадрів. Вона може обнулювати відстані, що робить можливим редагування дані всіх співробітників, навіть якщо компанія має філії в багатьох регіонах. Розподілені бази даних і комунікаційні технології дозволяють надавати інформацію багатьом частин одночасно. [9]
- Сучасні HRMS є частиною інтегрованої ІТ системи, яку використовує бізнес.
- Надання сучасних послуг та створення сучасного іміджу бізнесу, що робить його привабливим на ринку праці для існуючого і потенційного персоналу.

- Зниження трудовитрат за рахунок скорочення часу виконання, або скасування певних процедур.

- Також зміна стосунків між співробітниками та керівниками зміна взаємовідносин лінійних керівників з відділом кадрів. Тепер працівники можуть бути частиною керівництва та відділу кадрів для продуктивної співпраці з іншими керівниками.

- Підвищення ролі HR-менеджера та визнання її важливості.

- Удосконалення послуг, що надаються відділом кадрів, що походить від знання даних, а не володіння ними.

- Створення конкурентної переваги для бізнесу, враховуючи це людський капітал є найважливішим ресурсом бізнесу та поодиноці, яку важко «скопіювати» конкуруючим компаніям.

- Перенесення центру ваги з традиційних операцій на стратегію управління людськими ресурсами. Відділ кадрів звільняється від більш тривіальних процедур і забезпечує час для займатися стратегічними планами.

## **РОЗДІЛ 2 Аналіз сценарію**

### **2.1 Аналіз і дизайн**

Для аналізу та дизайну програми використовувалася мова UML. Згідно з офіційним довідником, Unified Modeling Language (Unified Modeling Language або UML) — це графічна мова загального призначення, яка використовується для ідентифікації, візуалізації, розробки та документування програмної системи. UML – це стандарт для моделювання програмних систем і використовується в моделювання об'єктно-орієнтованих систем (об'єктно-орієнтованих системи) [11].

#### **2.1.1 Діаграма варіантів використання**

Будь-який користувач, який увійшов у середовище програми, може виконувати різні функції та отримувати доступ до різних видів інформації залежно від його ролі.

Щоб представити ці функції, було створено сценарій використання системи.

Системою можуть користуватися лише авторизовані особи користувачів, яких ми поділяємо на 3 категорії

- Адміністратор/Admin (Співробітник відділу кадрів)
- Директор/Manager
- Простий Співробітник/Employee

Спочатку неавторизованому користувачеві дозволено отримати доступ в системі після реєстрації. Зокрема, є функція реєстрації Register.

Функція реєстрації призначена для створення облікових записів співробітників.

Запропоновано функцію додавання співробітника Адміністратором системи для створення облікового запису для Адміністратора та/або Менеджера.

Після створення облікового запису користувач повинен дочекатися активації його облікового запису Адміністратором. У разі створення облікового запису Адміністратором, обліковий запис уже активовано.

Цей крок було включено, оскільки важливо перевірити дані для перевірки особи користувача та підтвердження того, що він є співробітником організації/компанії.

Функція входу пов'язана з функціями перевірки пароля та відображається помилка підключення. Підключення функції Логін і Підтвердження пароля виконується за допомогою Include, тому що для підключення до системи потрібно пароль має бути успішно перевірений.

Натомість підключення функцій входу та відображення помилок входу здійснюється за допомогою функції Extend (Розширення). Ця кореляція фіксується як функція відображення.

Адміністратор як користувач, для якого розроблена ця система, має доступ у найбільшій кількості системних функцій.

Функції системи виконуються на даних компанії/організації та поділяються на такі категорії:

- Відділи компанії (Відділи);
- Співробітники компанії (Працівники);
- Особиста інформація;
- Фінансові дані;
- Робота працівників (Вакансії);
- Заявки на відпустку працівників;
- Витрати відділів (Витрати);
- Оголошення відділу (Оголошення);

У цих категоріях можуть виконувати різні функції користувачів системи відповідно до їх ролі

#### *Адміністратор*

Системний адміністратор має право виконувати такі функції:

- Перегляд списку розділів;
- Додавання розділу;
- Перегляд списку співробітників;
- Додавання співробітника;
- Перегляд списку вакансій;
- Додавання посади до працівника;
- Перегляд списку запитів на ліцензію;
- Подання заявки на отримання ліцензії;
- Прийняття/відхилення заявки співробітників;
- Перегляд списку зарплат співробітників;
- Керування відомостями про заробітну плату;
- Створення платіжу;
- Додавання витрат;
- Перегляд списку повідомлень;
- Додавання оголошення;
- Створення особистих подій в Календарі.

#### *Менеджер*

Директор товариства має право виконувати наступні функції, що стосуються лише відділу, в якому він розміщений:

- Перегляд списку співробітників відділу;
- Перегляд списку вакансій відділу;
- Додавання посади співробітника відділу;
- Перегляд списку запитів відділу;
- Подання заявки на отримання ліцензії;

- Прийняття/відмова заяв співробітників відділу;
- Додавання витрат відділу;
- Перегляд список повідомлень відділу;
- Додавання оголошення відділу;
- Створення особистих подій в Календарі.

### *Співробітник*

Рядовий працівник компанії має право на наступні функції:

- Подання заявки на отримання ліцензії;
- Перегляд його заявки на отримання ліцензії;
- Перегляд даних працівника;
- Перегляд списку оголошень відділу;
- Створення особистих подій в Календарі.

Оскільки розробнику потрібно краще зрозуміти поняття деяких основних функцій системи, важливо зобразити системні сутності через діаграму класів.

### **2.1.2 Діаграма класів**

На діаграмі класів представлені атрибути сутностей системи.

В даному конкретному випадку це: користувач (User), персональні інформація про користувача (User\_Personal\_Informaon), його фінансова інформація (User\_Financial\_Informaon), особисті дані користувача (Personal\_event), завдання користувача (Jobs), виплати, заявки на відпустку користувачів (Leave\_applicaon), відділи компанії (Відділ), і витрати відділів (Expense).

#### **Користувач**

Для користувача системи зберігається використане ім'я користувача під час входу. Також зберігається персональний код користувача, зашифровано за

допомогою бібліотеки bcrypt. Також є Ім'я користувача та статус його облікового запису, які розділені на активних і неактивних користувачів.

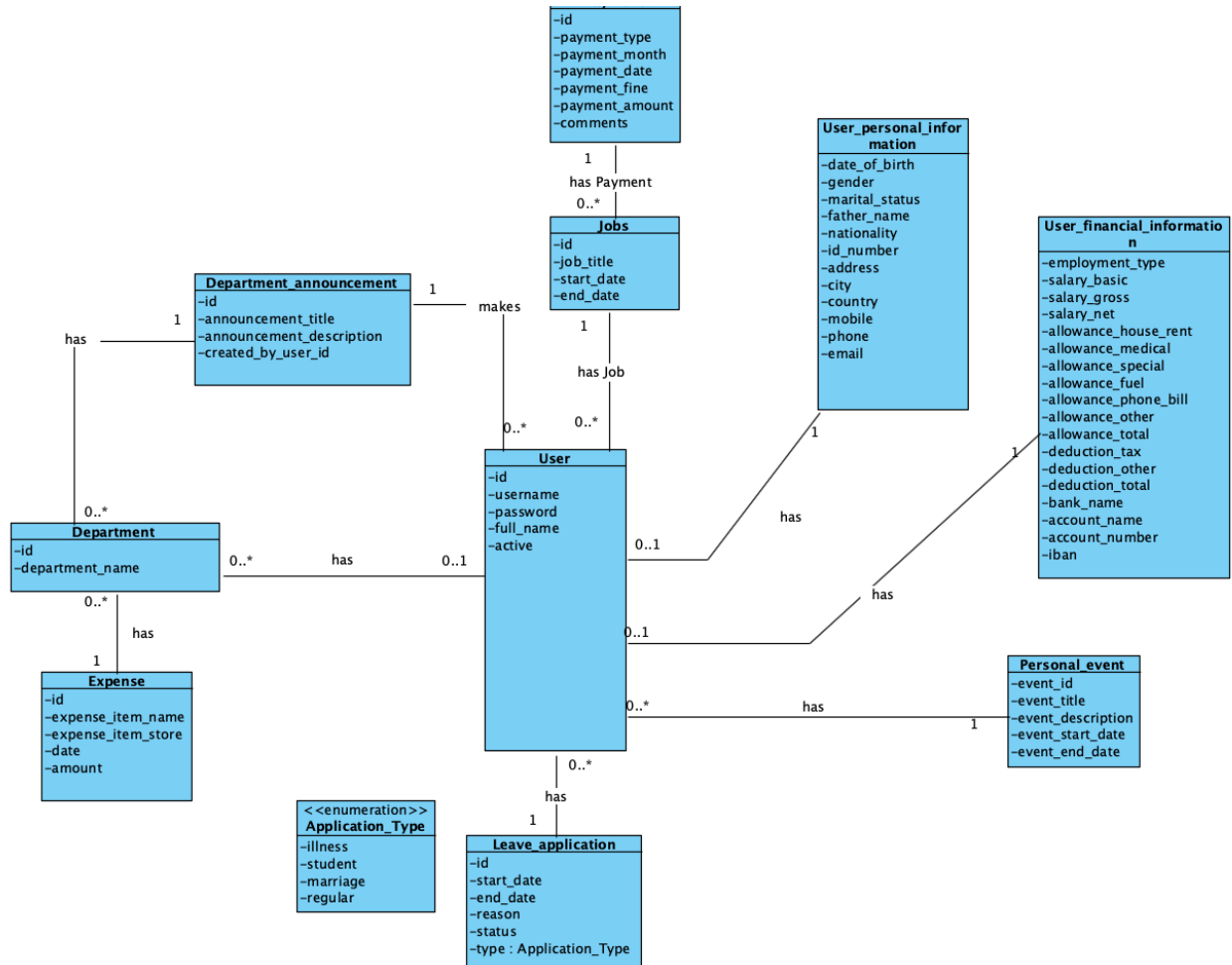


Рис. 1. Діаграма класів

### Особиста інформація користувача

В окремій таблиці та з функцією 1 до 1 з таблицею користувача зберігаються особиста інформація користувача, така як його дата народження, його стать, сімейний стан, ім'я по батькові, громадянство, ідентифікаційний номер, адреса, місто та країна проживання, мобільний та стаціонарний телефон та електронна пошта

### Фінансова інформація користувача



Зберігається в окремій таблиці та з функцією 1 до 1 з сутністю користувача фінансова інформація користувача, наприклад, тип зайнятості (повна зайнятість, неповна зайнятість). Час, його зарплата, його чиста зарплата, різні види надбавок, різні види податків, назва банку, назва рахунку, номер рахунку та номер IBAN. Загальна зарплата є результатом додавання зарплати до загальної суми надбавки. Чиста зарплата виходить із зарплати мінус усі податки.

#### Особиста подія

Для особистих подій кожного користувача, які відображаються у вигляді календаря інтерфейс користувача зберігає назву, опис, дату та час дата і час початку і закінчення.

#### Вакансії

У цій сутності зберігаються всі робочі місця користувачів системи, які ми бачимо в асоціації User з сутністю Jobs кожного Користувач може мати більше однієї роботи, як він може історія обіймання більш ніж однієї посади в конкретній компанії. Ось так його збирається історія для всіх користувачів, навіть тих, хто вийшов із системи. Інформацією, що зберігається, є назва, дата початку та термін придатності.

#### Оплата

У цій сутності зберігається історія платежів компанії робочих місць, а отже й користувачам системи. Будь-коли платежі здійснюються. Адміністратор може подати деталі платежів і в система, таким чином зберігаючи електронну історію та маючи кращу картину їх платежів компанії за допомогою діаграм. Зберігається спосіб оплати, місяць оплати, її дата оплати (коли гроші були внесені), можливі штрафи, накладені загальна оплата та можливі коментарі.

#### Залишити заявку

Запити на ліцензії, зроблені користувачем, зберігаються в цій сутності. збереженою інформацією є дата початку дії ліцензії, дата, термін дії, вид

відпустки (Звичайна, Студентська, Хвороба, Шлюб), можливі додаткові коментарі і статус ліцензії (Очікує на розгляд, Схвалено, Відхилено).

#### Відділ

Відділи компанії / організації зберігаються в цій сутності. Назва розділу збережено.

#### Витрати

У цій сутності зберігаються витрати відділів компанії/організації. Зберігається така інформація, як назва продукту, назва магазину в якому було придбано, дата покупки, сума покупки

#### Оголошення відділу

У цій сутності зберігаються оголошення відділів, які можуть для перегляду користувачами цих розділів. Така інформація, як її назва, зберігається оголошення, опис та ідентифікатор користувача, який створив оголошення.

### 2.1.3 Функціональні пакети

Щоб краще зрозуміти деякі основні функції системи, що обговорюється в розділі 2.2.1 були створені пакети активності, що дозволяють виділити конкретну функцію системи більш детально.

#### *Додати співробітника*

Функція додавання співробітника доступна лише його адміністраторам.

Додати види діяльності співробітника На цьому кроці заповнюються форми для заповнення особистої інформації (наприклад, ім'я, телефон, номер рахунку, ім'я користувача, пароль. На цьому кроці з'являються форми для заповнення такої інформації, як особиста (наприклад, ім'я, телефон, номер рахунку, ім'я користувача, пароль тощо)

#### *Управління відомостями про заробітну плату*

Функцію управління відомостями про заробітну плату виконує його системний адміністратор. Під час цього процесу адміністратор зможе додати їх відомості про заробітну плату працівника. Ці статті включають заробітну плату, загальна (валова) зарплата, чиста зарплата, різні надбавки.

Спочатку система відображає всі відділи. Потім користувач вибирає користувач відділу, даними якого він хоче керувати. Також до якщо адміністратор не пам'ятає, в якому відділі знаходиться співробітник, є виберіть «Усі відділи», щоб уникнути фільтрації відділ. Далі з'являються форми для заповнення відомостей про зарплату. Після успішного завершення, адміністратор вибирає «Зберегти» та дані зберігаються в Базі даних. Нарешті, сторінка перенаправляє користувача до списку зарплат.

#### *Додати платіж*

Під час цієї функції адміністратор має можливість додати платіж до робочого місця, тобто шляхом розширення до користувача. Збереження платежів дозволяє системі зберігати історію витрат і служити інструментом для розробки майбутніх стратегій рішення компанії (наприклад, наймання нових працівників). Під час цієї операції адміністратор вибирає відділ, а потім один працівник, який його цікавить. Потім він вибирає місяць, для якого він хоче внести платіж. Одночасно відображається список історії платежів конкретного користувача до компанії. Наявність цього списку допомагає адміністратору мати зображення попередніх платежів, щоб уникнути можливої помилки платежу.

#### *Заява про відпустку*

Кожен користувач системи, незалежно від ролі, має можливість подати заявку відпустки. Користувач заповнює деталі програми, такі як тип програми, дата початку, дата закінчення та можливі коментарі. При відправці заявки заявка зберігається в Базі даних статус "очікує на розгляд". Щоб прийняти/відхилити заявку, доведеться втрутитися системний адміністратор або директор департаменту, до якого належить користувач. Важливо

вказати, що не слід давати права адміністратору або менеджеру приймати власні заявки. Отже, у списку заявок виконується перевірка, чи належить запит користувачу, який увійшов у систему, і якщо належить, то йому надається можливість редагувати.

#### **2.2.4 Діаграма розгортання**

Діаграми розгортання представляють його відображення програмне забезпечення на блоках-вузлах обробки. До них можна звикнути показати, які компоненти працюють на яких вузлах. Вузол – один фізичний об'єкт, який у загальному випадку має принаймні пам'ять і здатність обробки.

У випадку застосування, реалізованого в цій дипломній роботі, впроваджено систему, за винятком бази даних, яка налаштована на локальному сервері (localhost).

Однак діаграму розгортання було створено, щоб показати, як розроблене програмне забезпечення може відповідати деяким налаштуванням і масивам на сервері.

На наведеній схемі розгортання є 4 вузли, одним з яких є комп'ютер використовується користувачем системи (ПК користувача), одним є веб-сервер, який відповідає за обслуговування статичних файлів, а також за переспрямовувати на локальний сервер Application, тобто сервер Application відповідає за створення html-файлів із коду Javascript (через використання Node.js і React.js) і, нарешті, сервер MySQL, який володіє базою даних які використовує система

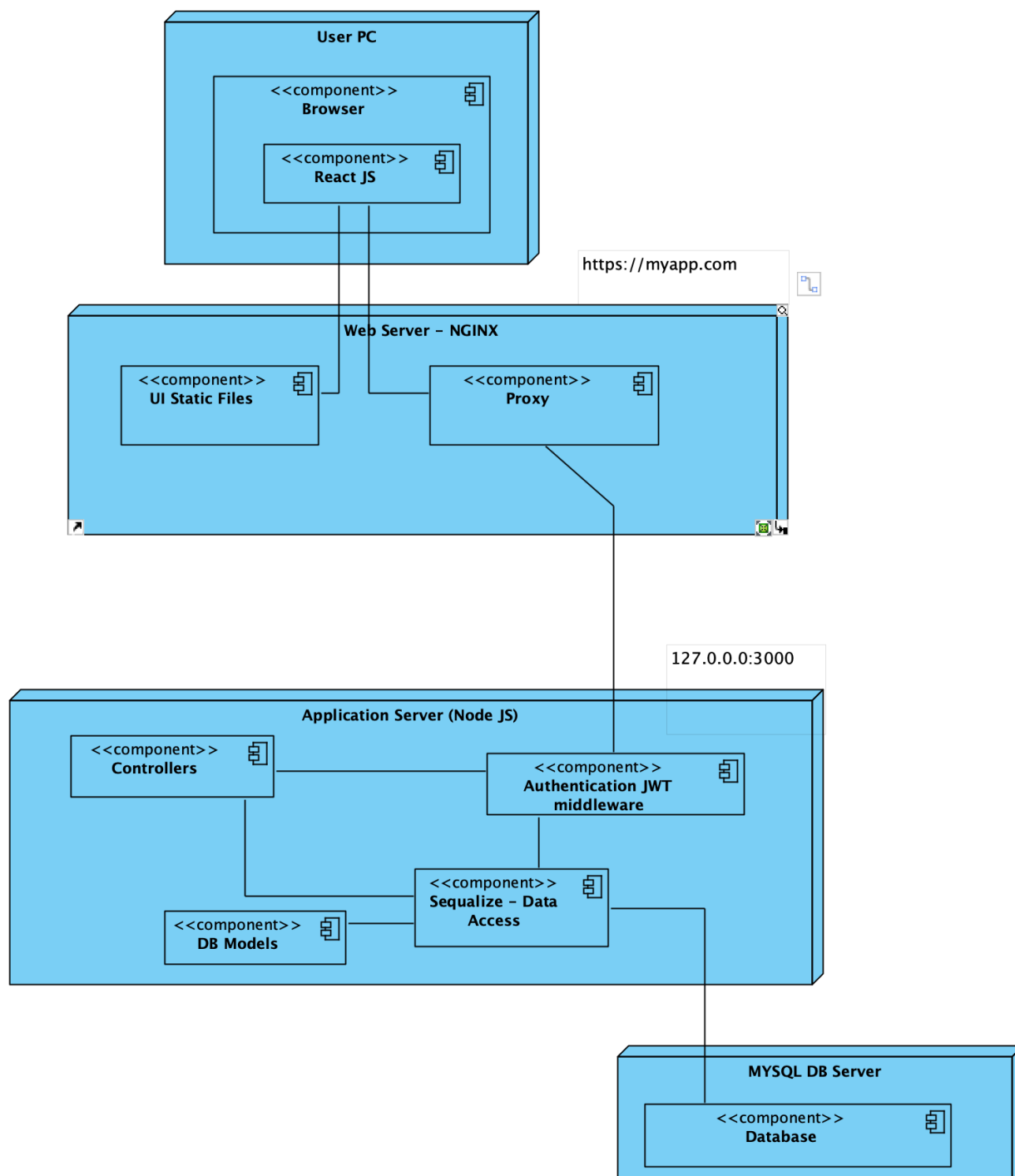


Рис. 2. Діаграма розгортання

У Application Server ми розрізняємо окремі вузли як Authentication Проміжне програмне забезпечення, яке відповідає за створення та перевірку JSON Web Токени, після перевірки імені користувача та пароля. Потім є спілкування з контролерами, які керують різними HTTP-викликами,

зробленими користувачами до системи. Контролери, які в цілому складають API системи, мають доступ лише до ORM (Object Relational Mapping) та його моделей. ORM тоді реалізація має прямий доступ до реляційної бази даних виконання запитів на створення, редагування та видалення записів.

Таким чином ми забезпечуємо безпеку системи, коли клієнт спілкується з сервером через API, який вимагає автентифікації.

### 2.2.5 Діаграма зв'язків сутностей

Модель зв'язку сутностей або ERM є абстрактною та концептуальною представлення даних, що використовується розробниками програмного забезпечення. ER-моделювання – це метод моделювання бази даних, який використовується для створення концептуальної схеми а основа відносин і її вимоги. Цей процес закінчується виробництвом діаграматичне представлення схеми бази даних називається Діаграма сутності-зв'язку (ERD).

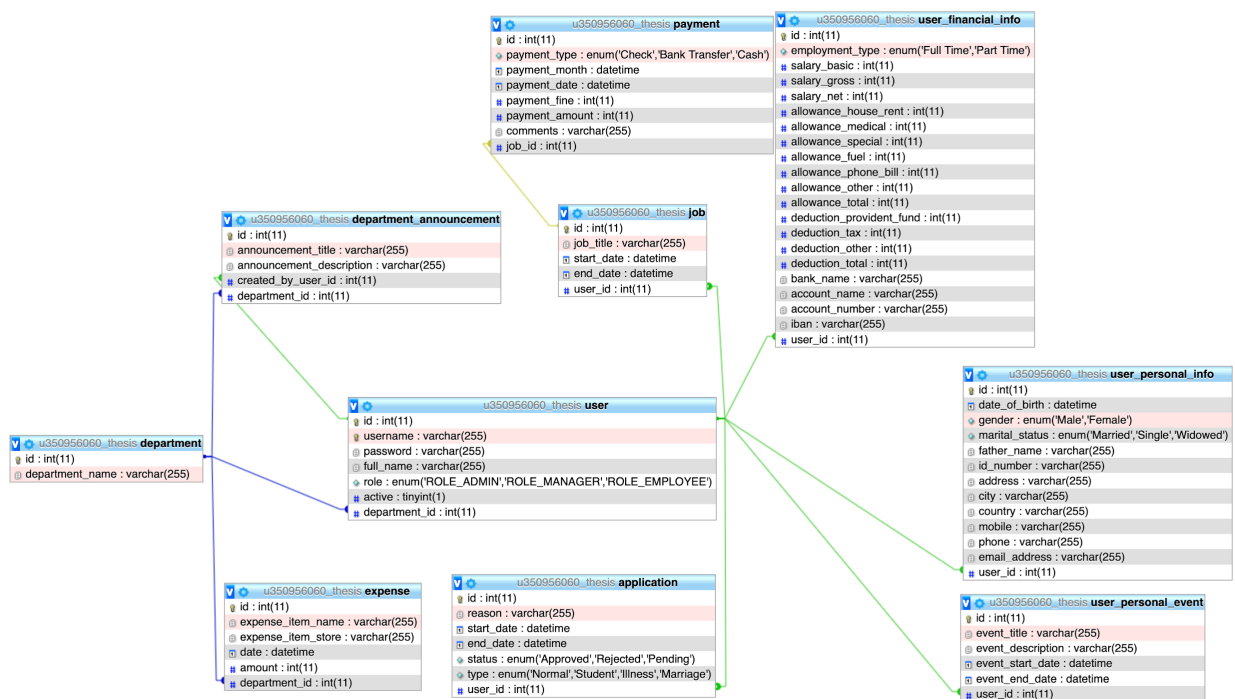


Рис. 3. Діаграма ERD

Від цього діаграма в поєднанні з діаграмою класів, згаданою вище, залежить від структури та створення бази даних і коду. Звідси і дизайн не повинний бути детальним, оскільки будь-яка помилка могла спричинити помилку в роботі і, звичайно, тривалу затримку в реалізації. У реляційній базі даних дані зберігаються у формі таблиць. Деякі дані в таблицях вказують на дані в інших таблицях, тому таблиці пов'язані. Логіка подання цієї структури відображається за допомогою діаграми «сутність-зв'язок».

За допомогою діаграми ER ми можемо краще зрозуміти розподіл і кореляцію між таблицями бази даних програми.

## 2.2 API

Зв'язок клієнта з сервером і отримання даних з базою даних через API.

Більшість вбудованих API захищені та вимагають аутентифікація користувачів для виклику. Деякі посилання, як-от посилання для входу та реєстрації, не вимагають їх аутентифікація.

Крім того, до API було додано додаткові заходи безпеки, щоб уникнути від викрадення Authentication та/або атак на розкриття даних. За допомогою проміжного ПЗ functions можна використовувати лише певні кінцеві точки API ролі користувачів. Тож навіть якщо автентифікований користувач намагається використовувати інші кінцеві точки API, він не матиме до них доступу.

Тому ці функції проміжного програмного забезпечення використовувалися належним чином у всіх Кінцеві точки API, які використовуються клієнтом, щоб уникнути витоку конфіденційних даних інформації.

## 2.3 Функціональні та нефункціональні вимоги

У цьому підрозділі розглянемо функціональні та нефункціональні системні вимоги. Функціональною вимогою, яку слід проаналізувати, є її

управління інформацією про можливе виведення/видалення користувачів із системи.

У різні моменти впровадження програми, під час видалення користувачів, користувачеві відображається відповідне повідомлення щодо оновлення та фіналу його згоди на видалення інформації.

Під час остаточного видалення користувача його дані також видаляються, наприклад, його особиста та фінансова інформація, можливі оголошення, його особисті дані, його заяви та його трудова історія.

На відміну від них, не видаляється жодна інформація, яка б вплинула на загальне зображення витрат і платежів компанії/організації.

При видаленні розділу спочатку проводиться перевірка на наявність користувачів, які знаходяться в системі. Якщо у відділі є користувачі, то адміністратор повинен переводити користувачів до іншого розділу, перш ніж отримати дозвіл на видалення відділу.

Що стосується нефункціональних вимог, то система, яку необхідно впровадити, повинна ефективно виконувати обробку запитів та відповідей. Для забезпечення захисту використовуються методи автентифікації (наприклад, токени API). Нарешті потрібно розробити простий у використанні зрозумілий інтерфейс користувача.



## РОЗДІЛ 3 Реалізація програми

Для реалізації програми використовувався набір певних мов програмування, технології, бібліотеки та фреймворки.

Для системного сервера використовували фреймворк і бібліотеку Node.js Express.js. Для системного клієнта використовувалася бібліотека React.js. Використана база даних MySQL.

### 3.1 Node.Js

Node.js — це платформа для розробки програмного забезпечення (переважно на стороні сервера). середовище JavaScript. Його мета — надати простий спосіб створення масштабованих веб-додатків. На відміну від більшості сучасних середовищ розробки мережових програм, в яких процес роботи вузла не покладається багатопотоковість (multithreading), а в асинхронній моделі зв'язку введення-виведення [12]. Цей вид операційної моделі спрямований на вдосконалення потужності обробки веб-додатків з багатьма функціями введення/виведення, а також веб-додатки в реальному часі (програми спілкування в реальному часі, браузерні ігри) [13].

Архітектура платформи передбачає програмування, кероване подіями (програмування, кероване подіями) на серверах, що дозволяє швидко розробляти сервери в Javascript [14]. Програмування, кероване подіями, є однією з моделей, у якій потік програми визначається такими подіями, як дії користувача (клацання мишею, натискання кнопки), виходи датчиків або повідомлення від інших програм/потоків. Ця домінуюча модель програмування використовується в графічних інтерфейсах користувача та програмах, орієнтованих на виконання певних дій у відповідь на введення користувача. В програм, керованих подіями, зазвичай існує базовий цикл, який чекає на подію, а потім запускає функцію зворотного виклику, коли відбувається подія [15].

За допомогою описаної вище моделі розробник може виконувати створення великомасштабних серверів без використання багатопоточності, але використовуючи спрощену модель. Спрощена модель програмування, керованого подіями, використовує зворотні виклики для сигналу про завершення процесу [14]. Вузол було створено тому, що паралелізм важко реалізувати в багатьох мовах і часто призводить до зниження продуктивності. Розвиток Node базується на механізмі Javascript V8 з відкритим вихідним кодом Google, що має чудову швидкість і вільне володіння основними Інтернет-протоколами HTTP, DNS, TCP [16]. Нарешті, основа платформи, мова Javascript, є такою широко поширеною, що робить її доступною для всієї спільноти веб-розробників.

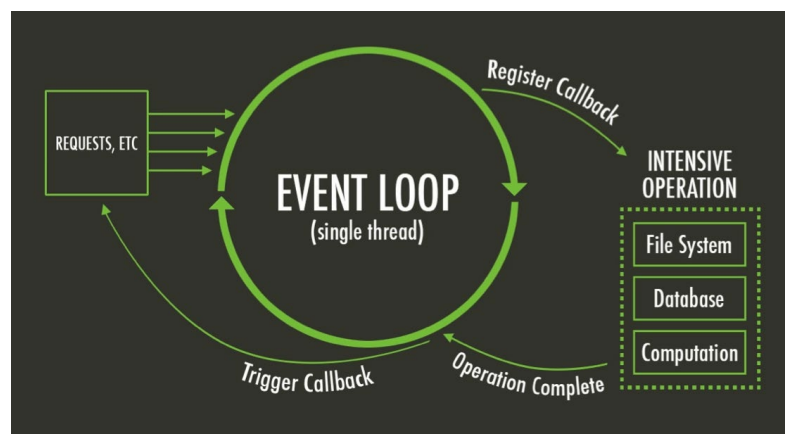


Рисунок 4. Архітектура NodeJS [14]

Деякі переваги NodeJS:

- Підвищена швидкість. Як згадувалося вище, Node — це платформа розробки програмного забезпечення з використанням рушія V8, який був побудований Google для інтеграції у свій браузер Chrome. Спеціальний рушій відмінно компілює та виконує код JavaScript з високою швидкістю в основному завдяки тому, що його компілятор перетворює Javascript безпосередньо в машинний код.

- Пропонує цикл подій. Цикл подій — це один потік, який виконується асинхронно, використовуючи паралельні потоки. Такий підхід має тенденцію бути подоланим завдяки збільшеній затребуваній пам'яті. І навпаки, коли програма Node на вимогу виконання операції введення-виведення відправляє асинхронне завдання у циклі подій разом із функцією зворотного виклику та продовжує його нормальний хід програми.

- Надає інструментарій npm. Ця бібліотека явно не є унікальною для Node і нагадує набори інструментів з інших систем, але відмінно справляється з визначенням і встановленням залежностей. Також зберігаються пакети встановлення залежностей ізольовано між завданнями, що дозволяє уникнути конфліктів версій.

- Він базувався на вже популярній мові Javascript. Найбільш широко підтримуються фреймворки розробки додатків на стороні клієнта загалом, якщо не виключно, у логіці Javascript. Платформі Node потрібно перекласти логіку клієнтської сторони на свою власну сторону сервера. Переклад теж не потрібен для даних HTTP, надісланих до різних об'єктів на стороні сервера.

### 3.2 React.Js

React був представлений як проект з відкритим кодом у травні 2013 року. Це бібліотека Javascript для створення інтерфейсів користувача. Відповідно орієнтована на перегляд програмна основа (від модель MVC). Для належної роботи не потрібна жодна залежність. Як новіша програмна основа, вона швидша за інші бібліотеки, легше освоюється. Ще одна потужна функція React полягає в тому, що кожен його компонент являє собою елемент інтерфейсу користувача (інтерфейс користувача) - елемент форми/таблиці, заголовок сторінки тощо. React створює власний DOM (віртуальний), де працюють компоненти. Це дає розробникам велику гнучкість і дивовижний приріст продуктивності, оскільки заздалегідь обчислює зміни, які необхідно внести в DOM. Так дозволяє уникнути дорогих операцій DOM і ефективно оновлювати.

React використовує синтаксис JSX, суміш Javascript та HTML. JSX спрощує весь процес написання компонентів для веб-сторінок.

Одна з найбільших проблем із фреймворками/бібліотеками Javascript полягає у певній незручності для пошукових систем (Search Engine Optimization/SEO). Хоча нещодавно в цій області відбулися деякі покращення, це теж не допомогло. React виділяється тим, що він також може працювати на сервері, і віртуальний DOM буде повернено та відтворено програмою перегляду як звичайна сторінка, цього разу оптимізована для пошукової системи

Optimization.

Перевагою React є те, що він пропонує можливість повторного використання компонентів у будь-який час. Це дуже важливо для економії часу. Компоненти в React ізольовані, і зміна одного не впливає на інші. Це дозволяє розробникам повторно використовувати компоненти, які не виробляють змін, і внести їх програмування більш точно, ергономічне та зручне для них [18].

### 3.3 Express.js

Express.js або просто Express — це структура, яка використовується в середовищі Node.js для створення Restful API [19].

### 3.4 MySQL

Мова структурованих запитів (SQL) використовується для керування даними бази даних. Менеджмент включає як створення та модифікацію таблиць програми, так і введення та пошук даних на основі конкретних критеріїв відбору. Стандартна мова SQL включає в себе наступні блоки:

Мова визначення даних (DDL): Ця мова містить команди, які дозволяють нам реалізовувати таблиці, зв'язки між таблицями і взагалі всієї структури бази даних.

Мова обробки даних (DML): Ця мова дозволяє керувати даними програми, наприклад імпортувати, видаляти, пошук і модифікація даних.

Визначення видів бази (View Definition): дозволяє створювати види баз даних, які визначені як віртуальні таблиці (віртуальні таблиці), що містять дані з однієї або кількох таблиць у базі даних.

Визначення повноважень (Authorization): дозволяє створювати групи користувачів і призначення різних прав доступу кожному з них, щоб кожна група користувачів могла керувати лише своїми даними.

Керування цілісністю (Integrity): дозволяє детально контролювати дані, які вносяться до бази даних, щоб не порушувати правила цілісності (обмеження цілісності), які наперед визначені та які, якщо їх дотримуються, видаляють ризик запису непослідовних даних (непослідовних даних).

Мова керування даними (DCL): обробляє авторизацію даних.

Мова обробки даних (Data Manipulation Language DML) дозволяє керування даними базових таблиць, а точніше імпортом, видалення та зміну записів таблиці. Крім того, є можливість отримати з таблиць дані, які відповідають деяким критеріям. За допомогою команд INSERT, DELETE, UPDATE і SELECT здійснюється реалізація цих процедур.

### 3.5 Sequelize

Sequelize — це ORM (Object-Relational Mapping) зіставлення, яке підтримує PostgreSQL, MySQL (використовується в цьому випадку), SQLite і MSSQL [20].

### 3.6 Bootstrap

Bootstrap — це набір інструментів з відкритим кодом (безкоштовне програмне забезпечення) для створення веб-сайтів та онлайн-додатків. Містить HTML і CSS для типографіки, кнопки навігації та інші елементи

середовища, а також додаткові розширення JavaScript. Це найпопулярніша програма в GitHub і використовувався NASA та MSNBC та інші.

Bootstrap був розроблений Марком Otto та Джейкобом Торнтоном у Twitter як одна структура для забезпечення узгодженості внутрішніх інструментів [21].

### 3.7 React Material Table

Під час впровадження системи виникла необхідність використання таблиць для перегляду списку інформації (перегляд списку співробітників, вакансій тощо). Оскільки система може вмістити великий обсяг інформації співробітників, то використовуються функції пагінації, фільтрації та класифікації. Тобто користувач повинен мати можливість контролювати ліміт записів на сторінку таблиці, щоб відсортувати таблицю за будь-яким із її стовпців побажань.

Після широкого вивчення всіх існуючих бібліотек було вирішено використовувати Бібліотеку React Material Table, яка ідеально поєднує існування всіх згадані функції [22].

### 3.8 Веб-токени JSON

Веб-токени JSON є стандартом, починаючи з платформи, яку вони несуть передається через JSON.

JWT містять у собі всю необхідну інформацію. Це означає, що JWT здатний передати основну інформацію про нього, його корисне навантаження (що зазвичай інформація, що стосується користувача) і підпис. JWT можна легко перенести. Оскільки JWT є автономними, вони знаходять використання в заголовках HTTP під час автентифікації API. Вони також можуть надсилаються через URL-адресу.

Визначити JWT легко. Це 3 буквено-цифрові символи, розділені крапкою. Наприклад: aaaaaaaaaa.bbbbbbbbbbbb.cccccccccccccc.

Розглянемо важливість кожного поля [23].

Оскільки у нас 3 поля, кожне з них створюється по-різному. Ці поля це:

- Заголовок
- Корисне навантаження
- Підпис

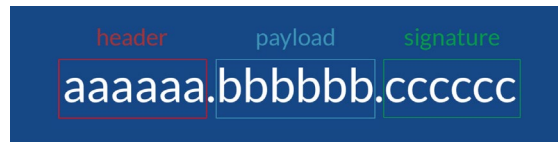


Рис. 5. Веб-токен JSON

Заголовок складається з 2 частин:

- посилання типу, яким є JWT
- використовуваний алгоритм хешування

Нижче наведено приклад:

```
{
  "тип": "JWT",
  «alg»: «HS256»
}
```

Оскільки ця частина закодована (base64encode), ми маємо першу частину JWT, яка це виглядає приблизно так: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 .

Корисне навантаження несе основне корисне навантаження JWT. Ось де надходить інформація ми хочемо передати, а також іншу інформацію, пов'язану з токеном.

Приклад корисного навантаження показано нижче:

```
{
  "iss": "scotch.io",
  «exp»: 1300819380,
  «name»: «Chris Sevilleja», «admin»: true
}
```

Який закодований у

eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOiJzMDA4MTkzODAsIm5hbWUiOiJD  
aHJpcyBTZXZpb

GxlamEiLCJhZG1pbiI6dHJ1ZX0

Це друга частина JWT.

Третя і остання частина - це підпис. Ця частина складається з хеш-значення заголовков, корисного навантаження та секретного ключа. Цей ключ знаходиться на сервері. Отже сервер може перевіряти справжність існуючих токенів і генерувати нові. Ось так і виходить і остання частина токена:

03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773 .

### 3.9 Додаткові модулі

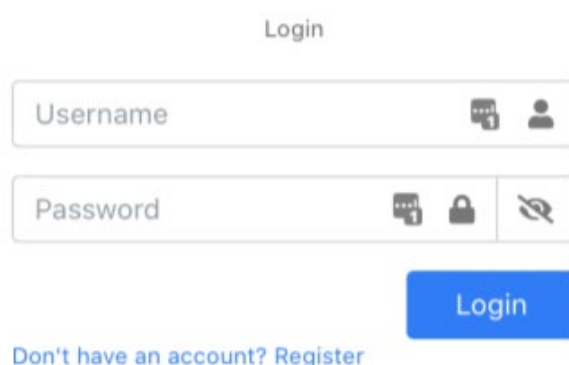
- Chart.js — для створення діаграм на веб-сайті
- Момент - використовується для форматування дат.
- Axios – клієнт на основі Promise для запитів HTTP
- ReactDatePicker - для системних календарів



## РОЗДІЛ 4: Сценарії використання

### 4.1 Загальні положення

Коли користувач переходить до застосунку, він стикається зі своїм екраном входу у систему.

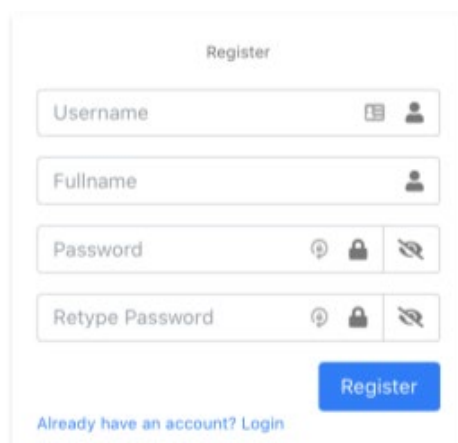


The image shows a login screen titled "Login". It features two input fields: "Username" and "Password". The "Username" field has a speech bubble icon with the number "1" and a user icon. The "Password" field has a speech bubble icon with the number "1", a lock icon, and an eye icon. Below the "Password" field is a blue "Login" button. At the bottom, there is a link that says "Don't have an account? Register".

Рис. 6. Екран входу

На цій сторінці користувач системи має можливість ввести дані. Якщо у користувача немає облікового запису, у нього є два варіанти:

- З опцією «Не маєте облікового запису? Реєстрація» може зареєструватися в його дані та зачекайте, поки адміністратор активує його його обліковий запис, таким чином надаючи йому доступ до системи. Обліковий запис, який створений цим процесом є простим Employee.



The image shows a registration screen titled "Register". It features four input fields: "Username", "Fullname", "Password", and "Retype Password". The "Username" field has a speech bubble icon with the number "1" and a user icon. The "Fullname" field has a user icon. The "Password" and "Retype Password" fields have a speech bubble icon with the number "1", a lock icon, and an eye icon. Below the "Retype Password" field is a blue "Register" button. At the bottom, there is a link that says "Already have an account? Login".

Рис. 7. Екран реєстрації

- Може повідомити системного адміністратора (через живе спілкування за номером телефону або електронною поштою), щоб створити обліковий запис через додаток. Цей спосіб рекомендовано для створення облікових записів менеджера або адміністратора.

Потім, коли співробітник входить в систему, є 3 сценарії.

- При введенні невірної інформації з'являється таке повідомлення

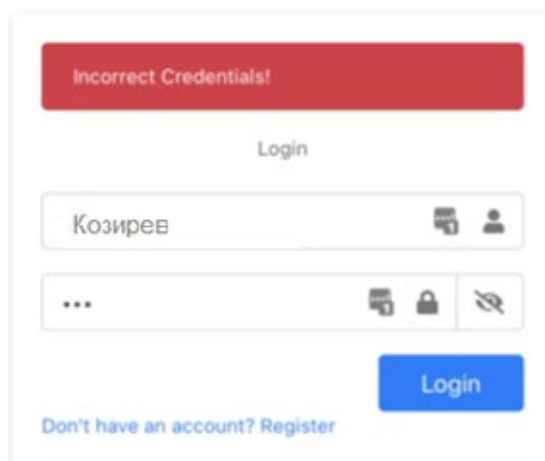


Рис. 8. Неправильний вхід

- Під час входу в обліковий запис, який не було активовано, відображається наступне повідомлення

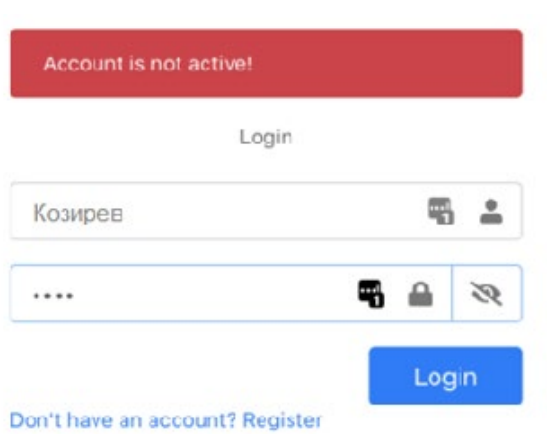


Рис. 9. Неактивний обліковий запис

- В іншому випадку користувач увійшов до системи

Коли адміністратор заходить в систему, він бачить сторінку «Інформаційна панель». На цій сторінці відбувається збір інформації з метою її швидкого оновлення користувача для стану системи.

У лівій частині завжди є бічна панель (вертикальна панель навігації), яка допомагає користувачеві орієнтуватися в програмі. Аналіз сценаріїв проводитиметься в 3 категоріях через три різні екрани які були створені, а також окремі системні ролі.

## 4.2 Адміністратор (Admin)

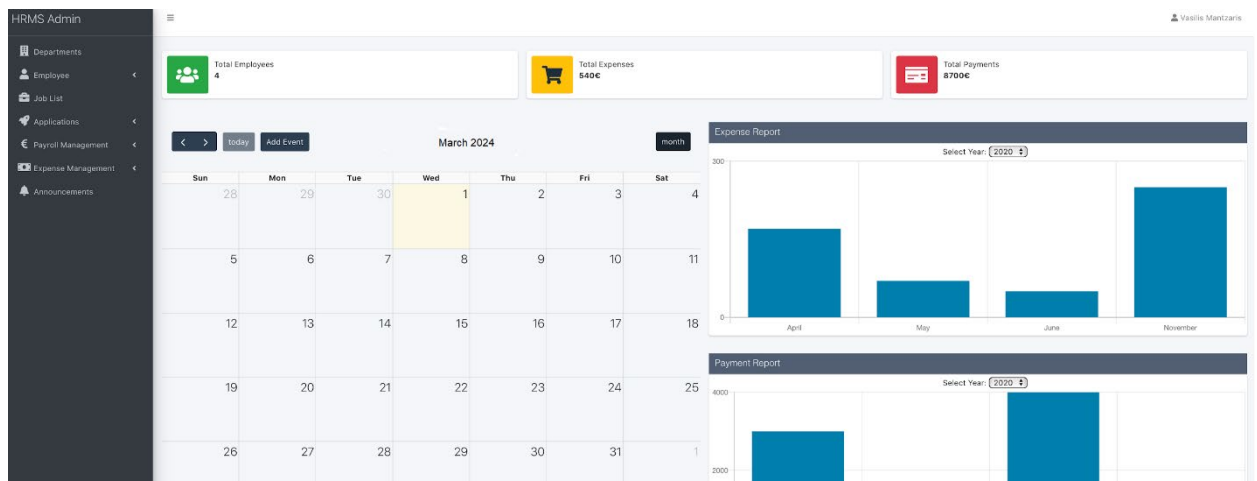
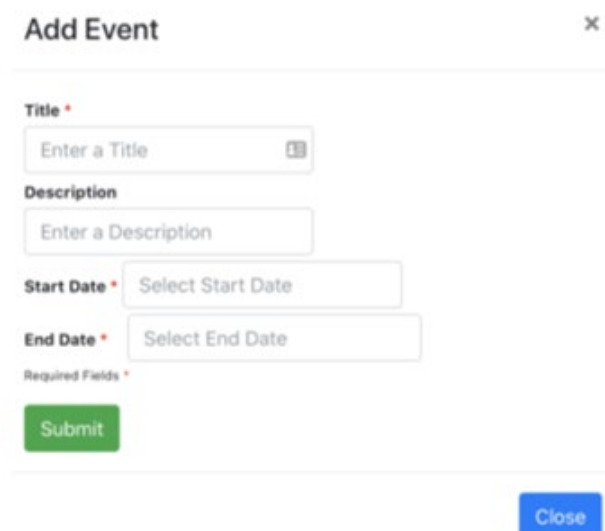


Рис. 10. Інформаційна панель адміністратора

У верхній частині є 3 інформаційні поля, які відображають таку інформацію, як загальна кількість користувачів системи, загальні витрати компанії та загальні платежі.

Потім є календар, за допомогою якого користувач може додавати, редагувати та видаляти свої особисті дані. Вибравши дату або опцію «Додати подію», розташовану вгорі Календаря, користувач може додати особисту подію.



**Add Event** x

**Title \***  
Enter a Title

**Description**  
Enter a Description

**Start Date \*** Select Start Date

**End Date \*** Select End Date

Required Fields \*

Submit

Close

Рис. 11. Додавання події

Необхідно заповнити назву, дату та час початку та закінчення. Дата кінцева дата має бути після дати початку. Інакше відображається відповідне повідомлення.

Праворуч є дві діаграми типу стовпчастої діаграми, які ілюструють витрати і платежі компанії за місяць. Користувач також має можливість, змінити рік, щоб швидко переглянути дані за попередні роки порівняння.

У нижній лівій частині є місце для останніх програм. Вони з'являється лише тип заявки, який користувач її подав, та її статус. Додатки які з'являються в цій області, це заявки протягом 2 тижнів.

Унизу праворуч є місце для останніх оголошень. Отже користувач може бачити на головному екрані 2 останні оголошення компанії.

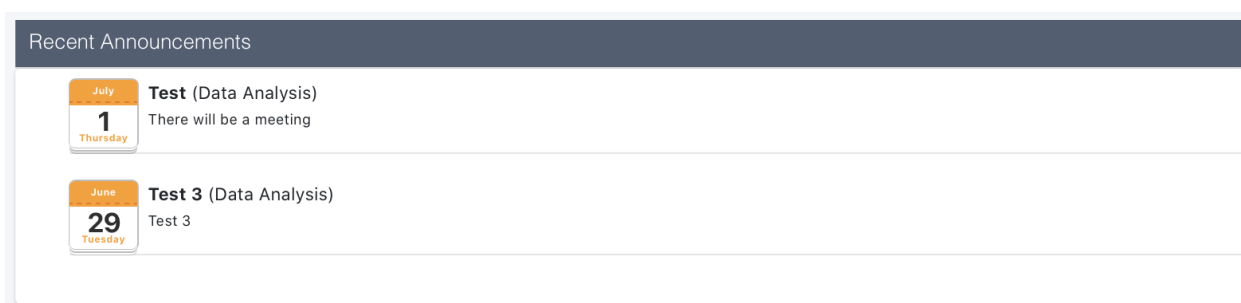


Рис. 12. Останні оголошення

На цій сторінці адміністратор може додати відділ, переглянути його список набір відділів компанії, а також для редагування та видалення

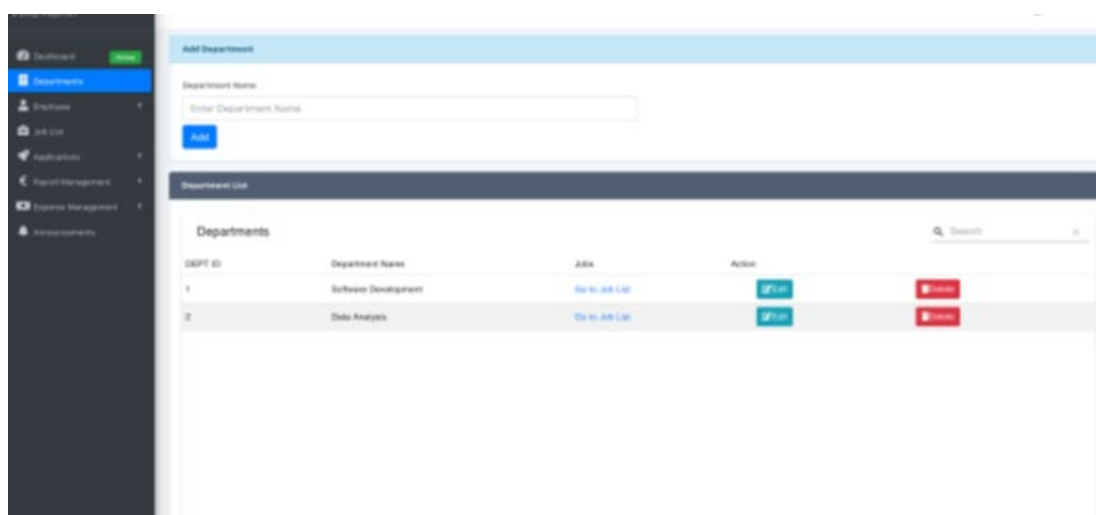


Рис. 13. Відділи

Також є посилання «Перейти до списку вакансій», яке перенаправляє користувача перелік посад кожного відділу.

### Співробітники

Управління працівниками було розділено на 2 сторінки.

На першій сторінці йдеться про додавання нового співробітника в систему. На цій сторінці адміністратор може ввести всі дані про співробітника, а також посаду, яку він займатиме в компанії. На цьому кроці не додаються

фінансова інформація користувача, наприклад його зарплата. Це робиться самостійно, на наступному кроці.

Друга сторінка присвячена перегляду списку всіх співробітників. На цій сторінці адміністратор може мати огляд усіх співробітників, статус своїх облікових записів (активний/неактивний), обробка та видалити їх. Також адміністратор може вибрати піктограму з вкладкою, для буде перенаправлено на вкладку співробітника, де відображається набір його даних.

### Вакансії

На цій сторінці адміністратор бачить список вакансій компанії. Спочатку потрібно вибрати розділ, який вас цікавить, або пропустити його категоризація за відділом, вибравши «Усі відділи». Він також має опцію для додавання роботи через індикатор «Додати роботу». Робота повинна бути доданою до існуючого працівника.

### Заявки на отримання ліцензії

Управління програмами розділено на 2 сторінки.

На першій сторінці користувач може подати заявку на отримання ліцензії, заповнивши її його початок і кінець, тип і можливі коментарі

На другій сторінці адміністратор може побачити список своїх запитів співробітників компанії та затверджувати/відхиляти їх. Важливо, щоб ще раз зазначимо, що менеджерам і директорам не надається можливості затверджувати власні заявки.

### Управління відомостями про заробітну плату

Управління відомостями про зарплату розділено на 3 сторінки. На першій сторінці адміністратор, обираючи відділ і співробітника, може керувати деталями фінансових даних таких як зарплата, податки та пільги.

На другій сторінці відображається список з фінансовими даними всіх посад роботи, отже, також і працівників компанії. У адміністратора є можливість редагування та перегляду фінансової звітності працівника, що більш детально показує всі його фінансові дані.

На третій сторінці адміністратор може ввести виплату зарплати у систему. Точніше, він вибирає відділ і співробітника, який його цікавить, а також місяць оплати. Потім відображається історія користувача. Єдина зміна, яку можна зробити це додавання можливого штрафу та коментаря до оплати.

### Відомчі витрати

Управління витратами відділу ділиться на 3 сторінки. На першій сторінці адміністратор або менеджер додає витрати по відділах (столи, стільці...).

Через другу сторінку вони мають доступ до списку видатків відділів, відфільтрованого за місяцями.

### Оголошення відділу

На цій сторінці адміністратор і менеджер можуть додавати оголошення відділу. Адміністратор має можливість розмістити оголошення по всіх відділах компанії. Також їх може бачити адміністратор оголошення всіх відділів, а директор тільки свого відділу.

## 4.3 Менеджер

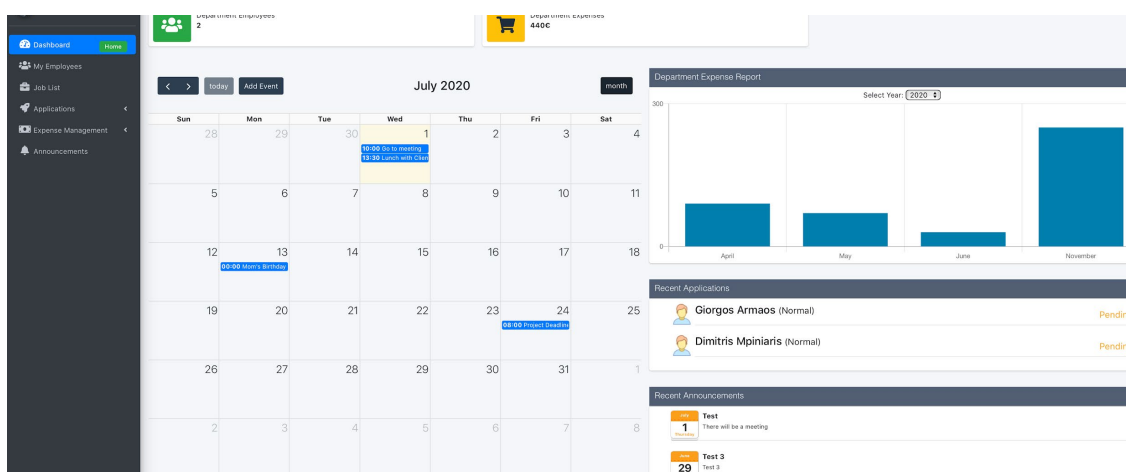


Рис. 14. Менеджер інформаційної панелі

На відміну від адміністратора, екран менеджера не містить інформації про загальні платежі системи. Крім платіжної системи, менеджер може збирати інформацію про свій відділ, наприклад, систему калькуляції, оголошення, співробітники відділу, їх робочі місця та їх застосування. Також дивіться останні оголошення відділу

#### **4.4 Співробітник**

На екрані звичайного користувача відображається його вкладка для перевірки інформації. Він також має право подавати особисті події та демонструвати їх. Він може переглянути вкладку фінансів даних, оголошення відділу, а також подати заяву.



## **ВИСНОВКИ**

В роботі проведено аналіз та проектування електронної системи управління персоналом. Для розробки обрана клієнт-серверна архітектура. Для реалізації використано мову Javascript як на сервері, використовуючи Node.js, Express, js, так і на стороні клієнта, з використанням React.js. Також використовуються бібліотеки для реалізації системних функцій.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://www.sdbor.edu/cffile/Agendas/HRFIS/HOW%20AN%20HRIS%20CAN%20I%20MACT%20HR.do>
2. <https://cdn.ymaws.com/alachicago.org/resource/resmgr/newsletter/novdec2012.pdf>
3. [https://www.researchgate.net/publication/37149629\\_Human\\_Resource\\_Management\\_Information\\_Technology\\_and\\_the\\_Competitive\\_Edge](https://www.researchgate.net/publication/37149629_Human_Resource_Management_Information_Technology_and_the_Competitive_Edge)
4. Kovach, K. & Cathcart, C. Human Resource Information Systems (HRIS): Providing Business with Rapid Data Access, Information Exchange and Strategic Advantage. *Public Personnel Management*, 1999. Vol.28, No.2
5. Walker, A.J. How The Web And Other Key Trends Are Changing Human Resources, In Walker, A. (Ed.), *Web-Based Human Resources*, 2001
6. <https://search.proquest.com/openview/65a6d83e1080fdc2f1ca947be8ead040/1?pq-origsite=gscholar&cbl=40682>
7. <https://www.employment-studies.co.uk/system/files/resources/files/398.pdf>
8. [https://www.academia.edu/7206134/E-HR\\_adoption\\_and\\_the\\_role\\_of\\_HRM\\_evidence\\_from\\_Greece](https://www.academia.edu/7206134/E-HR_adoption_and_the_role_of_HRM_evidence_from_Greece)
9. Schnitt, D.L. Reengineering the organization using information technology. *J. Systems Management*, 1993 January, pp.14-20, 41-42.
10. Ngai, E., & Wat, F. Human resource information systems: A review and empirical analysis. *Personnel Review*, 2006. Vol.35 No.3, pp. 306
11. Booch G., Rumbaugh J. and Jacobson I. *The Unified Modeling Language User Guide*, Addison Wesley, Boston, MA, 1999
12. <https://ieeexplore.ieee.org/document/5617064>
13. <https://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/>
14. <https://books.google.gr/books?id=ZH6bpbcrlvYC>

15. [https://www.academia.edu/32647425/A\\_Practical\\_Introduction\\_to\\_Hardware\\_Software\\_Codesign](https://www.academia.edu/32647425/A_Practical_Introduction_to_Hardware_Software_Codesign)
16. <https://ptgmedia.pearsoncmg.com/images/9780672335952/samplepages/0672335956.pdf>
17. Node.js <https://nodejs.org/en/about/>
18. React.js <https://reactjs.org>
19. Express.js <https://expressjs.com>
20. Sequelize <https://sequelize.org>
21. Bootstrap <https://getbootstrap.com>
22. Material Table <https://material-ui.com/components/tables/>
23. Jwt <https://jwt.io/introduction/>