Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

на тему

“ Дослідження та програмна реалізація застосування NoSQL баз даних в інформаційній системі”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп'ютерні науки»
спеціальності 123 «Комп'ютерна інженерія»
_____ Пауков О.С.
«____» _____ 2023р.

Керівник проекту
кандидат технічних наук, доцент
_____ Босько В.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *магістр*
Галузь знань *12 "Комп'ютерні науки"*
Спеціальність *123 "Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
"____" _____20___року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Паукову Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація застосування NoSQL баз даних в інформаційній системі*

2. Керівник роботи *Босько Віктор Васильович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 35-13 від 04.08.22

3. Строк подання роботи до захисту *08.01.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне дослідження та програмна реалізація застосування NoSQL баз даних в ІС*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	09.11.2023 р.	17.11.2023 р.
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2023 р.	21.11.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	12.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	18.10.2023 р.	
3.	Розробка моделі компонента	23.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	32.10.2023 р.	
6.	Програмування алгоритмів	11.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	16.11.2023 р.	
9.	Оформлення ПЗ	18.11.2023 р.	
10.	Попередній захист роботи	04.12.2023 р	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Пауков О.С. Дослідження та програмна реалізація застосування NoSQL баз даних в інформаційній системі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації застосувань NoSQL баз даних в розробці ПЗ.

Метою розробки є дослідження та програмна реалізація додатку з використанням NoSQL баз даних .

Об'єктом дослідження є процес реалізації ПЗ з застосуванням NoSQL баз даних.

Предметом дослідження є методи реалізації розробки ПЗ з застосуванням NoSQL БД.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація проекту засобами фреймворку NodeJS та MongoDB.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися у браузері Chrome, Firefox, Safari. Програму розроблено в середовищі NodeJS з використанням БД MongoDB.

Ключові слова: комп'ютерна інженерія, веб-сайт, NodeJS, бази даних, ООП, NoSQL, MongoDB.

ANNOTATION

Paukov O.S. Research and software implementation of the application of NoSQL databases in the information system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this master's thesis, software was developed, which is intended for the implementation of NoSQL database applications in software development.

The purpose of the development is the research and software implementation of the application using NoSQL databases.

The object of the study is the process of software implementation using NoSQL databases.

The subject of the study is the implementation methods of software development using NoSQL databases.

Research methods are based on coding theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the project using the NodeJS and MongoDB frameworks.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used in Chrome, Firefox, and Safari browsers. The program was developed in the NodeJS environment using the MongoDB database.

Keywords: computer engineering, website, NodeJS, databases, OOP, NoSQL, MongoDB.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ ТА СПЕЦІАЛЬНИХ ТЕРМІНІВ

ІС	–	інформаційна система.
ПЗ	–	Програмне забезпечення.
Хеш	–	Результат обробки даних хеш-функцією.
CSS3		Cascading Style Sheets 3 – каскадні таблиці стилів третього покоління.
HTML	–	HyperText Markup Language – мова розмітки гіпертексту.
	–	Hypertext Preprocessor – скриптова мова загального призначення, інтенсивно застосовується для розробки веб додатків.
БД		База даних.
JSON		JavaScript Object Notation – текстовий формат обміну даними.
SQL		Structured query language – мова структурованих запитів.
ПК	–	Персональний комп’ютер
NoSQL		not only SQL

ВСТУП

Актуальність теми. У сучасному інформаційному суспільстві, в якому обсяги даних стрімко зростають, важливою задачею стає ефективне управління та обробка інформацією. Традиційні реляційні бази даних, які довго служили основою для зберігання та організації даних, виявляються обмеженими в обробці великих обсягів інформації та в умовах високих навантажень.

Застосування NoSQL (Not Only SQL) баз даних виходить за межі традиційних підходів, надаючи нові можливості для зберігання, обробки та взаємодії з даними. Ці бази даних відрізняються гнучкістю схем, високою швидкодією роботою з великими обсягами даної і можливістю масштабування горизонтально для відповіді на зростаючі потреби систем.

Магістерська робота спрямована на дослідження та аналіз застосування NoSQL баз даних в інформаційних системах. Зокрема, вивчаються переваги та обмеження використання NoSQL технологій, їхній вплив на швидкодію та масштабованість систем, а також можливості використання в конкретних відомостях. Робота також присвячена вивченню практичних аспектів впровадження NoSQL рішень у реальних проектах та їхнього порівняння з традиційними реляційними базами даних**.**

Висновки мого дослідження стануть цінним ресурсом для розробників, архітекторів систем та фахівців, що займаються управлінням даними в умовах постійної зміни технологічного ландшафту.

Ця робота сприятиме розширенню знань у сфері баз даних, а також визначенню найбільш ефективних стратегій використання NoSQL підходів для розробок в сучасних інформаційних системах.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи аналізу застосування NoSQL баз даних в сучасних ІС, оцінці можливостей, переваг та обмежень цих технологій, а також виявленні

їхнього впливу на якість та швидкість обробки даних в різноманітних сценаріях використання.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

Аналіз сучасного стану баз даних:

– Провести аналіз сучасного стану баз даних, виявивши тенденції їхнього розвитку та основні виклики, з якими стикаються інформаційні системи в умовах постійних змін.

Дослідження технічних аспектів NoSQL баз даних:

– Ретельно вивчити технічні аспекти різних типів NoSQL баз даних, включаючи гнучкість схем, швидкодію, масштабованість та можливості взаємодії з іншими технологіями.

Оцінка переваг та обмежень NoSQL технологій:

– Здійснити критичний аналіз переваг та обмежень використання NoSQL баз даних у порівнянні з традиційними реляційними системами.

Дослідження впливу на швидкодію та масштабованість систем:

– Вивчити вплив застосування NoSQL баз даних на швидкодію та масштабованість інформаційних систем, розглядаючи різні сценарії навантаження та обсягів даних.

Аналіз практичних аспектів впровадження:

– Розглянути практичні випадки впровадження NoSQL технологій в реальних проектах, вивчивши виклики та переваги, що виникають при їхньому застосуванні.

Порівняння з традиційними реляційними базами даних:

– Провести об'єктивне порівняння NoSQL баз даних із традиційними реляційними системами з урахуванням різних критеріїв, таких як продуктивність, гнучкість та вартість підтримки.

Формулювання рекомендацій для практичного використання:

– На основі отриманих результатів сформулювати рекомендації для розробників та архітекторів інформаційних систем з урахуванням специфіки їхніх завдань та потреб.

Об'єктом дослідження є застосування NoSQL для реалізації – інформаційної системи управління HR-процесами.

Предметом дослідження є самі NoSQL бази даних та їхнє застосування в інформаційних системах. Методи та програмні засоби реалізації інформаційної системи управління HR-процесами.

Методи дослідження базуються на методах зберігання даних, методах математичної статистики, методах теорії масового обслуговування, методах розробки програмного забезпечення, теорії алгоритмів та об'єктно-орієнтованого проектування.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– удосконалено метод вибору NoSQL БД для роботи з сучасними ІС;

– розроблено інформаційну систему на основі NOSQL БД, яка має більш широкі можливості та швидкість доступу до даних на відміну від існуючих аналогів.

Практична цінність отриманих результатів дослідження включає практичні висновки з впровадження NoSQL технологій в реальних проектах. Це дозволяє розробникам і архітекторам систем отримати конкретні вказівки та рекомендації для ефективного використання NoSQL в певних умовах. Для практичного використання отриманих теоретичних результатів розроблено інформаційну систему автоматизації HR-процесів з використанням нереляційної бази даних MongoDB, яка базується на використанні сучасних інформаційних технологій

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями

параметрів на функціонуючій ІС, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи аналізу застосування NoSQL БД в сучасних ІС, є актуальною задачею, яка потребує вирішення у даній магістерській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

У сучасному цифровому світі, де величезні об'єми даних генеруються в реальному часі та зростають з несподіваною швидкістю, проблема зберігання, управління та аналізу інформації стала найбільш актуальною та викликаючою для багатьох сфер життя. Цей нестримний розквіт кількісних даних супроводжується неабиякими викликами і новими можливостями, визначення та вирішення яких є важливим завданням для сучасних інформаційних систем.

Інформаційні системи (ІС) в сучасному світі виконують різноманітні та ключові функції, визначаючи ефективність та конкурентоспроможність підприємств та організацій.

Призначення ІС охоплює широкий спектр завдань, спрямованих на обробку, зберігання та передачу інформації з метою підтримки прийняття рішень та забезпечення ефективного функціонування організаційних процесів.

Автоматизація бізнес-процесів

Однією з ключових функцій ІС є автоматизація бізнес-процесів. Вони оптимізують внутрішні та зовнішні операції підприємства, зменшуючи час виконання завдань та покращуючи точність операцій. Інформаційні системи дозволяють автоматизувати фінансовий облік, управління запасами, обробку замовлень та інші важливі аспекти діяльності.

Підтримка прийняття рішень

Інформаційні системи надають ключову інформацію для прийняття стратегічних та тактичних рішень. Вони забезпечують аналіз даних, розробку звітів та прогнозування, сприяючи ефективному управлінню ресурсами та формуванню стратегій розвитку.

Забезпечення доступу до інформації

Інформаційні системи забезпечують структурований та швидкий доступ до інформації для різних користувачів в організації. Це включає в себе створення

зручних інтерфейсів, можливість обміну даними та забезпечення конфіденційності інформації.

Впровадження електронного бізнесу

Інформаційні системи відіграють ключову роль у впровадженні електронного бізнесу, сприяючи взаємодії з клієнтами, партнерами та постачальниками через Інтернет. Вони дозволяють створювати та обслуговувати електронні майданчики, автоматизувати транзакції та реагувати на зміни на ринку.

Забезпечення безпеки інформації

Інформаційні системи включають засоби захисту конфіденційності, цілісності та доступності інформації. Вони використовуються для виявлення та запобігання кіберзагрозам, забезпечуючи надійний захист даних підприємства.

Управління взаємодією з клієнтами

Інформаційні системи допомагають підтримувати взаємодію з клієнтами, забезпечуючи CRM-системи для зберігання та аналізу даних про клієнтів, автоматизуючи процеси обслуговування та підтримки клієнтів.

1.1. Призначення системи

База даних є головною і складної частиною інформаційних систем, які призначені для обробки і збереження інформації. Сучасний світ характеризується експотенційним ростом обсягів даних, які генеруються в різних галузях, включаючи медицину, фінанси, наукові дослідження, соціальні мережі і т. д. Окрім того, дані стають все різноманітнішими і можуть включати текст, графи, геодані, зображення, відео та інші формати.

Метою роботи є розробка інформаційної системи автоматизації HR-процесів з використанням сучасних технології та нереляційних систем управління даними із врахуванням показників узгодження в базах даних NoSQL на етапі проектування інформаційних систем.

Використання традиційних реляційних баз даних для таких різноманітних даних може бути обмеженим, і NoSQL бази даних надають нові можливості для ефективної роботи з ними.

1.2. Область застосування

NoSQL бази даних знайшли широке застосування в різних галузях, включаючи фінанси, медицину, телекомунікації, аналіз даних, Інтернет речей (IoT), веб-розробку та інші. Дослідження та реалізація застосування NoSQL баз даних в інформаційних системах може допомогти розробникам та архітекторам систем зрозуміти, як ці технології можна використовувати для різних завдань. Нижче наведено головні напрямки застосування таких типів БД.

Веб-застосунки та соціальні мережі

Великі веб-сайти та соціальні мережі, де швидкість доступу та масштабованість грають ключову роль, використовують NoSQL бази даних. Гнучкість схем дозволяє швидко змінювати та розширювати дані, а масштабованість — ефективно обробляти великі обсяги інформації.

Аналітика та обробка великих даних

NoSQL бази даних використовуються для зберігання та обробки великих обсягів даних в аналітичних системах. Їхні можливості масштабування дозволяють швидко аналізувати та отримувати інсайти з великих датасетів.

Інтернет речей (IoT)

NoSQL бази даних ефективно використовуються для зберігання та обробки даних, зібраних з різноманітних сенсорів та пристроїв IoT. Гнучкість схем дозволяє зберігати різноманітні типи даних, а масштабованість — впоратися з великим обсягом одночасних записів.

Електронна комерція

В індустрії електронної комерції, де потрібно ефективно обробляти та зберігати велику кількість інформації про товари, клієнтів та транзакції, NoSQL

бази даних можуть забезпечити високий рівень продуктивності та масштабованості.

Геопросторові дані

В сферах, пов'язаних із геолокацією та геопросторовими даними, такими як геонавігація, картографія, агротехнології, NoSQL бази даних можуть ефективно зберігати та опрацьовувати великі обсяги геоданих.

Логістика та управління ланцюгом постачання

В області логістики та управління ланцюгом постачання, де необхідно відстежувати та керувати потоками товарів та інформацією, NoSQL бази даних дозволяють швидко виконувати операції з великою кількістю даних.

Наукові дослідження

У галузі наукових досліджень, де часто виникає потреба в обробці складних та великих датасетів, NoSQL бази даних можуть використовуватися для зберігання та швидкого доступу до експериментальних даних

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи аналізу застосування різних типів баз даних в сучасних ІС, є актуальною задачею, яка потребує вирішення у даній магістерській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

Сучасний стан автоматизації кадрових процесів характеризується динамічним розвитком у сучасному світі, багато професій виходять на новий рівень.

Системи автоматизації та цифровізації HR-процесів набирають обертів, стають все більш популярними та пропонують все більш широкий функціонал.

Діджиталізація поступово перетворюється з прагнення до оптимізації процесів у структуру необхідності сучасного HR. Активно розвиваються комплексні експерти, які допомагають компаніям. Це дуже важливий тренд у світі управління людськими ресурсами. Завдяки їй поступово з'являється розуміння необхідності демонтажу HR та автоматизації частини процесу, щоб звільнити ресурси для інших важливих завдань. Ведення електронних таблиць і навчальні матеріали можна автоматизувати, але важливо автоматизувати відносини з фахівцем з кадрів для підтримки мотивації та професійного розвитку співробітників.

Потреба в автоматизації управлінських HR-задач виникла не так давно, тому можна сказати, що цей напрямок ще знаходиться в стадії розвитку. До вибору системи автоматизації управлінської діяльності слід підходити з особливою ретельністю, оскільки функції управління в кожній організації специфічні, а значить, необхідно орієнтуватися на максимальну відповідність алгоритму аналізу інформації, який існує в компанії, і вбудований у систему.

Основною проблемою, з якою стикаються HR-фахівці, є висока складність управління, величезна кількість завдань, функцій і процесів, якими необхідно керувати швидко і ефективно.

Спеціальні системи автоматизації управління персоналом формалізують і оптимізують всі процеси діяльності персоналу і дозволяють більш ефективно управляти людськими ресурсами. Автоматизована система управління кадровими процесами дозволяє не тільки систематизувати управління

персоналом, а й реалізувати його моніторинг і контроль, в якому важко буде знайти місце для помилок, порушень і зловживань.

Тому розробка інформаційної системи автоматизації кадрових процесів є актуальним напрямком наукових досліджень.

У процесі аналізу кадрової системи було визначено важливість серверної частини, яка виступає засобом зберігання інформації. В якості серверної частини використовується СУБД. Виходячи з сучасних тенденцій, системам NoSQL зараз приділяється особлива увага.

У таких системах(реляційних) дані зберігаються у вигляді таблиць, також передбачається наявність схеми бази даних. Але при створенні великих систем (Big Data) з використанням реляційних СУБД розробники почали відчувати значні труднощі: при наявності великої кількості вузлів виникає проблема забезпечення необхідної стійкості до системних помилок.

Як спроба вирішити накопичені проблеми реляційних баз даних, виникли альтернативні засоби зберігання та обробки даних, які отримали назву «бази даних NoSQL». Піонерами в цій галузі були дві компанії: Google і Amazon.

У базі даних NoSQL багаторазова реплікація (копіювання) записів використовується для забезпечення високої відмовостійкості. Але бази даних NoSQL мають недолік: ці системи не підтримують спосіб підтримки транзакцій і блокувань, тому виникає проблема відповідності реплікації.

Важливими показниками координації репліки в системах баз даних NoSQL є ймовірність читання застарілого запису протягом часу, коли оновлення розподіляються по вузлах системи, час очікування початку читання записів з оновлених серверів, кількість версій записів у NoSQL. бази даних і час їх обробки, ймовірність заборони доступу до запису БД тощо.

Ці особливості необхідно оцінювати на етапі проектування системи, оскільки це дозволяє уникнути ручного вибору необхідних значень параметрів для великої кількості типів записів бази даних на етапі налагодження та

необхідності повномасштабного моделювання. екстремальне навантаження на систему.

Оскільки технологія розробки інформаційних систем на основі баз даних NoSQL досить нова, математичні моделі, необхідні для оцінки показників відповідності реплік, або відсутні, або неадекватні.

2.1. Огляд існуючих систем

Існують різні системи автоматизації кадрових процесів. Розберемо найвідоміші рішення, відзначимо основні переваги і недоліки. Система BambooHR, головне вікно якої показано на рисунку 2.1.

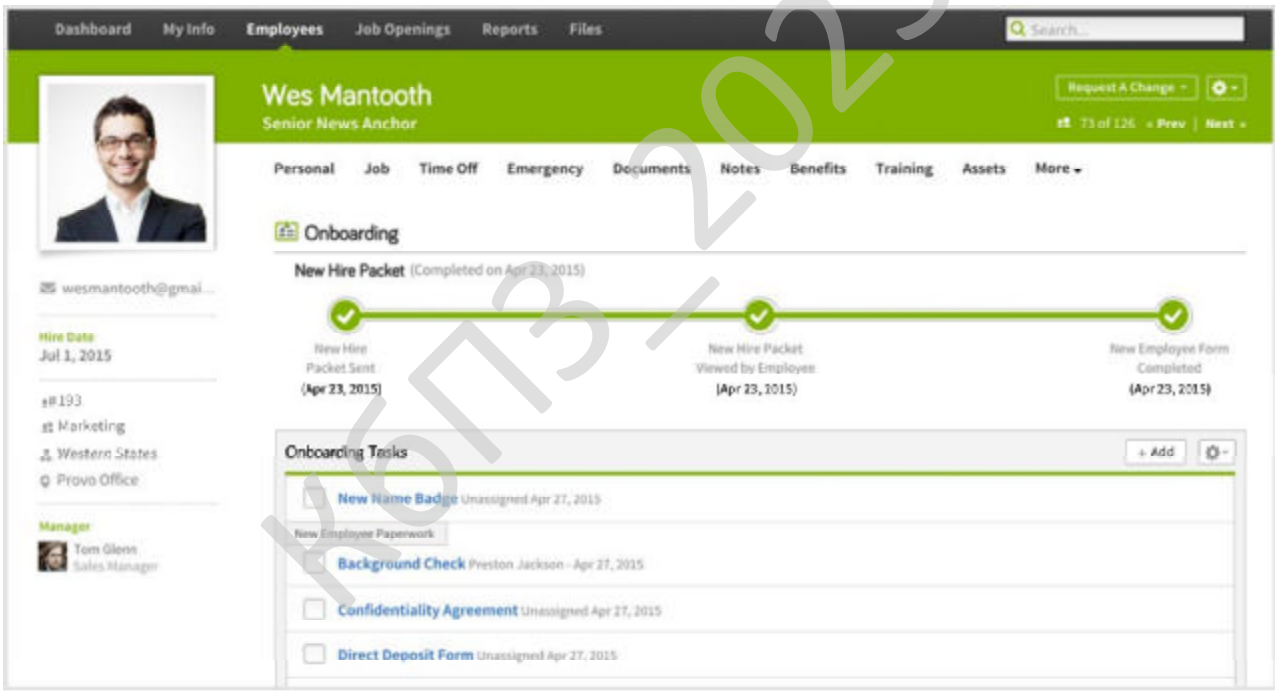


Рисунок 2.1 - Система BambooHR

Система для кадровиків, що пропонує комплексні рішення для управління персоналом. Найкраще підходить для зростаючого, малого та середнього бізнесу. Особливості BambooHR:

- система пошуку та моніторингу кандидатів;

- система відстеження винагороди;

- інтеграція з іншими інструментами (Zapier, Okta, OneLogin, Indeed, Greenhouse) і детальна аналітика;

- вбудований календар подій;

- моніторинг та управління навчанням працівників;

- детальна документація (FAQ);

- доступ через мобільні платформи.

Серед недоліків системи варто відзначити відсутність автоматизації наступних бізнес-процесів:

- присвоєний рейтинг користувачів серед співробітників;

- запис голів за команду;

- історія оцінок співробітників.

Система Hurma

Нещодавно на ринку з’явилося рішення, яке поєднує процеси підбору персоналу, HR та OKR (цілі та ключові результати). У системі є пакети для компаній різного розміру, які підходять як для зростаючих команд, так і для великих.

Особливості системи Hurma:

- моніторинг настрою працівників;

- організація структури підприємства у вигляді дерева;

- синхронізація з календарем Google, повідомлення про заходи компанії;

- особистий кабінет кожного працівника;

- продовжити розбір;

- автоматизація перевірки ефективності, привітання /коригування/ випробування/ вихідної співбесіди;

- можливість телефонного зв'язку з кадрами та керівництвом;

- робота з OKP;

- управління відсутністю;

- база вакансій компанії та база кандидатів у рекрутери;

- публікація вакансій на сайті компанії;

- інтеграція з LinkedIn;

- ведення статистики процесу працевлаштування та кадрового процесу.

Недоліки системи:

- у фільтрі неможливо відібрати кандидатів за певними курсами;

- сортування вакансій тільки за назвою та датою створення;

- не можна коментувати завдання щодо затвердження відпусток, дистанційної роботи та лікарняних.

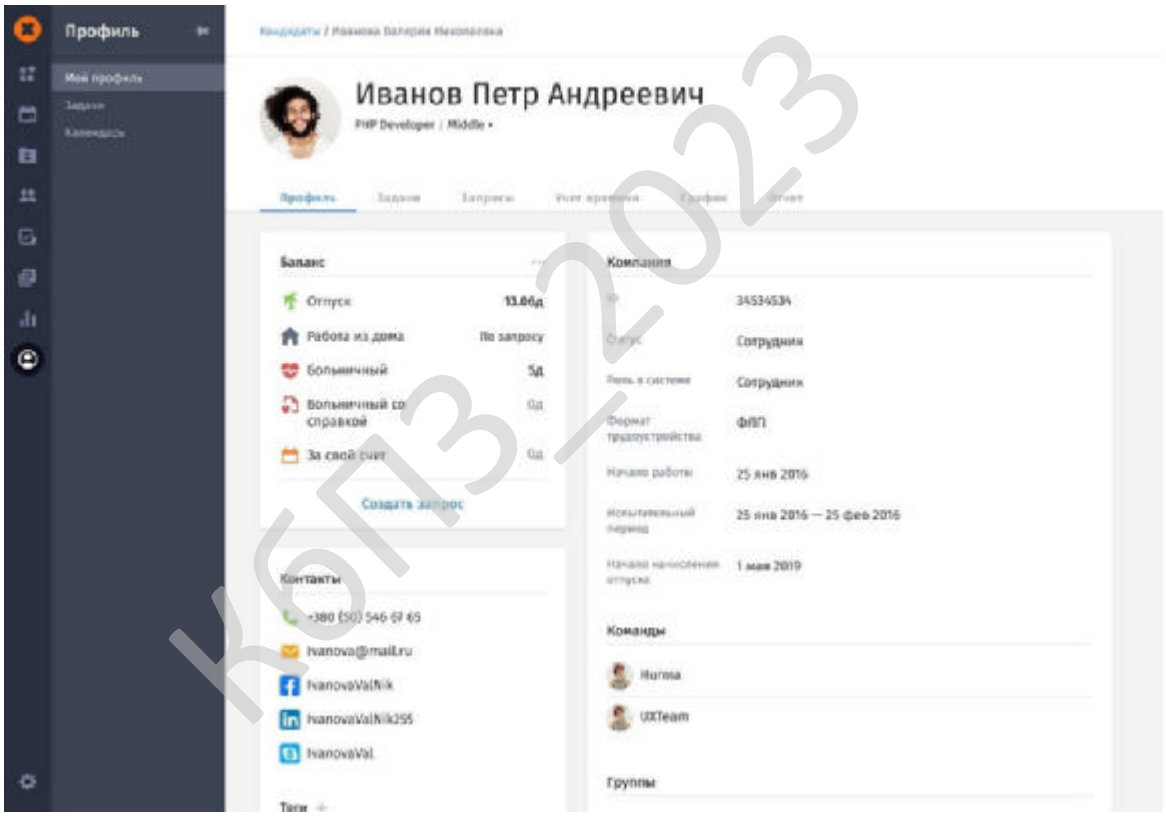


Рисунок 2.2 - Система Hurma System

Workable

Система управління процесом підбору персоналу, яка поєднує відстеження кандидатів (система відстеження кандидатів) і платформу підбору персоналу з потужною пошуковою системою на основі машинного навчання.

Функціональні характеристики:

- швидкий підсумковий аналіз;

- автоматизація пошуку співробітників (конструктор для створення анкет, резюме та вакансій);

- інтеграція з порталами працевлаштування;

- інтеграція з LinkedIn та можливість пошуку кандидатів через соціальні мережі;

- брендування інтерфейсу системи відповідно до фірмового стилю Вашої компанії;

- синхронізація календаря та розкладу завдань відділу кадрів;

- імпорт існуючих баз даних;

- персональний менеджер, служба підтримки по телефону та e-mail;

- Мобільний додаток;

- Розширення Google Chrome для швидкого пошуку кандидатів.

Система Zoho People

Програмне забезпечення для управління персоналом, розроблене для малого та середнього бізнесу. Він має простий інтерфейс і працює прямо з коробки.

Особливості Zoho People:

- наявність функціоналу для реалізації управління відсутністю;

- шаблони типових документів з особового складу;

- наявність мобільного додатку та веб-версії;

- моніторинг часу;

- впровадження аналітики для HR;

- функції перевірки ефективності;

- електронний підпис;

- внутрішня аналітика та звітність.

Робота кадровика надзвичайно важлива для компанії. Менеджер з персоналу виконує роль буфера між співробітниками та керівниками, розвиває

бренд компанії, покращує корпоративну культуру, мотивує та адаптує співробітників. Крім того, потрібно вести документацію, відстежувати всі лікарняні, відпустки, відрядження, проводити співбесіди, планувати заходи тощо. Всі ці завдання потрібно виконувати на високому рівні та безпомилково, адже від HR часто залежить робота інших спеціалістів.

Автоматизація дозволяє скоротити обсяг рутинних завдань, оптимізувати їх і перенести фокус на завдання вищого рівня, такі як підвищення лояльності команди, мотивації та продуктивності, адаптація нових співробітників, розвиток корпоративної культури тощо.

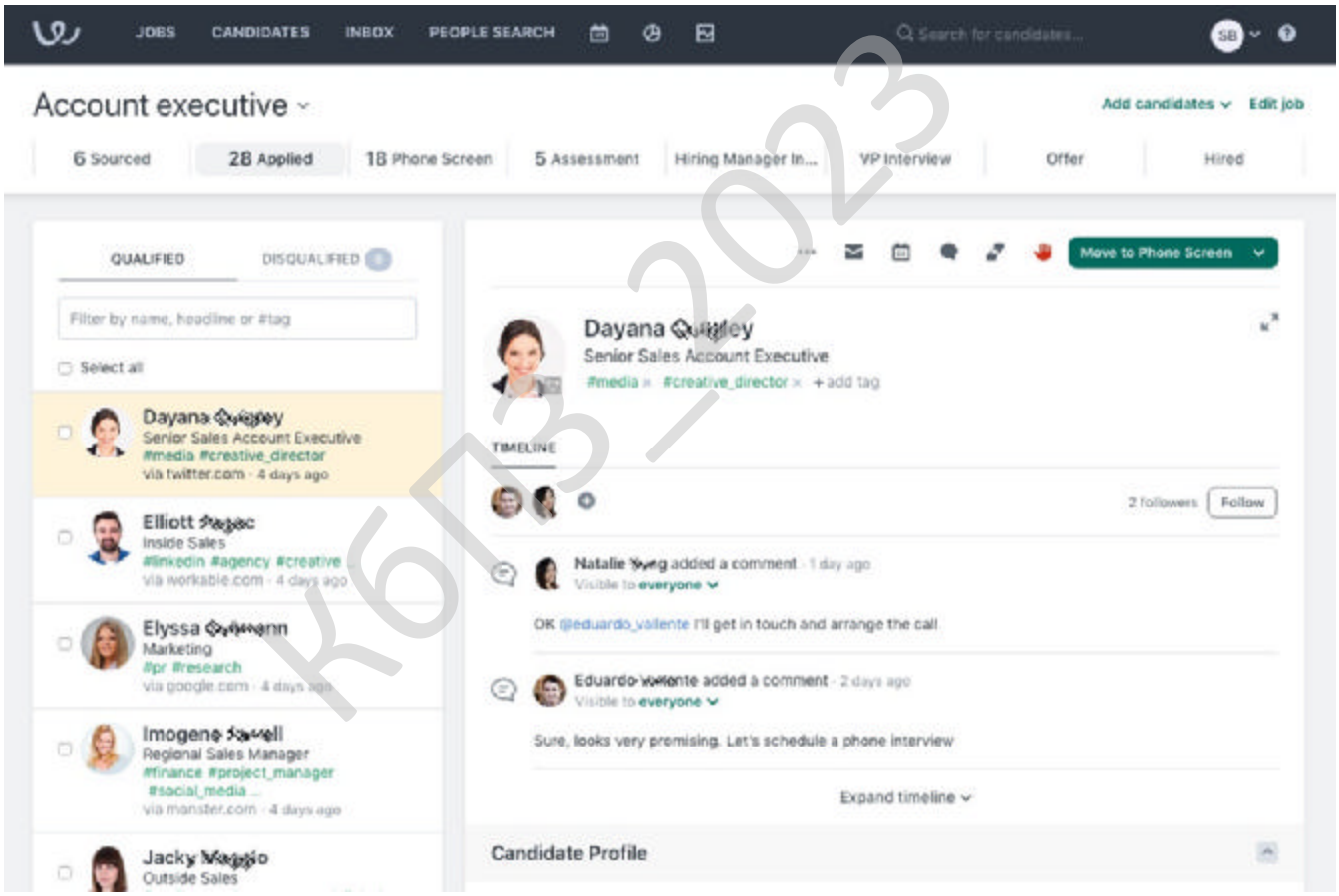


Рисунок 2.3 - Система Workable

2.2 Обґрунтування вибору методів розробки

У процесі аналізу HR системи було визначено важливість серверної частини, яка виступає засобом зберігання інформації.

В якості серверної частини використовується СУБД. Виходячи з сучасних тенденцій, системам NoSQL зараз приділяється особлива увага.

В останні кілька років у сфері обробки даних домінували реляційні СУБД. У таких системах дані зберігаються у вигляді таблиць, також передбачається наявність схеми бази даних.

Але при створенні великих систем (Big Data) з використанням реляційних СУБД розробники почали відчувати значні труднощі:

1) процедура агрегації даних ускладнилася, оскільки вимагає зчитування записів із великої кількості зв'язаних таблиць (виникла проблема втрати відповідності);

2) виникло протиріччя між необхідністю зберігати великі обсяги неструктурованих даних і необхідністю якось їх структурувати шляхом розробки схеми бази даних;

3) для зберігання великих обсягів інформації необхідне придбання дорогих спеціалізованих апаратно-програмних комплексів паралельних систем баз даних (Teradata, Sun Oracle Database Machine та ін.);

4) за наявності великої кількості вузлів виникає проблема забезпечення необхідної відмовостійкості системи.

Як спроба вирішити накопичені проблеми реляційних баз даних, виникли альтернативні засоби зберігання та обробки даних, які отримали назву «бази даних NoSQL». Піонерами в цій галузі були дві компанії: Google і Amazon. У базі даних NoSQL багаторазова реплікація (копіювання) записів використовується для забезпечення високої відмовостійкості. Але бази даних NoSKL мають недолік: ці системи не підтримують спосіб підтримки транзакцій і блокувань, тому виникає проблема відповідності реплікації.

Важливими показниками координації репліки в системах баз даних NoSKL є ймовірність читання застарілого запису протягом часу, коли оновлення розподіляються по вузлах системи, час очікування початку читання записів з оновлених серверів, кількість версій записів у NoSKL. бази даних і час їх обробки, ймовірність заборони доступу до запису БД тощо. Ці особливості необхідно оцінювати на етапі проектування системи, оскільки це дозволяє уникнути ручного вибору необхідних значень параметрів для великої кількості типів записів бази даних на етапі налагодження та необхідності повномасштабного моделювання. екстремальне навантаження на систему.

Оскільки технологія розробки інформаційних систем на основі баз даних NoSQL досить нова, математичні моделі, необхідні для оцінки показників відповідності реплік, або відсутні, або неадекватні.

Типи NoSQL баз даних

Незважаючи на різноманітність баз даних NoSQL, в процесі функціонування вони виконують деякі спільні функції, пов’язані з координацією реплік:

- розміщення реплік записів бази даних у кластері та забезпечення їх узгодження при оновленні записів;

- узгодження версій реплік (об'єднання кількох версій записів в один запис у процесі ведення треків версій);

- звірка (відновлення) реплік після усунення несправності у вузлі.

Бази даних NoSQL зазвичай використовують два методи налаштування та оновлення реплік: головний-підлеглий і кільцевий.

Перший метод передбачає зберігання даних на головному вузлі та їх реплікацію на підлеглі вузли. Всі зміни вносяться на головному вузлі і зберігаються в пам'яті цього вузла. Підлеглі вузли періодично опитують головний вузол, зчитують накопичені зміни та зберігають їх у своїй пам'яті.

Прикладами таких баз даних є MongoBD, HBase, Neo4j тощо. При розміщенні даних «на кільці» необхідно визначити, скільки секцій (v-вузлів) буде в ньому. Ці розділи впорядковані за серверами (на кільце).

На рисунку 2.4 показаний приклад. Тут визначено 64 розділи, які розміщені по колу на трьох фізичних серверах A, B, C. База даних виділить 21 або 22 розділи для кожного сервера (64/3).

Коли запис включається в базу даних, обчислюється хеш ключа, який визначає номер розділу (v-node), який визначає сервер, на якому буде зберігатися цей запис. Інші репліки записів (їх номер N-1) зберігаються на наступних (N-1) серверах, розташованих кільцем за годинниковою стрілкою від першого сервера. Прикладами розподілених кільцевих баз даних є Riak, Amazon Dynamo та інші. Використання віртуальних вузлів (v-nodes) має такі переваги [26]:

- якщо вузол стає недоступним (через поломку або планове обслуговування), навантаження рівномірно розподіляється між вузлами;

- коли вузол знову стає доступним або до системи додається новий вузол, цей вузол приймає приблизно однакове навантаження від кожного з інших доступних вузлів;

- кількість віртуальних вузлів, за які відповідає реальний вузол, можна визначити виходячи з потужності сервера, що дозволяє збалансувати навантаження в неоднорідній інфраструктурі.

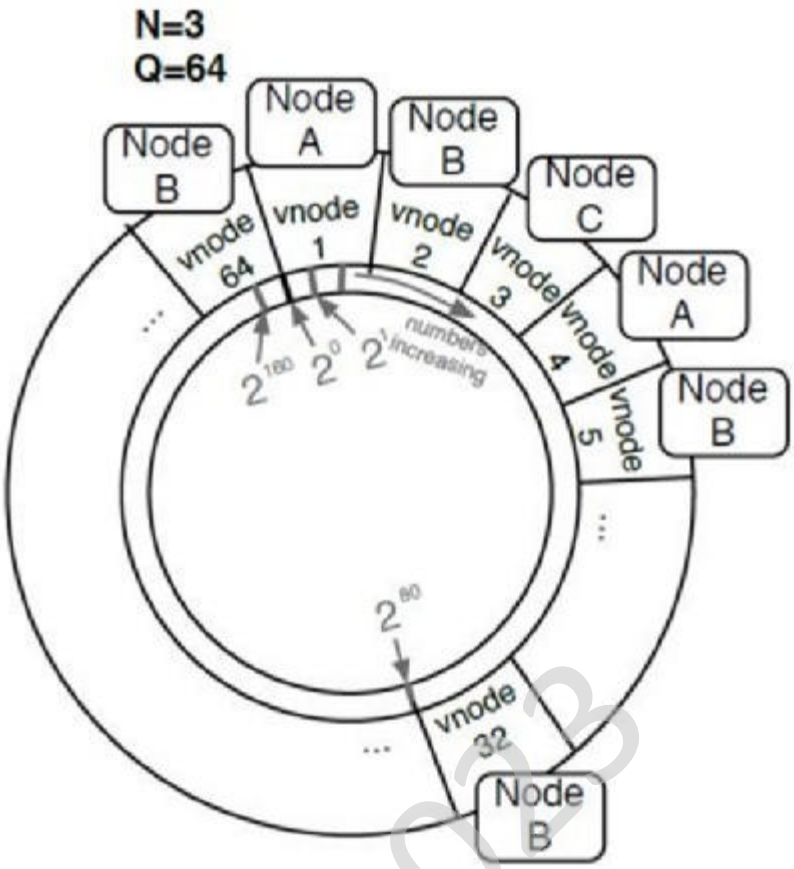


Рисунок 2.4 - Кільце з 64 v-вузлами і трьома серверами

Реплікація використовується для підвищення продуктивності та відмовостійкості. Коли запис оновлюється, інші репліки оновлюються не відразу, а під час вікна неузгодженості - часу, який мине з моменту оновлення запису на будь-якому вузлі до моменту оновлення копій запису на всіх інших вузлах. повний. Розрізняють такі види консистенції [17]:

- сувора послідовність - кожен запит на читання завжди повертатиме останнє оновлення запису;

- погана узгодженість - система не гарантує, що запит на читання завжди повертатиме останнє оновлення запису;

- можлива узгодженість - особлива форма слабкої узгодженості: система зберігання гарантує, що якщо немає нових оновлень запису, зрештою запит на читання поверне останнє оновлення.

Узгодженість тісно пов’язана з доступністю системи, а також із стійкістю до втрати з’єднання.

Цей зв'язок було сформульовано Еріком Брюером у теоремі CAP [18]. Теорема стверджує, що можна створити розподілену систему, яка є 1) узгодженою, 2) доступною та 3) толерантною до розділів, але лише дві з цих трьох властивостей можуть бути гарантовані одночасно. При узгодженості властивості 2 і 3 остаточно гарантовані.

Крім того, у статті обговорюється сувора узгодженість реплік і узгодженість реплік у кінцевому рахунку (CR-узгодженості). Давайте представимо такі теги:

N - кількість вузлів, на які в кінцевому підсумку буде тиражуватися запис (з певною затримкою);

V - кількість вузлів (реплік), в які повинні бути записані дані до того, як користувачеві (або додатку) буде надіслано відповідь про успішне завершення операції (якщо V < N, то система продовжує тиражувати дані на решту N-V вузли);

R — кількість вузлів, від яких база даних очікує відповіді на успішне завершення читання записів.

Відсутність блокувань дозволяє одночасно читати і змінювати один і той же запис бази даних на різних вузлах. Це призводить до конфліктів, які вирішуються звичайними інструментами NoSKL або вручну.

Перший, найпростіший спосіб розв’язати конфлікти – це використовувати мітку часу, надану кожному запису. У разі конфлікту перевага надається останньому запису. Але, як зазначено в [18], такий підхід важко реалізувати в кластері вузлів.

Іншим способом вирішення конфліктів є підтримка версій записів, реалізована за допомогою векторного годинника (Vector Clock - VC) [14, 26]. Вектор годинника — це масив пар <користувач, номер версії запису для цього користувача>, який описує порядок оновлення цього запису.

На рисунку 2.5 показано приклад ведення вектора годин (в дужках показано вектор реєстрації годин). Запис D (наприклад, якийсь документ) оновлюється користувачами A1, A2, A3. Перші два оновлення виконує користувач A1 послідовно. Потім користувачі A2, A3 одночасно читають і оновлюють цей запис (випадковий збіг). У базі даних зберігаються дві версії записів: D1 і D2. Під час читання документа D дві версії запису з однаковим ключем (D1 і D2) повертаються користувачеві A1. Змінити записи, наприклад, об'єднати оновлення, зроблені користувачами A2, A3. База даних зберігає одну узгоджену версію векторного запису години, включаючи ідентифікатори трьох користувачів.

Переваги годинникового вектора: відсутність єдиної точки відмови системи, оскільки при використанні часових позначок в записах необхідно виконувати точну синхронізацію часу з єдиним стандартом.

Недоліки тактового вектора: неможливість автоматичного вирішення конфліктів, а також збільшення довжини тактового вектора при багаторазовому оновленні запису. Однак у NoSKL є механізми для скорочення тактового вектора. Наприклад, в системі Ріак можна задати частоту підстроювання вектора на рівні сегмента, а також максимальний розмір (довжину) тактового вектора [19].

Наступний підхід до вирішення конфлікту полягає у створенні атемпорального тегу (може бути хешем вмісту, GUID, лічильником тощо), який повертається користувачеві разом із необхідними даними з СУБД. Після того, як користувач вносить зміни, система порівнює отримане від користувача значення тегу з тим, що зберігається в базі даних.

Якщо значення не збігаються, операція відхиляється. Користувач повинен прочитати та оновити запис ще раз. Такий підхід використовується в системі CouchDB [30].

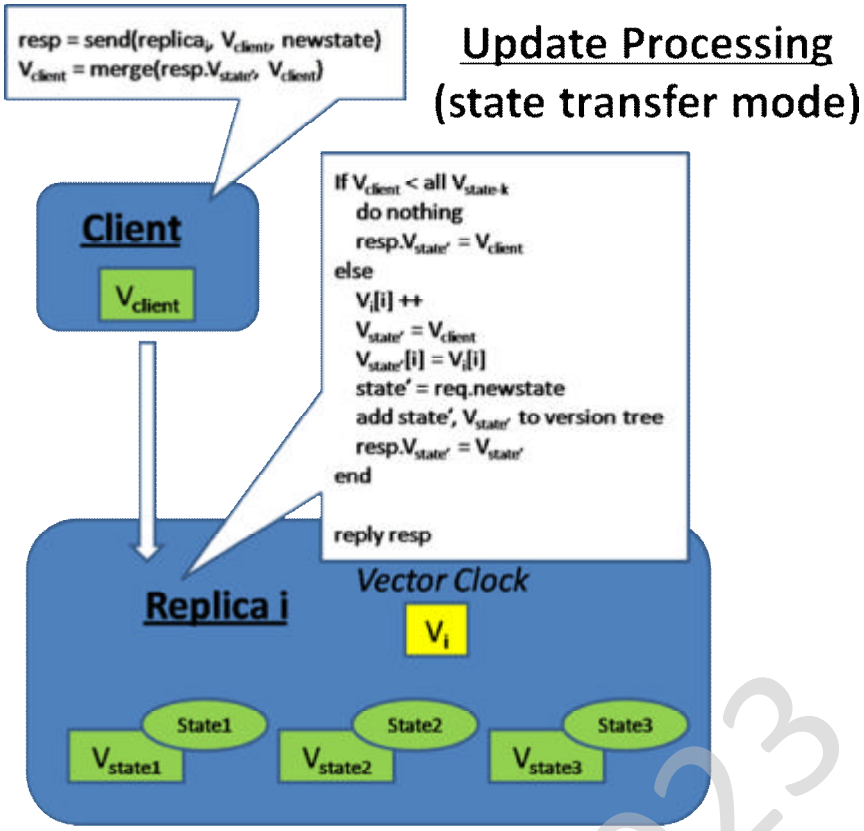


Рисунок 2.5 - Приклад ведення вектора годин

Наведемо приклади невідповідностей даних, які призводять до появи кількох версій записів, а також способи їх усунення:

- кілька співробітників одночасно виправляють копію одного документа (запису) - наприклад, пропозиції щодо розвитку компанії; усунення невідповідностей - уніфікація (вибір) коригувань керівником групи;

- кілька експертів одночасно оцінюють копію одного документа;

- наприклад, сліпа оцінка статті, представленої на конференції; усунення невідповідностей - вибір оцінки президента комісії;

- з одним примірником документа одночасно працюють кілька співробітників, і один співробітник може бачити результати роботи іншого співробітника - наприклад, кілька авторів працюють над однією статтею; може бути декілька варіантів запису, невідповідності усуває керівник авторської групи;

- користувачі обговорюють подію в блозі. Під час багатофазної обробки запитів з використанням технології MapReduce вихідні дані однієї фази є вхідними даними іншої фази - наприклад, коли виконується операція об'єднання під час доступу до структурованих даних або коли складні запити виконуються в браузерах (репліка вхідні дані наступної фази можуть бути суперечливими); цю невідповідність усуває NoSQL (узгодженість на основі потенційної причинності).

Додаток генерує серію пов'язаних записів, вони зберігаються в репліках не одночасно і в різній часовій послідовності - наприклад, результати індексації документа в пошукових системах; неузгодженість репліки усуває NoSQL (узгодженість на основі явної причинності).

З наведених вище прикладів зрозуміло, що забезпечення узгодженості дуже важливо для стабільного та належного функціонування системи. Паралельна робота призводить до конфліктів, які може вирішити, наприклад, керівник групи за вектором годин. Знання того, як оцінити вікно незгоди, допоможе запобігти конфлікту в цій ситуації.

У випадку багатофазної обробки не повинно бути невідповідності, оскільки вихід однієї фази використовується як вхід іншої. Знання затримки відповіді системи при забезпеченні суворої узгодженості дозволяє зробити точну оцінку часу виконання всього багатофазного завдання.

Коли з одним записом бази даних одночасно працює велика кількість користувачів, кількість версій цього запису може бути великою. Тому виникає проблема оцінки навантаження на користувача в залежності від кількості користувачів, які одночасно працюють із записом: оцінка кількості версій запису, часу їх обробки користувачем тощо. За результатами цих оцінок можна давати рекомендації щодо максимально можливої кількості користувачів, які одночасно беруть участь в обговоренні.

2.3 Розгорнута постановка завдання

Відповідно до технічного завдання магістерської роботи буде реалізовано програмне забезпечення, що відображає роботу ІС з концепцією NoSQL.

У процесі розробки магістерської роботи необхідно виконати наступний обсяг робіт:

а) проаналізувати існуючі аналогові системи для виявлення їх позитивних і негативних характеристик. Результати аналізу будуть враховані при подальшій розробці. Проаналізувати особливості узгодження реплік в базах даних NoSQL, а сосбливо MongoDB.

б) здійснити порівняльний аналіз відомих HR-систем, виділити їх основні переваги та недоліки;

в) розробити системне програмне забезпечення, що забезпечує реалізацію поставленого технічним завданням завдання. Побудувати блок-схеми програмних алгоритмів і підпрограм;

г) організувати інтерфейс користувача для створення та відображення повідомлень про некоректні дії користувача та нетипові ситуації;

д) розробити рекомендації щодо організаційно-методичних заходів щодо забезпечення впровадження системи у промислове використання та її подальшої успішної експлуатації;

д) виконати розрахунки для визначення економічної ефективності розробленої системи;

ж) розробити заходи з охорони праці під час впровадження та експлуатації системи та розробити заходи з охорони праці;

з) зробити висновки про обсяги виконаної роботи та досягнуті результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Аналіз і вибір типу СУБД та мови програмування

Етап аналізу полягає у вивченні системних вимог і проблем. Більш точно зміст поняття аналізу відображають терміни аналіз вимог (тобто дослідження системних вимог) і об'єктно-орієнтований аналіз (дослідження об'єктів предметної області). У процесі об'єктно-орієнтованого аналізу основна увага приділяється визначенню та опису об'єктів (або понять) з точки зору предметної області.

Аналіз вимог може включати опис процесів або сценаріїв використання програми, які можуть бути представлені у формі прикладів. Об'єктно-орієнтований аналіз відноситься до опису предметної області з точки зору класифікації об'єктів. Декомпозиція предметної області завдання полягає у виявленні понять, атрибутів і асоціацій з предметної області, важливих для вирішення завдання. Результат аналізу виражається в моделі предметної області (моделі предметної області), яка ілюструється набором діаграм, що показують поняття або об’єкти предметної області.

Найлогічніше було б виділити процеси, прив'язавши їх до існуючих структурних елементів. Існуючі елементи створені за функціональним принципом - виконання будь-якої функції, створення готового продукту. Таким чином, розподіл процесів здійснюватиметься в рамках існуючої системи управління. Процеси в домені можна описати у формі прецедентних словесних описів у структурованому форматі. Прецедент — це набір взаємопов’язаних сценаріїв успіху та невдачі, які описують використання виконавцем системи для вирішення запиту. На основі запитів зацікавлених осіб [11] сформовано запити, описані нижче.

Функціональний. Функції з назвою «керування» припускають наявність опцій: додати, змінити, видалити тощо.

Опис інформаційної системи

У магістерській роботі реалізована інформаційна система, яка зберігає інформацію про людські ресурси, освіту всіх співробітників, їхній останній досвід, графіки роботи, управління відпустками та поточні розподілені проекти. Буде створено адміністративну панель для управління всією інформацією про співробітників. Він міститиме статистичні дані про рівень зайнятості працівників різних установ та середній показник плинності кадрів з різних установ.

Цей проект буде побудовано на Node Express, який є веб-додатком для Node.js, а інтерфейс побудовано з використанням HTML, Bootstrap, CSS і JS.

Систему утворюють наступні основні структурні модулі:

- окремі облікові записи користувачів для всіх адміністраторів і співробітників;

- різні види даних і чіткий доступ до даних для учасників на основі їхніх привілеїв;

- реєстрація співробітників/адміністраторів;

- тайм-менеджмент усіх співробітників;

- управління заробітною платою всіх працівників.

- ведення обліку загального стажу та практичного досвіду працівника.

- управління поточними розподіленими проектами співробітника всередині організації.

У системі вибираються такі групи користувачів:

– Адміністратор. Має повний доступ до системи, що включає реєстрацію співробітників; вирішувати привілеї для інших співробітників, переглядати та змінювати сьогоднішню відвідуваність, переглядати та змінювати заробіток працівника, видаляти записи або профіль працівника, призначати та перепризначати проект працівникові, схвалювати або відхиляти запит працівника на відпустку.

– Персонал. Він зможе позначати свій графік, переглядати свою історію, переглядати поточні зарплати, переглядати поточний профіль співробітника (включаючи освітню та трудову історію), переглядати всі свої проекти в організації, бачити інших співробітників, які поділяють той самий проект з ним, подавати запит на відпустку та переглянути статус заяв на відпустку.

– Лідер проекту. Він може ознайомитися з навичками співробітників, надати оцінку продуктивності співробітника, а також мати доступ до співробітників на проекті, де він є керівником.

– Бухгалтерський працівник: може створювати платіжну відомість для кожного працівника, встановлювати бонуси для працівника, установлювати зарплату для працівника, підвищувати зарплату працівника, надсилати виписку про зарплату електронною поштою кожному працівнику.

Розробка архітектури інформаційної системи

Архітектура програми визначає її компоненти, функції та взаємодію. Розроблений додаток включає взаємодію клієнт-сервер, ідентифікацію та авторизацію користувачів, зберігання даних.

Програма повинна передавати запити по мережі. Архітектура програми повинна включати клієнтську та серверну частини.

Робота клієнт-серверної програми повинна працювати наступним чином:

- клієнт створює і відправляє запит на сервер;

- сервер виконує необхідні маніпуляції з даними і надсилає запит до бази даних;

- сервер отримує результат запиту з бази даних, формує результат і передає його клієнту;

- клієнт отримує результат, виводить його на пристрій виведення і чекає подальших дій користувача.

Цикл повторюється, доки користувач не закінчить роботу з сервером. Для вирішення цієї бізнес-проблеми була обрана клієнт-серверна архітектура (рис.

3.2). Він найбільше підходить для деталей такого типу, коли необхідно забезпечити доступ до системи великої кількості користувачів. Найголовніше те, що клієнт-серверна архітектура забезпечує можливість віддаленого доступу [2].

Проект, побудований за архітектурою клієнт-сервер, повинен складатися з трьох частин: зв'язку з базою даних, представлення даних клієнту та бізнес-логіки проекту, яка обробляє запити користувачів і відображає саме ту інформацію, яку бажав користувач.

Обробка та зберігання даних відбувається на стороні сервера відображення даних і відправка запитів на їх модифікацію здійснюється на стороні клієнта.

У процесі проектування основна увага приділяється концептуальному рішенню (у вигляді програмного або апаратного забезпечення), яке забезпечує виконання основних вимог. Наприклад, програмні об'єкти або схеми баз даних описуються на етапі проектування.

Концепцію проектування можна розділити на об'єктно-орієнтоване проектування (об'єктно-орієнтоване проектування) і проектування бази даних (проектування бази даних). У процесі об’єктно-орієнтованого проектування визначаються програмні об’єкти та методи їх взаємодії з метою виконання вимог системи.

Об'єктно-орієнтоване проектування відноситься до визначення об'єктів програмного забезпечення, їх відповідальності та методів взаємодії. Діаграма послідовності використовується для ілюстрації зв’язків між об’єктами, яка є типом діаграми взаємодії UML. Він показує потоки повідомлень між програмними об’єктами та викликами методів. На додаток до динамічного представлення взаємозв'язків об'єктів, яке показано на діаграмі взаємодії, дуже корисно побудувати фрагмент системи у вигляді діаграми класів проектування.

3.2 Розробка структурної схеми

Структурна схема сайту – це сукупність об’єктів та частин сайту та взаємозв’язки між ними. Призначенням структурної схеми є наглядне відображення складових частин сайту, його основних блоків, вузлів та взаємозв’язок між ними.

Структурна схема розробленої системи зображена на рисунку 3.12. На ній показано структуру.

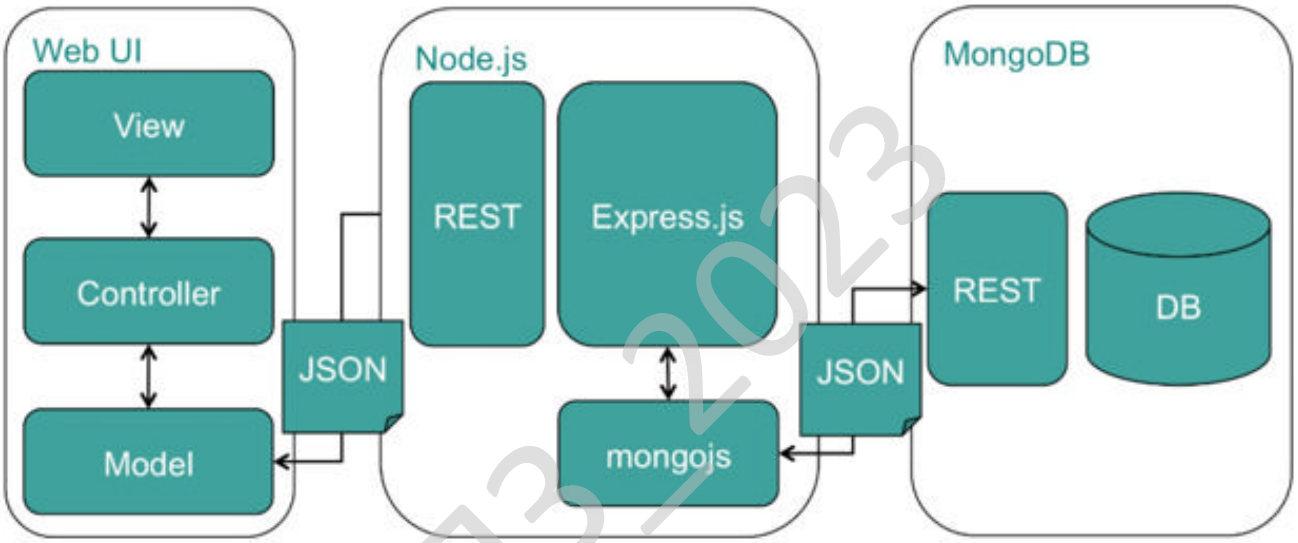


Рисунок 3.1 – Структурна схема

Ця структурна схема показує, як кожен компонент взаємодіє з іншими, створюючи цілісну систему, яка забезпечує веб-додаток з серверною частиною, базою даних та іншими ключовими функціональними блоками.

Структурна схема системи включає в себе різні компоненти, які взаємодіють між собою для забезпечення функціональності системи. Нижче наведено короткий опис кожного блоку та їхній взаємодії:

WEB UI (користувацький інтерфейс):

– відповідає за візуальне взаємодію з користувачем;

– взаємодіє з сервером через HTTP-запити, ініціюючи запити на

отримання або оновлення даних.

NodeJS:

- використовується як середовище виконання для серверної сторони;
- обробляє HTTP-запити від WEB UI та взаємодіє з іншими компонентами системи.

Express.js:

- веб-фреймворк для Node.js, який допомагає швидко створювати API;
- обробляє маршрутизацію та керує HTTP-запитами.

Controller:

- обробляє бізнес-логіку за даними, отриманими від Express.js;
- керує взаємодією між WEB UI і базою даних.

MongoDB:

- нереляційна база даних, де зберігаються дані;
- Controller взаємодіє з MongoDB для зберігання, читання та оновлення даних.

Rest:

- використовується для створення API для взаємодії між WEB UI та серверною частиною системи;
- визначає стандарти для обміну даними між компонентами через HTTP-протокол.

JSON:

- використовується для форматування та обміну даними між компонентами;
- дані, що передаються між WEB UI, NodeJS, та базою даних, можуть бути у форматі JSON.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.1.

Функціональна схема допомагає розібрати ролі кожного компонента та

його взаємодію для забезпечення функціональності системи як цілісної одиниці.

Розглянемо кожен з блоків окремо.

"UseInterface" - це компонент, що відповідає за користувацький інтерфейс (UI) веб-додатка. Цей блок включає в себе ряд технологій і ресурсів для створення зручного і естетичного інтерфейсу для користувача. Давайте розглянемо кожну складову цього блоку більш детально:

HTML (HyperText Markup Language)

HTML є основним мовою розмітки для створення структури веб-сторінок.

Використовується для визначення елементів, таких як текст, зображення, форми тощо.

Лежить в основі структури сторінки та визначає, які елементи будуть відображені.

JS (JavaScript)

JavaScript використовується для динамічного взаємодії з користувачем на сторінці.

Забезпечує можливість реагувати на події, змінювати вміст сторінки, валідувати введені дані та інше.

Використовується для взаємодії з сервером через AJAX-запити для асинхронного завантаження даних.

CSS (Cascading Style Sheets)

CSS відповідає за оформлення та стилізацію елементів, визначених в HTML.

Визначає зовнішній вигляд, кольори, розміри та розташування елементів на сторінці.

Дозволяє створювати адаптивний та привабливий дизайн.

Fonts (Шрифти)

Вибір і використання шрифтів впливає на зручність читання та загальний вигляд інтерфейсу.

Можливість використовувати різні шрифти для створення унікального

стилю веб-додатка.

Images (Зображення)

Використовуються для візуального представлення інформації.

Зображення можуть бути фоном, ілюстраціями, логотипами тощо.

Оптимізація та коректне використання зображень покращують продуктивність інтерфейсу.

Блок RequstManegment

Цей блок описує архітектуру та функціональність серверної частини веб-додатка, що використовує фреймворк Express.js, менеджер маршрутів, контролери, API для взаємодії з клієнтською стороною, бібліотеку Mongoose для роботи з базою даних MongoDB та моделі для визначення структури даних.

"Express" у контексті функціональної схеми веб-додатка вказує на використання фреймворка Express.js для реалізації серверної частини додатка. Давайте розглянемо кожну частину цього блоку:

Express.js

Express.js - це веб-фреймворк для Node.js, який дозволяє швидко створювати веб-додатки та API.

Функції:

– маршрутизація: Визначення шляхів та обробників для обробки HTTP-запитів;

– обробка запитів та відповідей: Взаємодія з HTTP-запитами та формування відповідей;

Route Manager (Менеджер маршрутів)

Опис: Відповідає за управління маршрутами в Express.js.

Функції:

– реєстрація маршрутів: Визначення шляхів та обробників для обробки запитів;

– керування потоком обробки запитів відповідно до визначених маршрутів.

Controllers (Контролери)

Контролери в Express.js відповідають за обробку бізнес-логіки додатка на серверному боці.

Функції:

– обробка запитів: Виконання логіки, пов'язаної з конкретними маршрутами.

– взаємодія з моделями: Отримання та оновлення даних з бази даних через моделі.

API (Application Programming Interface)

Визначає інтерфейс для взаємодії між різними компонентами програмного забезпечення.

Функції:

– визначення структури та формату даних для обміну.

– надання можливості іншим частинам системи викликати конкретні функції чи отримувати дані.

Mongoose (Moongowe)

Mongoose - бібліотека для взаємодії з базою даних MongoDB у середовищі Node.js.

Функції:

– Визначення схем даних: Створення моделей даних та валідація структури документів.

– Взаємодія з базою даних: Виконання операцій читання, запису, оновлення та видалення даних в MongoDB.

Models (Моделі)

Опис: Моделі в Express.js використовуються для представлення та взаємодії з даними у базі даних.

Функції:

– опис схеми даних: Визначення структури та типів даних;

– використання методів Mongoose для роботи з базою даних.

Блок Module, який складається з блоків Email, ExcelExport, Auth та Tasks, вказує на наявність різних модулів або компонентів у веб-додатку.

Email (Електронна пошта)

Відповідає за взаємодію з електронною поштою у системі.

Функції:

– відправка електронних листів: Надсилання повідомлень електронною поштою.

– обробка сповіщень: Взаємодія з іншими модулями для відправки сповіщень.

ExcelExport (Експорт в Excel)

Забезпечує можливість експорту даних у формат Excel.

Функції:

– генерація Excel-файлів: Створення файлів у форматі, сумісному з Excel.

– забезпечення коректного відображення даних у таблицях Excel.

Auth (Автентифікація та авторизація)

Відповідає за здійснення процесу автентифікації та авторизації користувачів.

Функції:

– перевірка ідентифікації: Перевірка вірності введених даних користувача;

– надання доступу: Визначення рівня доступу користувача до різних ресурсів.

Tasks (Завдання)

Відповідає за управління завданнями або задачами в системі.

Функції

– створення та редагування завдань: Можливість додавати нові завдання та редагувати існуючі;

– відстеження стану завдань: Визначення стану виконання та інших

параметрів завдань.

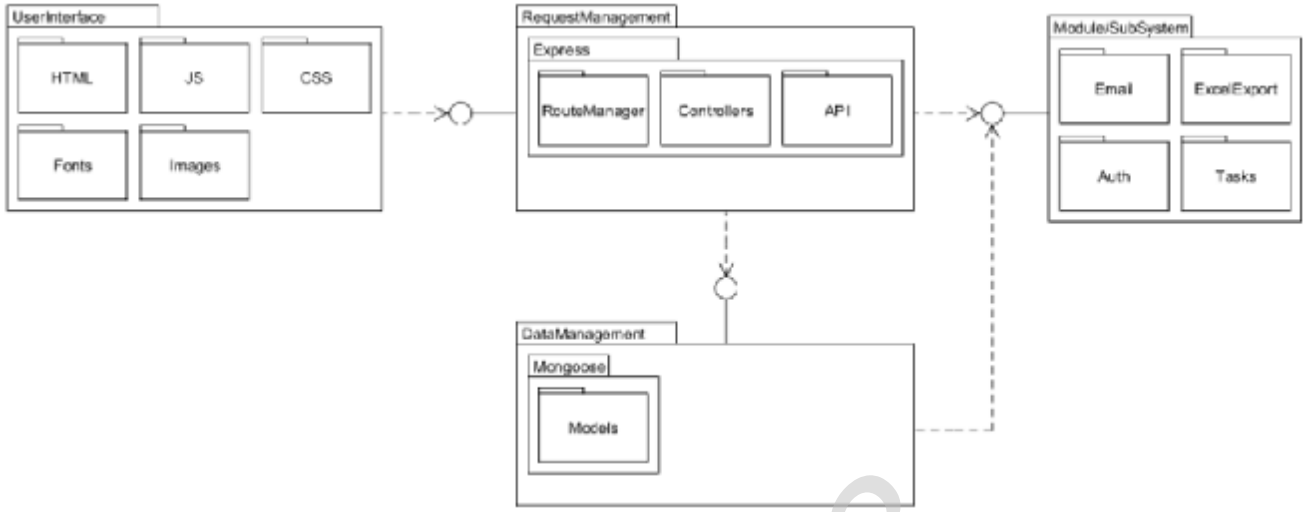


Рисунок 3.2 – функціональна схема

3.4 Розробка діаграми процесів

Діаграми процесів, як і діаграми послідовності, належать до діаграм взаємодії. Вони ілюструють взаємодію об’єктів у форматі графіка чи сітки, відображають потік подій і зосереджують увагу на зв’язках між об’єктами.

Моделювання виконання операції виконується наступним чином:

1) Ідентифікація параметрів, повернених значень та інших об'єктів, видимих для операції.

2) Якщо операція тривіальна, візуалізуйте її реалізацію безпосередньо в коді, який можна розмістити на задньому плані моделі або явно візуалізувати в анотації.

3) Якщо операція є алгоритмічно складною, ми змоделюємо її виконання за допомогою діаграми діяльності.

4) Якщо операція потребує великого обсягу детального проектування, розгляньте можливість спільної реалізації. У майбутньому ви зможете розширити структурні та поведінкові компоненти співпраці за допомогою

діаграм класу та взаємодії відповідно. Відношення - це зв'язок між двома примірниками класів, які визначають певну форму руху та видимості між ними. Посилання є прикладом асоціації. Повідомлення, що передаються між об’єктами, представлені як імена цих повідомлень над лініями з’єднання зі стрілками. Над однією лінією зв’язку можна перерахувати будь-яку кількість повідомлень. Щоб показати послідовність повідомлень у поточному потоці керування, поруч із повідомленням вказується порядковий номер.

Діаграми співпраці, як і діаграми послідовності, можна використовувати для відображення взаємодії як між об’єктами домену, так і між об’єктами програмного забезпечення.

Існують різні способи організації взаємодії між клієнтом і сервером. Одними з найпопулярніших технологій є WebSocket і AJAX. Суть технології AJAX полягає в зміні вмісту завантаженої веб-сторінки без повного перезавантаження, завдяки чому досягається висока динамічність сайтів. Технологія заснована на обміні даними та навчанні певних компонентів за потреби. AJAX з'явився в 1998 році і тому підтримується старими версіями різних браузерів [9].

WebSocket — це стандартна технологія HTML5, яка дозволяє встановлювати повнодуплексне TCP-з’єднання між сервером і веб-браузером (клієнтом). Ця технологія покликана вирішити проблему зняття обмежень на обмін даними між браузерами і серверами. Браузер повинен мати підтримку WebSocket.

Розроблено діаграму процесів, щоб розглянути деякі передумови для вибору технічних або інших інструментів і технологій, а також дизайну програми. Визначимо кілька варіантів використання за схемою.

Діаграму процесів зображено на рисунку 3.3.

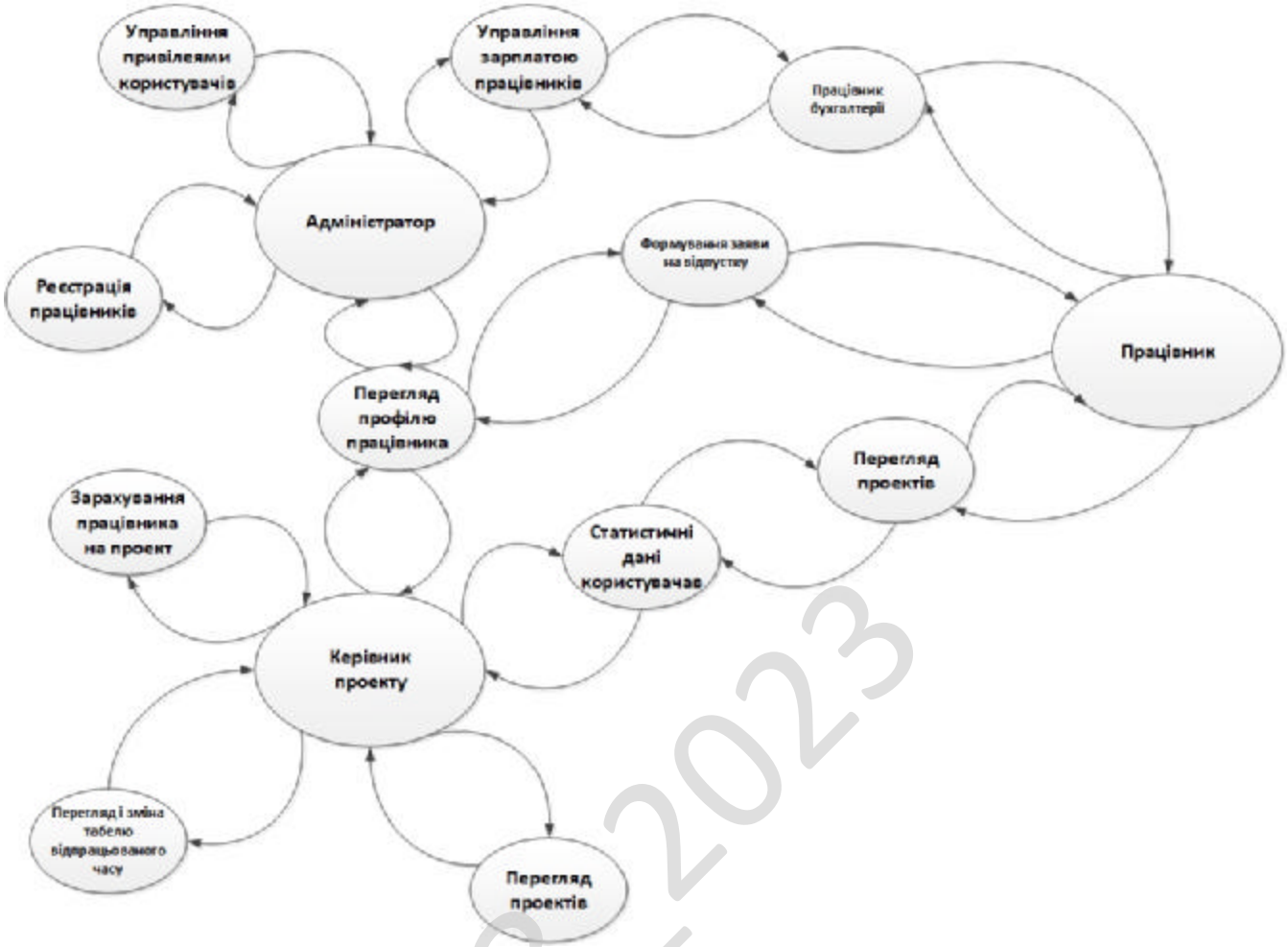


Рисунок 3.15 – Діаграма процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

Опис технологій для створення системи

Аналіз сучасних веб-технологій показав, що платформа Node.js підходить для побудови системи. Node.js — це середовище JavaScript, створене на движку JavaScript Chrome V8. Node.js використовує неблокуючу модель введення-виведення на основі подій, що робить його легким і ефективним.

Модель, прийнята в Node.js (рисунок. 4.1), принципово відрізняється від звичайних платформ для побудови серверів, в яких масштабованість досягається завдяки багатопоточності. Завдяки подієво-орієнтованій архітектурі зменшується споживання пам'яті, підвищується пропускна здатність.

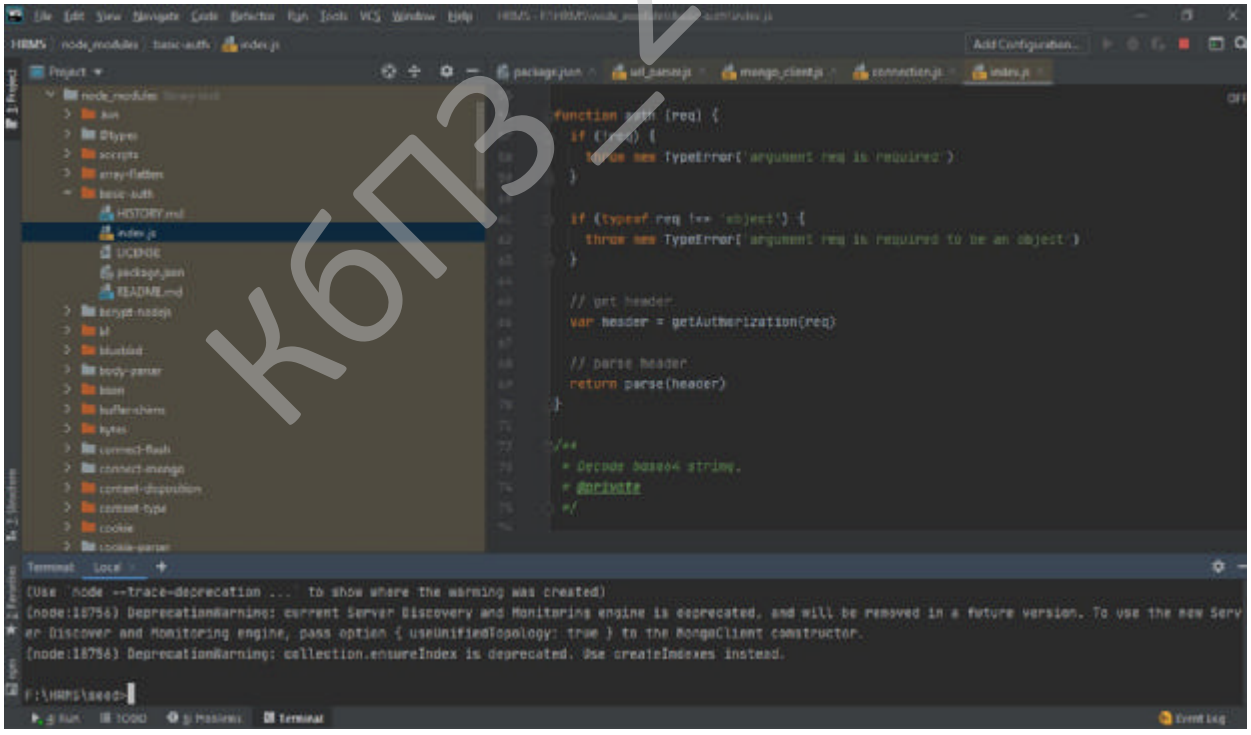


Рисунок 4.1 - Середовище розробки з Node.js

У Node.js при старті створюється єдиний програмний потік. Node.js виконується в цьому потоці. Це означає, що дві або більше частини додатка одночасно не можуть виконуватися. Можливість “одночасного” виконання коду надається за допомогою використання асинхронної моделі, побудованої на черзі подій (event-loop). На рисунку 4.2. зображена діаграма розгортання.

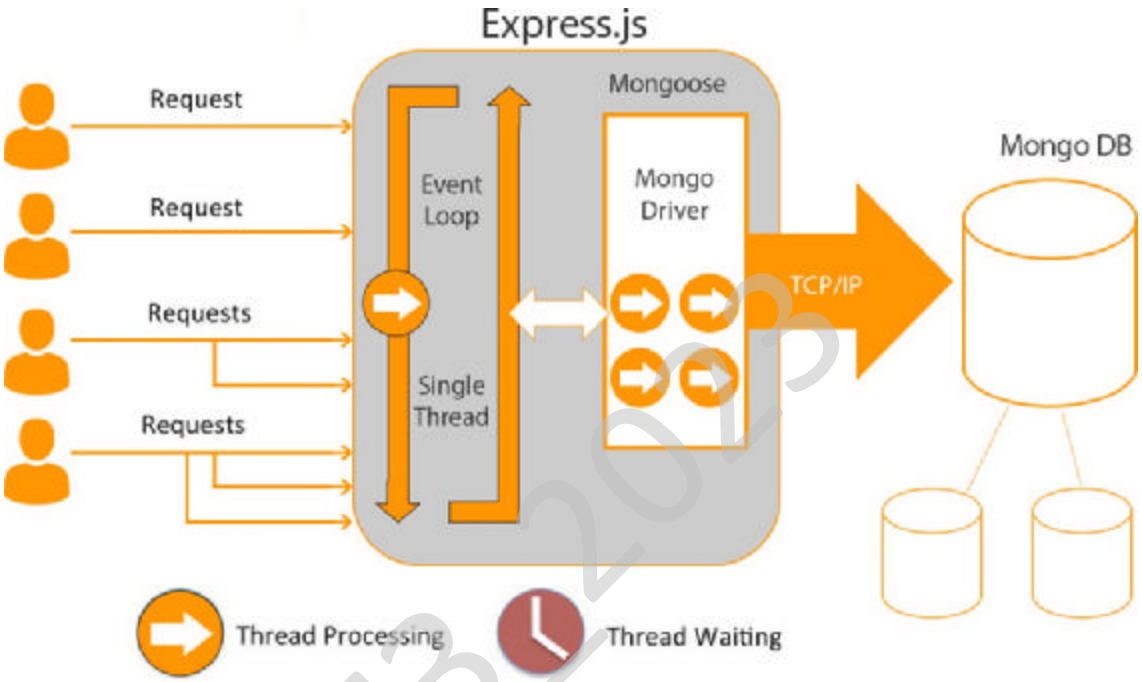


Рисунок 4.2 - Діаграма розгортання, включаючи Node.js Express, MongoDB

База даних MongoDB була обрана як СУБД з наступних причин:

- можливість обробки неструктурованих даних;
- відсутність або частковий зв'язок між колекціями;
- збереження даних не вимагає високої продуктивності.

Кожен екземпляр знаходиться в базі даних у впорядкованій колекції (рисунок 4.3).

Загальна архітектура веб-додатку. Система розробки базується на клієнт-серверній архітектурі. Клієнт робить запит серверу на виконання певних дій або надання інформації, сервер отримує запит, виконує його і надсилає відповідь.

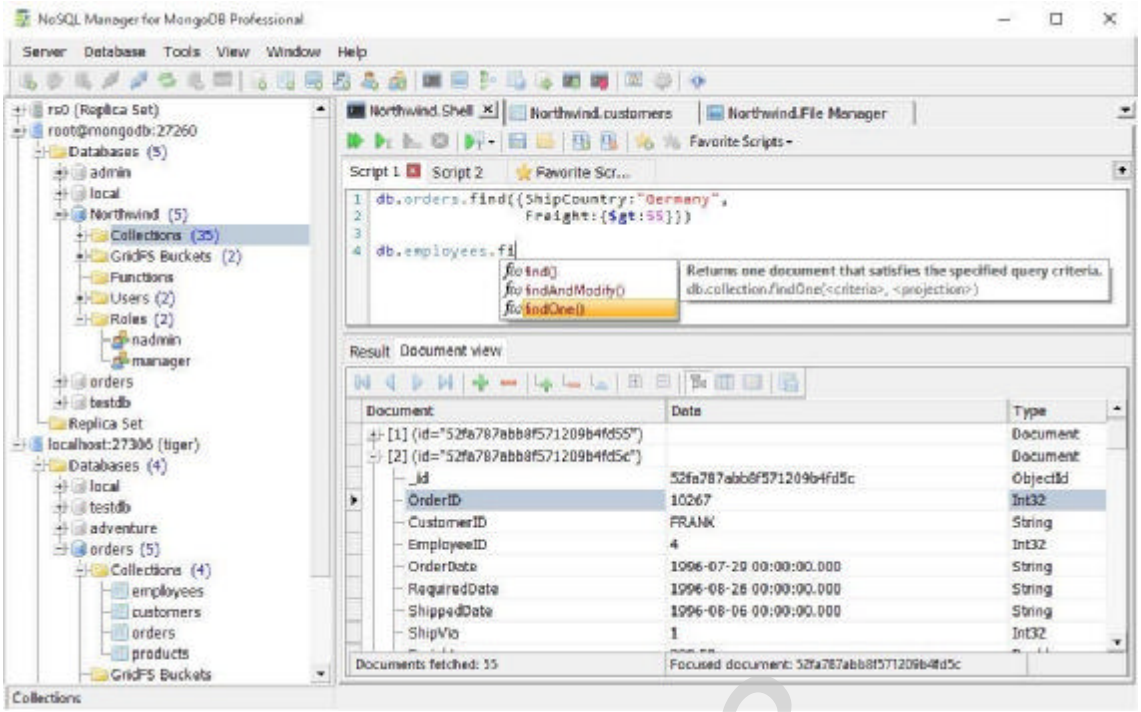


Рисунок 4.3 - MongoDB Manager

Серверна частина системи управління кадровими процесами обробки запитів повинна мати три основні компоненти, які будуть відповідати за: обробку запиту, попередню обробку надісланих даних і обробку маршруту. У компоненті, відповідальному за обробку запиту, об’єкт даного запиту створюється та пересилається компоненту, відповідальному за обробку надісланих даних, після чого екземпляр, створений для обробки запиту, звільняється.

Функції, що відповідають проміжній обробці надісланих даних, у «Express» називаються «проміжним програмним забезпеченням». Вони об'єднуються в ланцюги і після виконання всього «проміжного програмного забезпечення» конкретного ланцюжка управління передається компоненту, що відповідає за обробку маршруту. Звільнення екземпляра обробки запиту та подальша передача управління компоненту, що відповідає за обробку маршруту, здійснюється за допомогою функції зворотного виклику.

Побудова серверної частини за допомогою програмної платформи Node.js і документо-орієнтованої бази даних MongoDB дозволяє створювати

високошвидкісні додатки, орієнтовані на велику кількість одночасних підключень, вимагаючи невеликого обсягу оперативної пам’яті.

Програмна реалізація системи

Для розробки проекту було вирішено використовувати мову документів JavaScript з платформою Node.js. Це безкоштовна платформа, яка дозволяє писати програми на JavaScript і містить багато корисних модулів, тому немає необхідності писати код «з нуля».

Node.js складається з середовища виконання та бібліотек. При створенні програми були використані такі бібліотеки JavaScript:

- ekpress.js – фреймворк для практичного створення веб-додатків [12];

- socket.io - бібліотека JavaScript для реалізації з'єднання Web-Socket у веб-додатках і обміну даними в реальному часі. Він складається з двох частин: клієнта, який працює в браузері, і сервера для node.js. Обидва компоненти мають подібний API [14];

- WebStorm - незалежний редактор коду з підсвічуванням синтаксису, темами і гарячими клавішами, код, написаний на JavaScript, легко вбудовується в будь-яку веб-сторінку [10];

- monk.js – обгортка для бази даних MongoDB, яка дозволяє швидко та зручно взаємодіяти з колекціями баз даних [13].

WebSocket був обраний для реалізації зв’язку в реальному часі між клієнтом і веб-сервером. При такому з'єднанні обидві частини (клієнт і сервер) є рівноправними і мають можливість перевірити з'єднання за допомогою керуючих кадрів PING і PONG. Той, хто хоче перевірити з'єднання, надсилає кадр PING із довільним тілом. Його приймач повинен відповісти кадром PONG з тим самим тілом протягом прийнятного часу.

Обробники подій розроблені та реалізовані для обміну даними між стороною клієнта та сервером. Список обробників подій на стороні клієнта показано на рисунку 4.4.

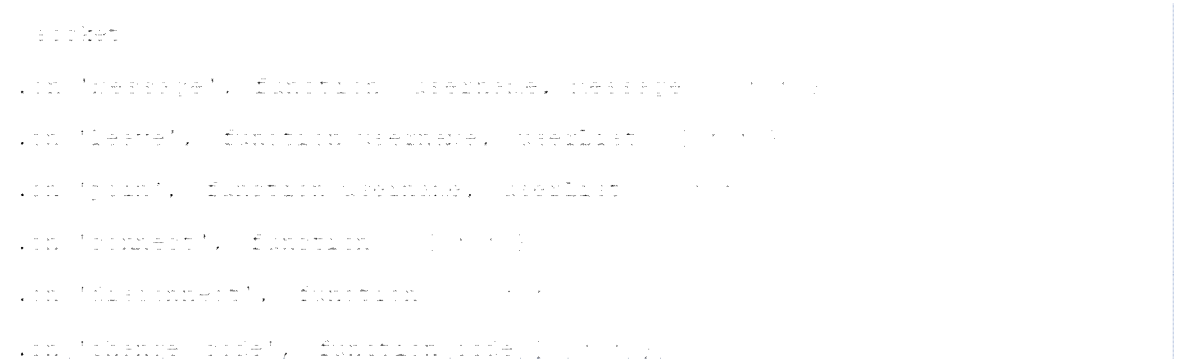


Рисунок 4.4 - Список обробників подій на стороні клієнта

Обробники подій на стороні клієнта можна більш детально описати наступним чином:

- повідомлення - подія при відправленні повідомлення в чаті веб-додатку;
- вихід - подія при закритті всіх вкладок браузера учасника, виході з його документа;
- приєднання - подія підключення учасника до системи;
- connect - подія під час активного з'єднання з сервером;
- disconnect - подія, коли з'єднання з сервером втрачається;
- change - подія, коли змінюється доступ до проекту;
- change lang - подія при зміні мови;
- зміна прав - подія, коли змінюються права певного учасника.

Також були розроблені та реалізовані обробники подій на стороні сервера, які показані на рисунку 4.5.

```
socket
.on('add user', function(user, cb) {/*…*/})
.on('message', function(text, cb) {/*…*/})
.on('disconnect', function(username, cb) {/*…*/})
.on('change code', function(code, cb) {/*…*/})
.on('change lang', function(lang) {/*…*/})
```

Рисунок 4.5 - Список обробників подій на стороні сервера

Слід розуміти, що хоча назви деяких подій на стороні клієнта та на стороні сервера однакові, реалізація подій принципово відрізняється. Так, наприклад, при обробці події повідомлення на стороні клієнта повідомлення, отримане від сервера, повинно відображатися в чаті, тоді як сервер обробляє цю подію, надсилаючи повідомлення всім учасникам чату.

На етапі проектування було вирішено використовувати документоорієнтовану СУБД NoSQL. Таких систем декілька: CouchDB, Couchbase, MarkLogic, mongoDB, eXist тощо. При цьому важливо, щоб обрана система забезпечувала зручну взаємодію з реалізованим веб-додатком.

З перерахованих СУБД MongoDB є найпопулярнішою для обраної платформи. Це розширювана, високопродуктивна СУБД із відкритим вихідним кодом без схем, яка зберігає всі дані у двійковому форматі JSON (BSON). Головною відмінністю MongoDB є її гнучке використання у веб-додатках. Колекції з полями, показаними на рисунку 4.6, призначені для зберігання даних документів, створених користувачами.

У MongoDB база даних складається з колекцій. Як правило, колекція відображає сутність предметної галузі. Колекція складається з «документів» (далі — записи). Запис — це об’єкт JSON з обов’язковим ідентифікатором ObjectId, який MongoDB автоматично встановлює та контролює його унікальність, аж до унікальності в усьому світі [12]. ObjectId — це тип даних. У порівнянні з реляційною СУБД колекція — це таблиця, а запис — рядок у таблиці.

Тип даних, що зберігається в записі, є довільним, наприклад: рядок; Масив (структура даних, доступ до якої здійснюється за цілим індексом); логічний; дата; Ціле число; Нуль. Об’єкт (структура даних, що дозволяє отримати доступ до елементів за ключем).

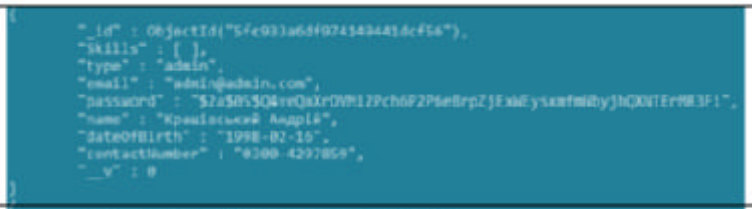
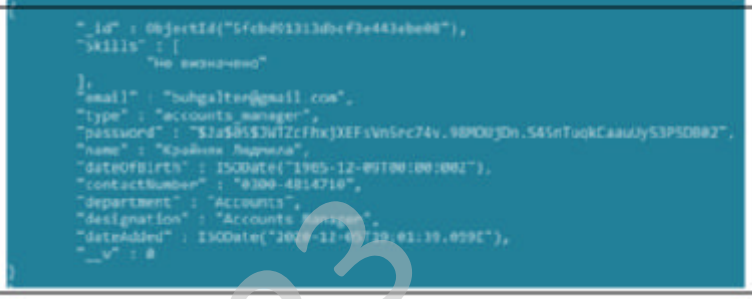
MongoDB не підтримує реляційну цілісність. Це завдання вирішується на рівні коду сервера додатків, який реалізується за допомогою засобів ODM, - Мангуст.

Рисунок 4.6 - Модель документів бази MongoDB

Колекція містить ідентифікатор користувача, який створив деталь, ідентифікатор проектів, деталі поточного отримання реквізитів платежу та ідентифікатор запису в колекції. Було вирішено, що нова вкладка документа сприймається як картка того ж користувача і що з'єднання з клієнтом припиняється лише тоді, коли всі вкладки, відкриті цим документом, закриті.

Складність полягає в тому, що за стандартом WebSocket нова вкладка браузера є повноцінним з'єднанням клієнт-сервер. Таким чином, таблиця підключених користувачів створюється для конкретного проекту, щоб відстежувати всі сокети, які підключив користувач. Модель збережених даних наведена в таблиці 4.1.

Таблиця 4.1 - Приклад списку формування списку працівників

Ідентифікатор користувача	Роль	Реалізований Json-документ
5fc933a6df97 4149441dcf56	Адміністратор	
5fcbd91313dbc f3e443ebe08	Бухгалтер	

Для того, щоб мати представлення зв’язку між сутностями, була розроблена схема бази даних, на якій показані псевдозв’язки, рисунок. 4.7, нотація Гордона Евересту [21].

Додамо до схеми, що імена колекцій і псевдозв’язків можуть представляти дещо інше значення, ніж таблиці в реляційних базах даних, оскільки там є вкладені об’єкти. Прикладом є колекція під назвою product_type.

Типи псевдозв'язків документа з іншими сутностями з урахуванням неструктурованості даних:

- HasOne: має один тип документа.

- HasOne: має одну мову.

- ManiToMani: належить до одного або кількох продуктів (BelongsTo), оскільки є загальним. З іншого боку, один продукт має багато документів різними мовами (HasMani).

- HasMani: є багато змін.

- BelongsTo: Належить до того самого завантаження, електронної пошти або замовлення.

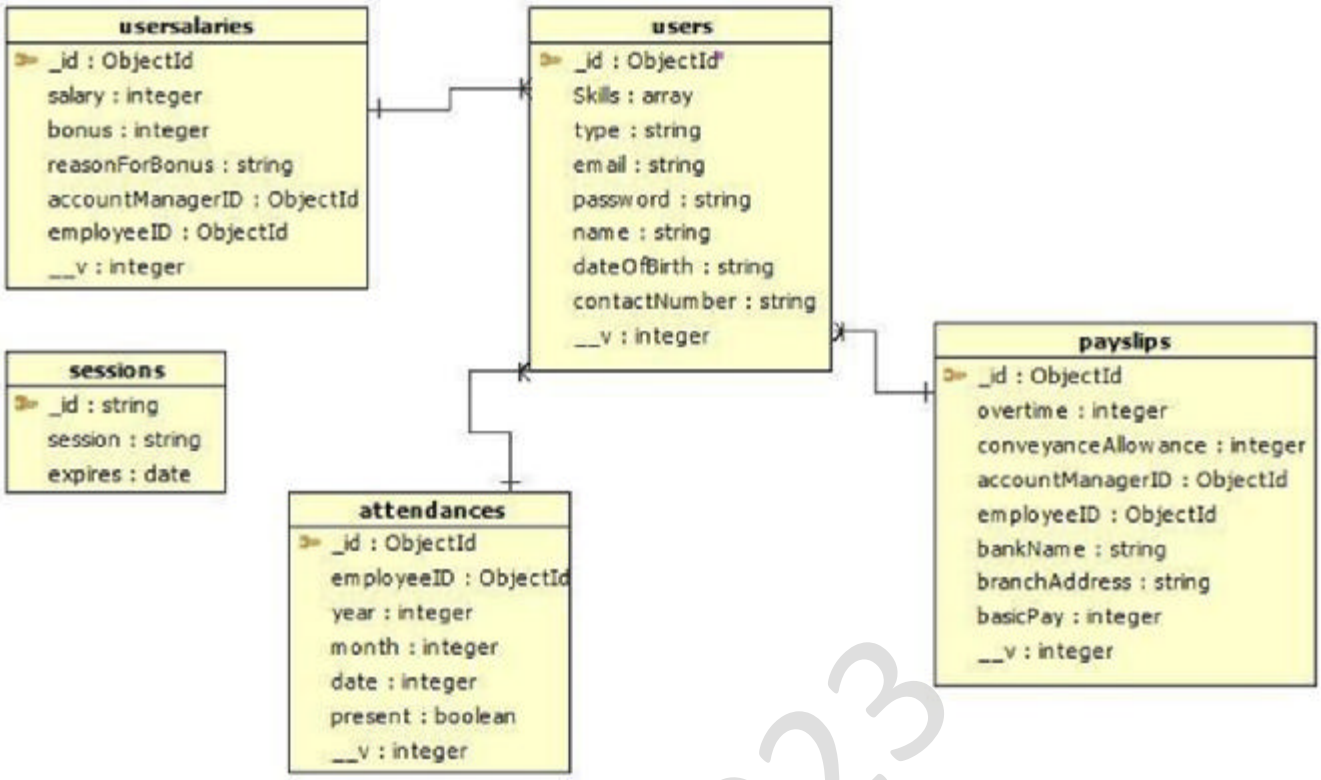


Рисунок 4.7 - Схема бази даних з позначенням псевдо-зв’язків

Наступні поняття використовуються для опису функцій дизайну схеми в MongoDB:

- Вбудовування - повне збереження об'єкта(ів) однієї сутності в іншій.

- Посилання - унікальний ідентифікатор ObjectId зовнішньої сутності. Використання інструменту Mongoose дозволяє автоматично генерувати ObjectId під час вставки вкладеного об’єкта. Наприклад, коли ви додаєте продукт до типу товару. Під час проектування були визначені деякі конструктивні особливості схеми бази даних у MongoDB, а саме:

- З композитними псевдоключами втрачається гнучкість роботи з даними.

- Зберігання лише ідентифікатора об’єкта зовнішньої сутності вимагатиме додаткового запиту для отримання всіх даних зовнішньої сутності.

Вкладеність об’єкта сутності має сенс, коли вкладений об’єкт з’являється разом із батьківським.

Вкладеність об’єкта сутності відображає зв’язок «один до багатьох» у реляційній схемі.

Посилання забезпечують більшу гнучкість із частими змінами вкладених об'єктів.

Рядок посилання на об’єкт сутності відображає зв’язок «багато до багатьох» у схемі відношення. Атрибути основної сутності Users наведені в таблиці 4.2, а на рисунку 4.8 показано структуру проекту інформаційної системи зберігання даних.

Таблиця 4.2 - Атрибути основної сутності Users

Найменування атрибуту	Тип	Опис
_id	ObjectId	Унікальний ідентифікатор Users
Skills	array	Практичні навички
type	string	Тип
email	string	email
password	string	Пароль
name	string	Прізвище, ім’я, по батькові
dateOfBirth	string	Дата народження
contactNumber	string	Контактний номер телефону
__v	date	Дата нагадування актуальності
department	string	Підрозділ
designation	string	Посада

При запиті користувача пошук за атрибутами: ім'я, контактний телефон, відділ, ярлик. Результат має повернути інформацію про користувача з відповідними даними результату.

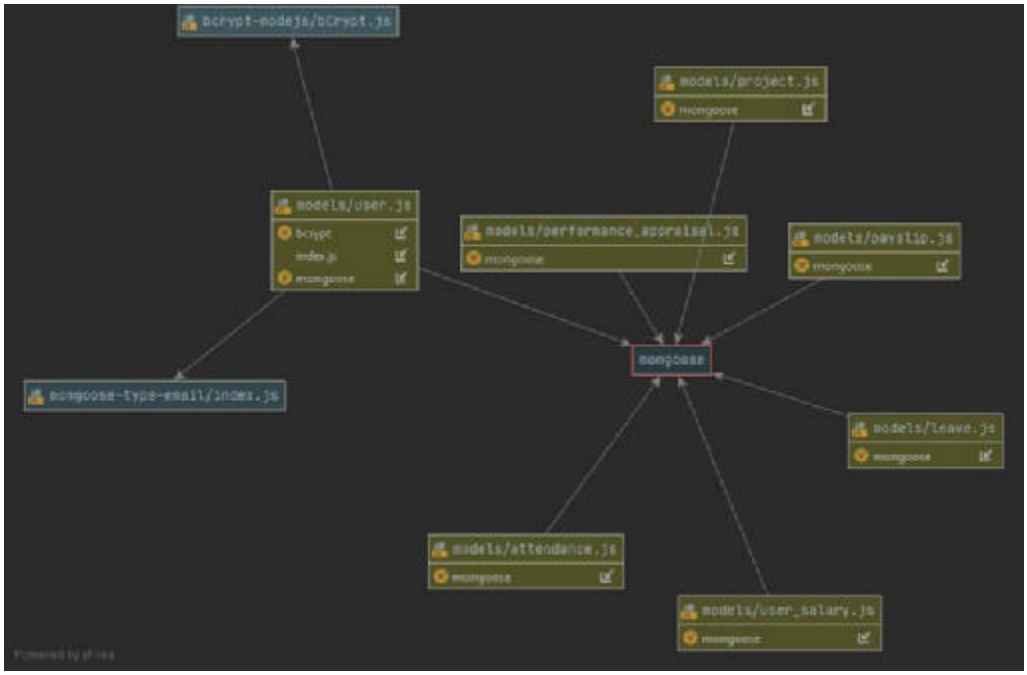


Рисунок 4.1 Структура проекту інформаційної системи для зберігання даних

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1.

Як тільки користувач потрапив на головну сторінку сайту він повинен пройти авторизацію або реєстрацію у випадку відсутності аккаунта. На рисунку 4.1 зображено блок-схему алгоритму роботи сторінки реєстрації користувача.

На сторінці реєстрації користувач має можливість пройти реєстрацію нового аккаунта для доступу до закритої частини сайту. Для успішної реєстрації користувач повинен надати свою фотографію, ім’я, поштову адресу та бажаний пароль. Після введення всіх даних система перевіряє заповненість усіх полів та присутність завантаженого зображення і, або виводить помилку, якщо щось було пропущено, або створює два записи у таблицях бази даних. Перший запис містить введену користувачем інформацію про його аккаунт, а другий містить інформацію потрібну для активації даного аккаунта, що зберігається у таблиці з інформацією про аккаунти, що очікують активації. Далі система надсилає листа

на введену користувачем поштову адресу з посиланням, яке містить код активації аккаунта, і повідомляє користувача, що на вказану ним адресу було відправлено листа з кодом активації.

Після того як користувач перейшов по отриманому посиланню система перевіряє інформацію у цьому посиланні, а саме код активації аккаунта і якщо даний код буде збігатися з кодом у рядку в таблиці з неактивованими аккаунтами – система видалить інформацію про неактивований аккаунт користувача та встановить, що даний аккаунт користувача активовано.

Потім система перенаправляє користувача на сторінку авторизації на якій він має можливість увійти на свій аккаунт та розпочати роботу.

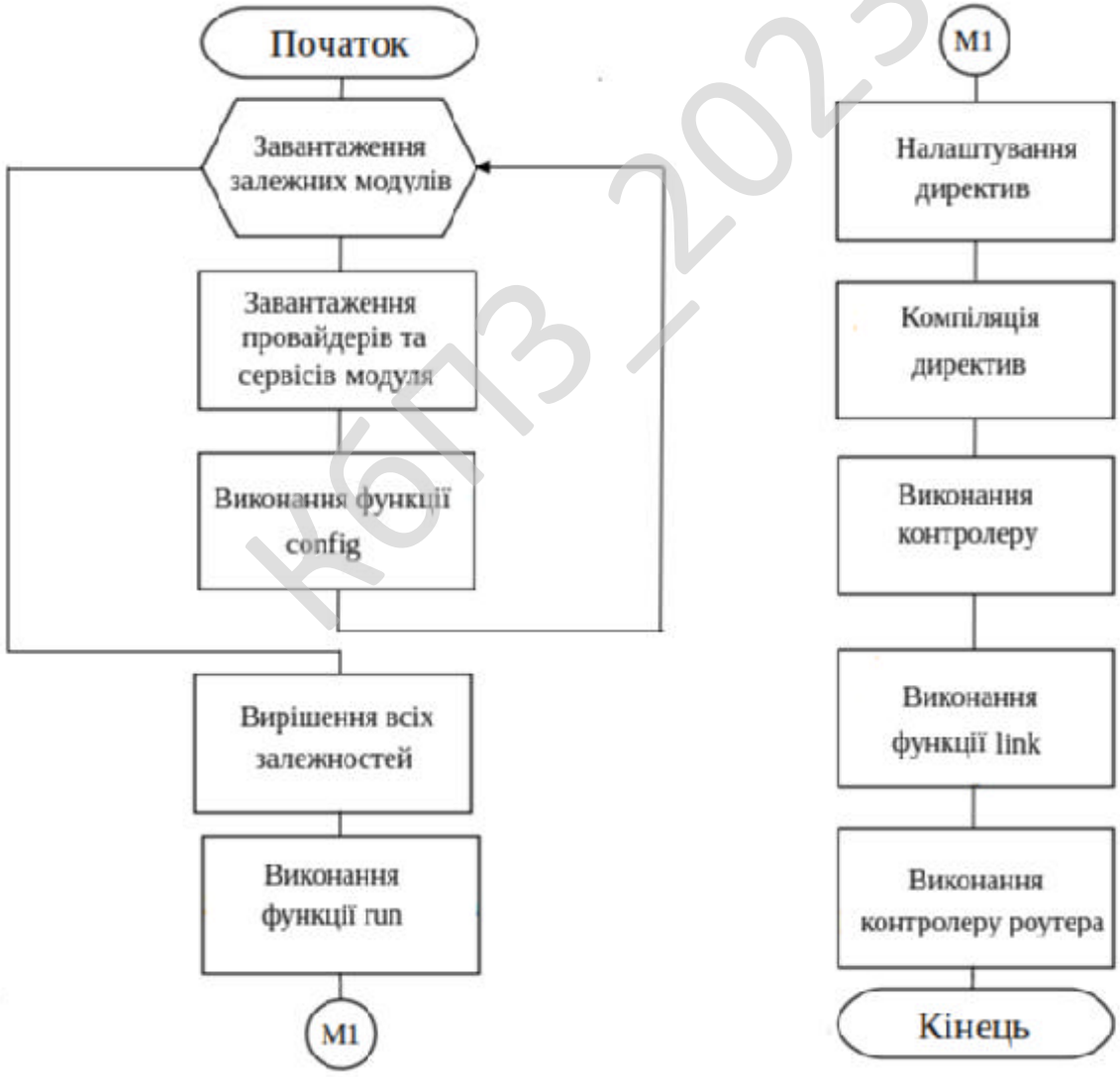


Рисунок 4.2 – Блок-схема алгоритму роботи ІС HR

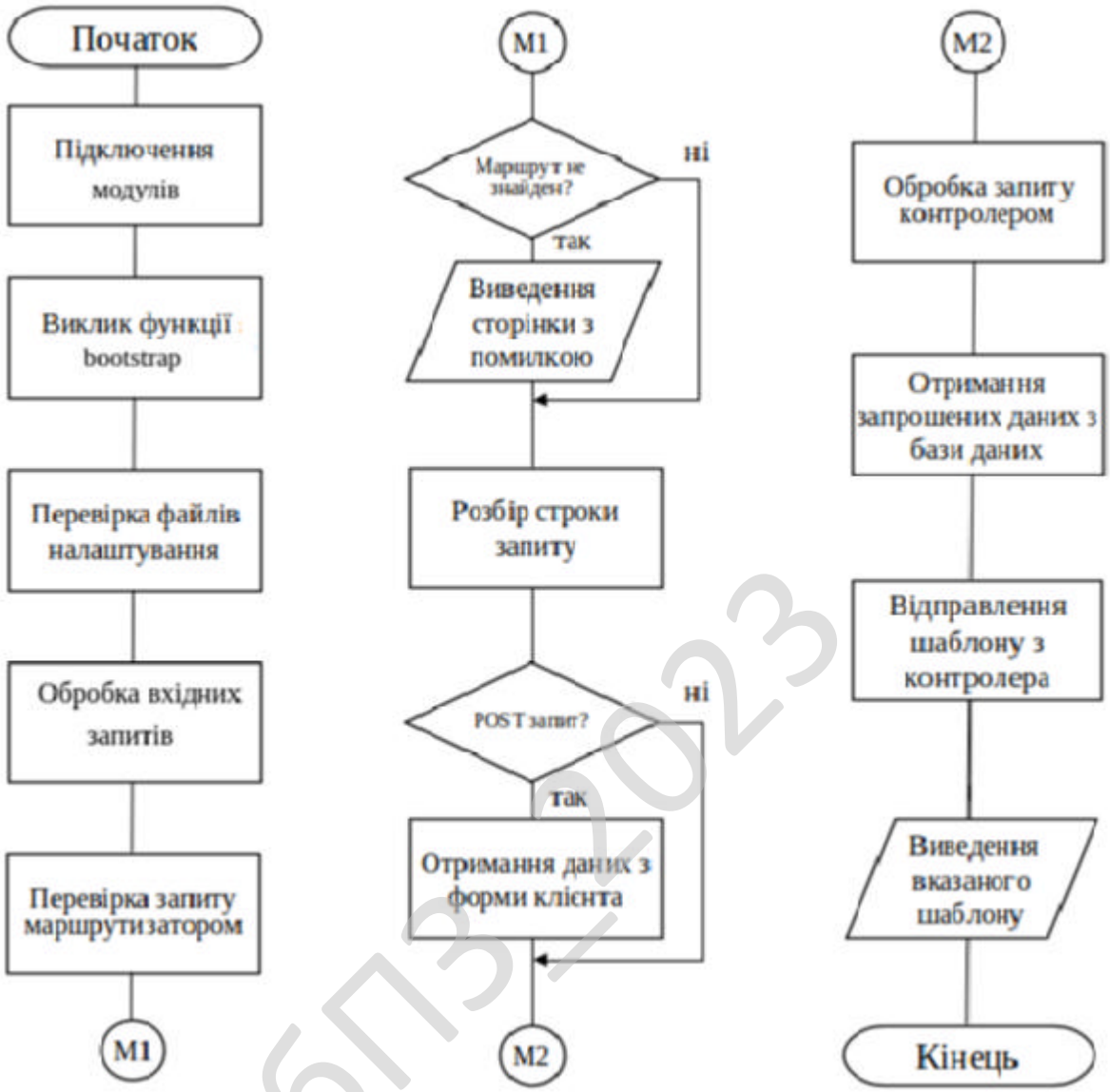


Рисунок 4.3 –Блок-схема алгоритму роботи основної програми

4.2 Захист розробленого програмного забезпечення

Для захисту даних користувачів на сайті використовується крипто алгоритм bcrypt.

bcrypt - адаптивна криптографічний хеш функція формування ключа, що використовується для захищеного зберігання паролів. Розробник: Нільс Провос. Функція заснована на шифрі Blowfish, вперше представлена на USENIX в 1999

році. Для захисту від атак за допомогою райдужних таблиць bcrypt використовує salt; крім того, функція є адаптивною, час її роботи легко налаштовується і її можна уповільнити, щоб ускладнити атаку перебором.

Шифр Blowfish відрізняється від багатьох алгоритмів обчислювально складною фазою підготовки ключів шифрування.

Провос скористався цією особливістю, але змінили алгоритм підготовки ключів, отримавши шифр «Eksblowfish» (expensive key schedule Blowfish). Кількість раундів в підготовці ключів повинно бути ступенем двійки; конкретна ступінь може здаватися при використанні bcrypt.

Спочатку реалізовано в функції crypt в OpenBSD. Існують реалізації для Java, Python, Nim, C #, Ruby, Perl, PHP 5.3, Node.js і деяких інших

Алгоритм bcrypt використовує алгоритм настройки ключів з «Eksblowfish»:

```
EksBlowfishSetup(cost, salt, key)
    state
    {\displaystyle \gets }
\gets  InitState()
    state
    {\displaystyle \gets }
\gets  ExpandKey(state, salt, key)
    repeat (2cost)
        state
    {\displaystyle \gets }

\gets  ExpandKey(state, 0, key)
        state
    {\displaystyle \gets }
\gets  ExpandKey(state, 0, salt)
    return state
```

ExpandKey(state, salt, key)

for(n = 1..18)

Pn

Функція InitState відповідає оригінальній функції з шифру Blowfish; для заповнення масиву P і S-box використовується дрібна частина числа {\ displaystyle \ pi} \ pi.

Функція ExpandKey:

{\displaystyle \gets }

\gets key[32(n-1)..32n-1]

{\displaystyle \oplus }

\oplus Pn //treat the key as cyclic

ctext

{\displaystyle \gets }

\gets Encrypt(salt[0..63])

P1

{\displaystyle \gets }

\gets ctext[0..31]

P2

{\displaystyle \gets }

\gets ctext[32..63]

for(n = 2..9)

ctext

{\displaystyle \gets }

\gets Encrypt(ctext

{\displaystyle \oplus }

```
\oplus  salt[64(n-1)..64n-1]) //encrypt using the current key schedule and treat the salt as cyclic
        P2n-1)
    {\displaystyle \gets }
\gets  ctext[0..31]
        P2n
    {\displaystyle \gets }
\gets  ctext[32..63]
    for(i = 1..4)
        for(n = 0..127)
            ctext
    {\displaystyle \gets }
\gets  Encrypt(ctext
    {\displaystyle \oplus }
\oplus  salt[64(n-1)..64n-1]) //as above
            Si[2n]
    {\displaystyle \gets }
\gets  ctext[0..31]
            Si[2n+1]
    {\displaystyle \gets }
\gets  ctext[32..63]
    return state
```

Для обчислення хешу bcrypt обробляє вхідні дані еквівалентно шифрування 'eksblowfish (усиленний_ключ, input)':

```
bcrypt(cost, salt, key, input)
    state
    {\displaystyle \gets }
\gets  EksBlowfishSetup(cost, salt, key)
```

```
    ctext
    {\displaystyle \gets }
\gets  input
    repeat (64)
        ctext
    {\displaystyle \gets }
\gets  EncryptECB(state, ctext)
    return Concatenate(cost, salt, ctext)
```

У різних ОС (linux, OpenBSD), що використовують алгоритм bcrypt в стандартній функції crypt (3), як input подається константа «OrpheanBeholderScryDoubt»

bcrypt був розроблений в 1999 році і був захищений від ефективного перебору на апаратних засобах того часу. В даний час одержали широке поширення ПЛІС, в яких bcrypt реалізується ефективніше. У 2009 був створений алгоритм scrypt, що вимагає для своєї роботи значний обсяг пам'яті , об'єм пам'яті налаштовується.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Щоб WEB-сайт став доступний для відвідувачів, його потрібно розмістити на Internet сервері, підключеному до Мережі і дати йому доменне ім'я.

Дуже важливо, щоб WEB-сайт був розміщений на цілодобово функціонує потужному сервері з високою пропускною здатністю каналу.

Після запуску браузера введіть адресу нашої системи (якщо вона не налаштована системним адміністратором на автоматичне завантаження). У нашому випадку - http://localhost:3000/. У вікні ми бачимо початкову сторінку для авторизації в системі (рисунок 5.1). Після авторизації користувач потрапляє в систему, зовнішній вигляд якої залежить від призначених прав доступу, а також від ролі, наданої користувачеві під час його реєстрації в системі. Як було сказано раніше, система передбачає чотири рівні доступу до системи: адміністратор, керівник проекту, співробітник, бухгалтер.

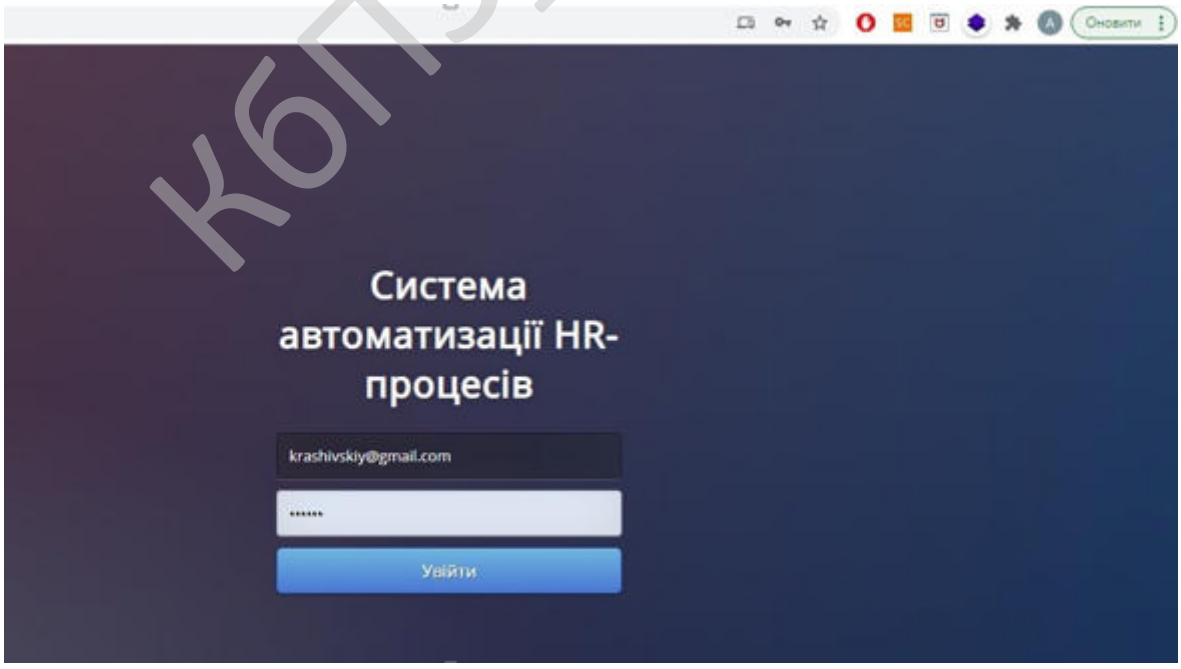


Рисунок 5.1 – Головна сторінка для входу в систему

На рисунку 5.2 зображено головну сторінку системи для адміністратора. У лівій частині сторінки знаходиться головне меню, яке використовується для виклику всіх функцій системи.

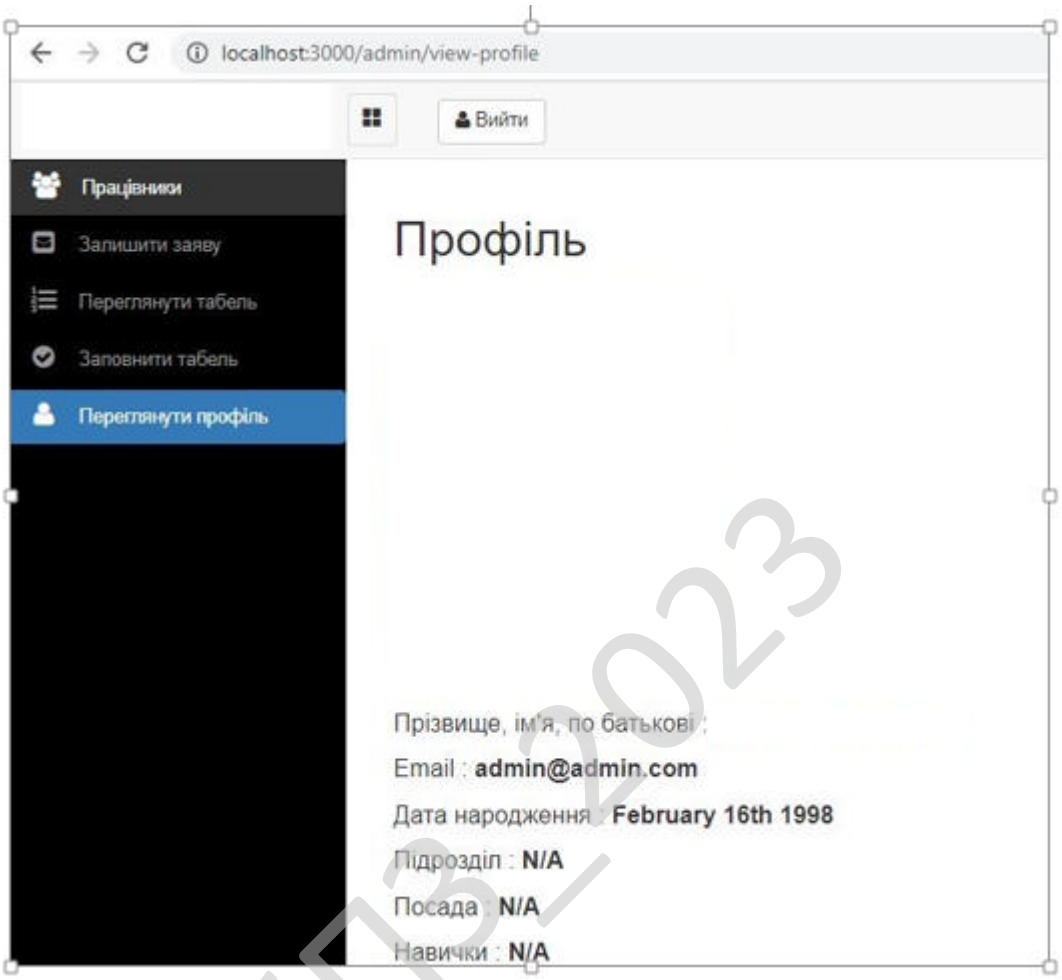


Рисунок 5.2 – Реалізація функціональності для адміністратора

Адміністратор має повний доступ до системи, що включає реєстрацію співробітників, прийняття рішення щодо привілеїв для інших співробітників, перегляд і зміну записів про робочий час, перегляд і зміну заробітної плати співробітників, видалення записів або профілів співробітників, призначення та перепризначення проектів кожному співробітнику, схвалення та відхилення заяви працівників про відсутність.

На рисунку 5.3 зображено форму реєстрації нового співробітника в системі. Після заповнення відповідних даних вся інформація буде зберігатися у відповідних документах розробленої бази даних MongoDB.

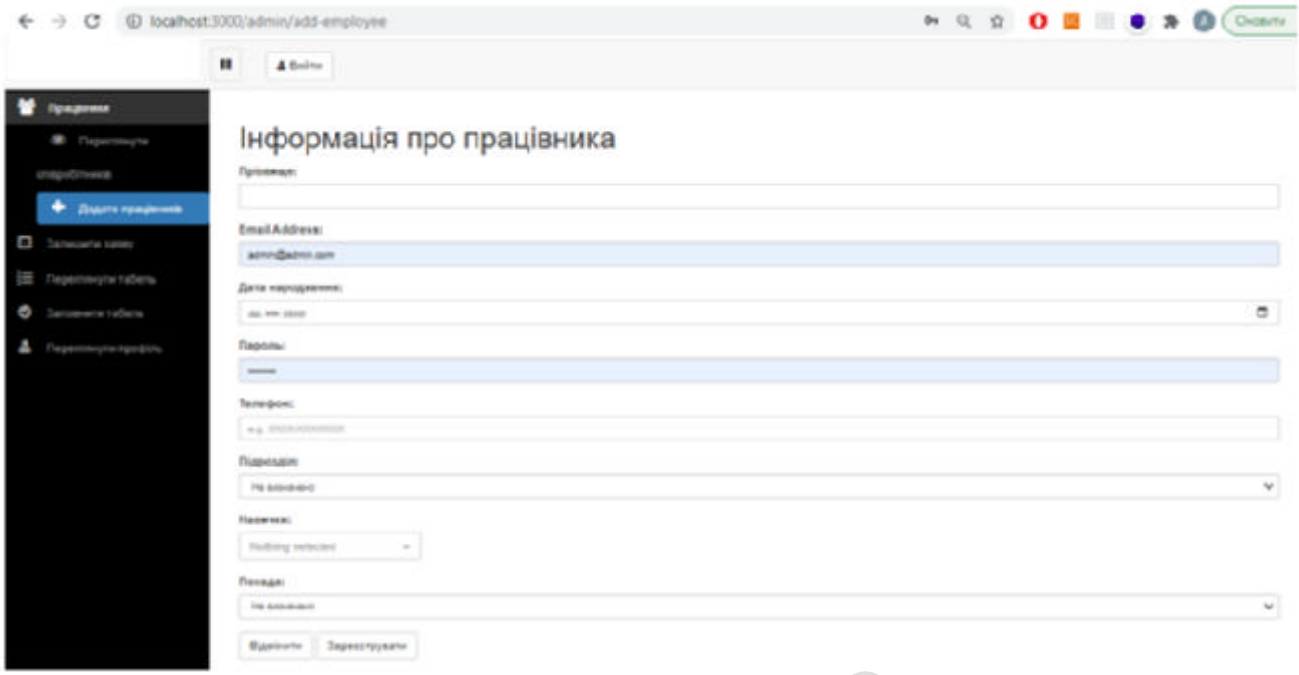


Рисунок 5.3 – Форма реєстрації нового працівника

Обов'язковою умовою успішного додавання співробітника є відмітка відповідних практичних навичок, які будуть використані на етапі реалізації конкретного проекту (рисунок 5.4).

Також адміністратор може бачити інформацію про всіх співробітників, які на даний момент зареєстровані в системі, їх контактну інформацію, посаду, підрозділ, проекти, до яких вони закріплені.

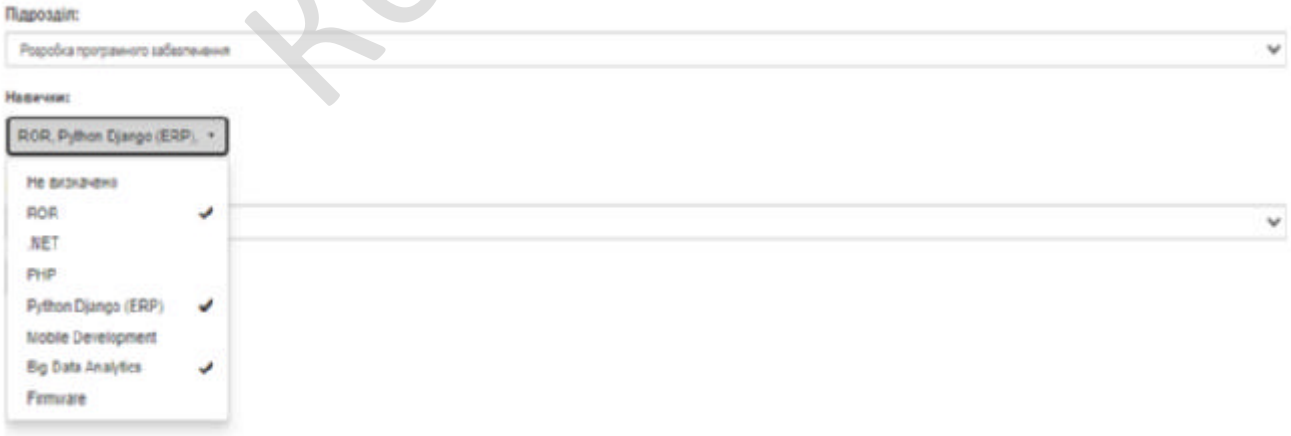


Рисунок 5.4 – Форма відзначення практичних навичок

На рисунку 5.5 представлено сторінку з відображенням інформації про працівників.

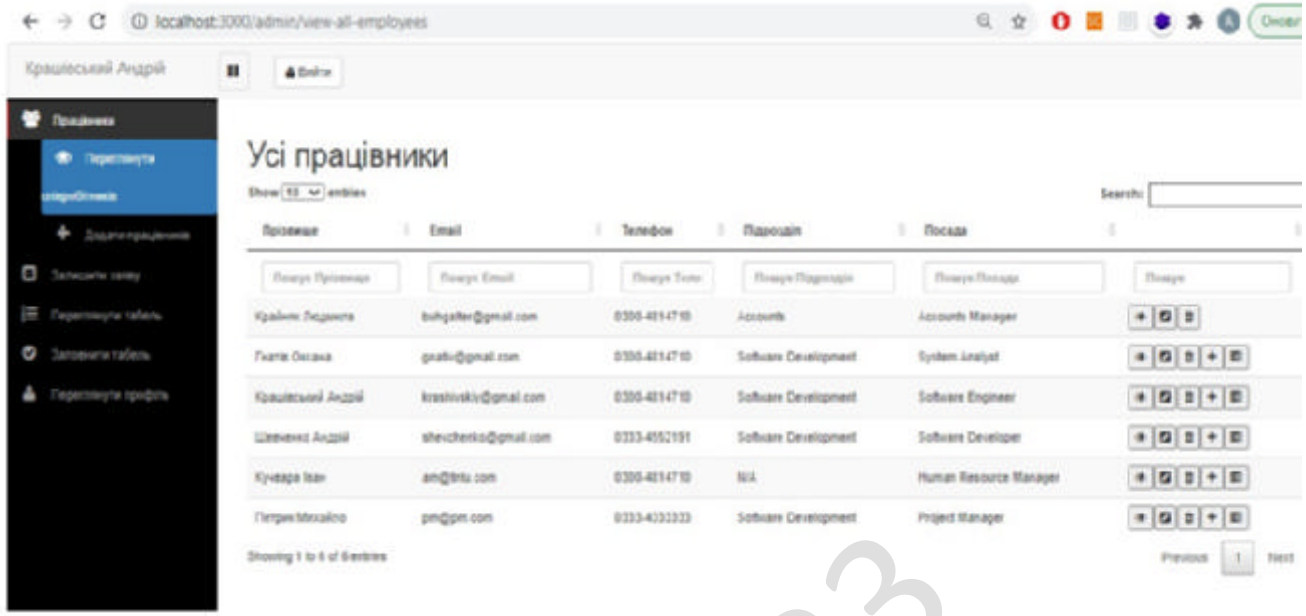


Рисунок 5.5 – Сторінка відображення інформації про працівників, які зареєстровані в системі

Також в ІС реалізована можливість перегляду профілю кожного працівника (рисунок 5.6).

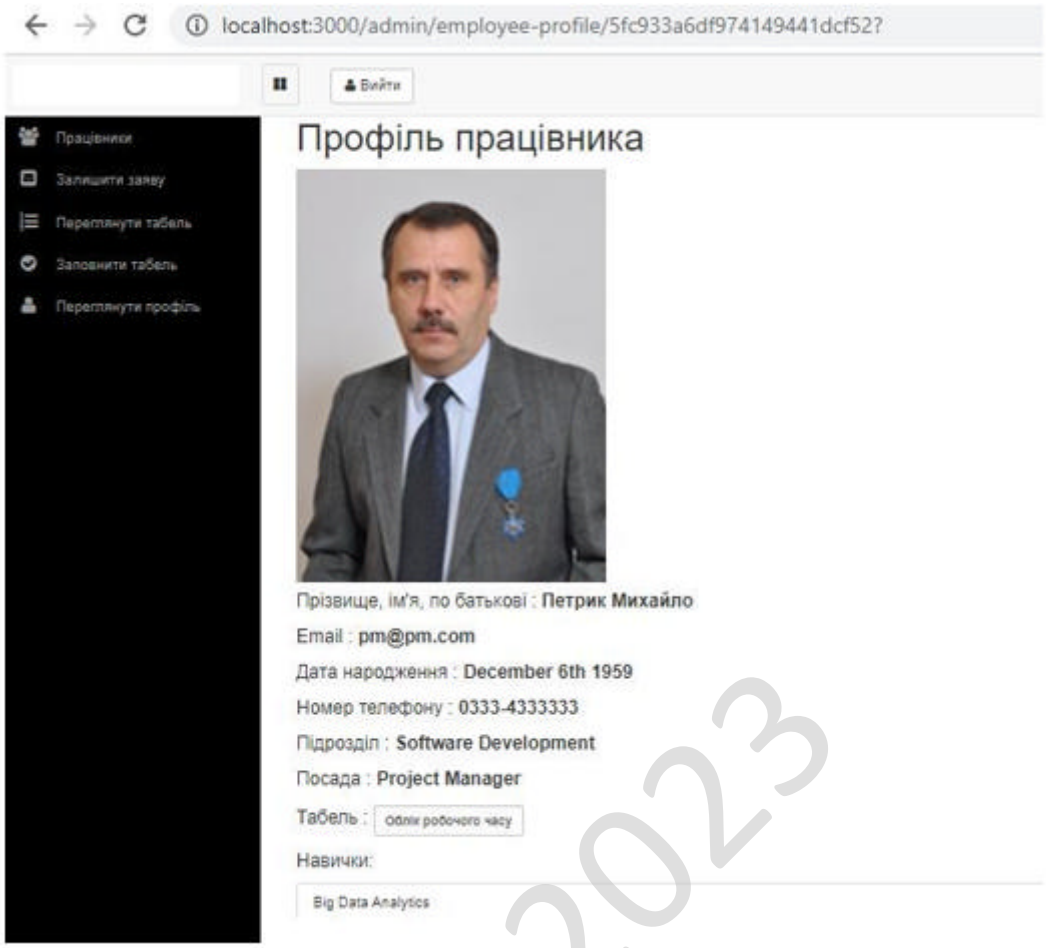


Рисунок 5.6 – Сторінка відображення інформації про працівника

Також в системі реалізовано функціонал, пов’язаний з підрахунком відпрацьованого часу, а також наданням щорічних відпусток, реєстр відповідних заяв. На рисунку 5.7 зображено форму подання працівником заяви про надання відпустки.

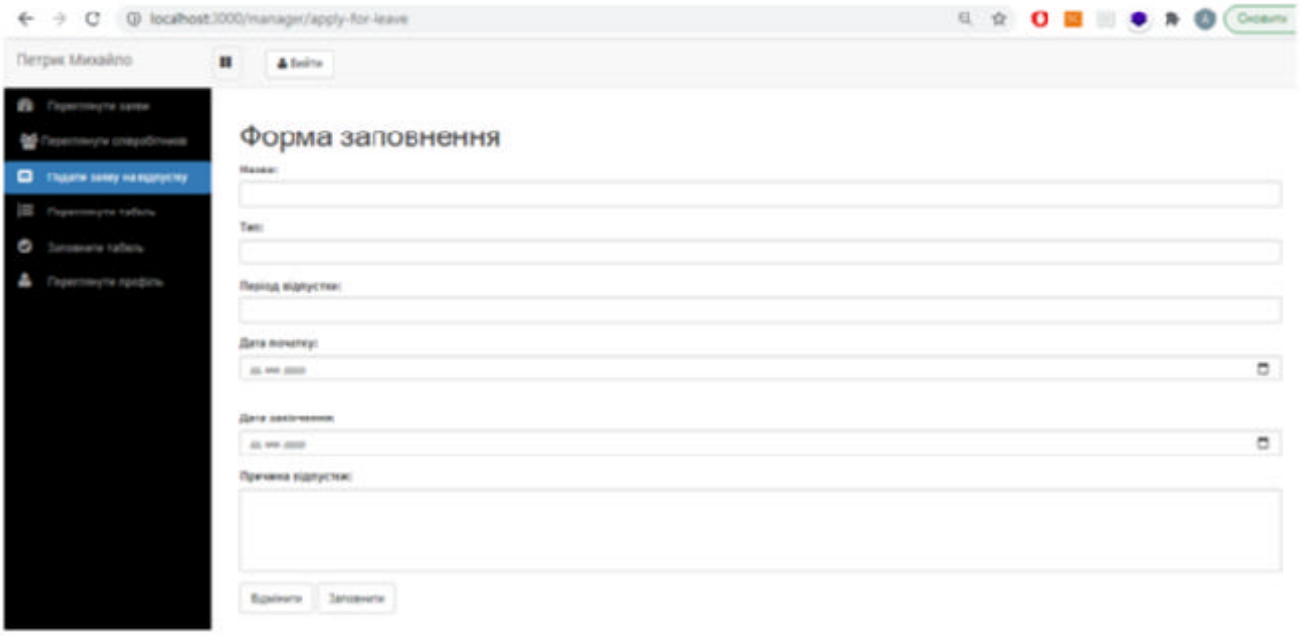


Рисунок 5.7 – Форма подачі заяви на відпустку

Керівник проекту в ІС може призначати працівників на відповідні проекти, а також здійснювати оцінку їх діяльності, формуючи відповідну систему якісної оцінки кожного працівника.

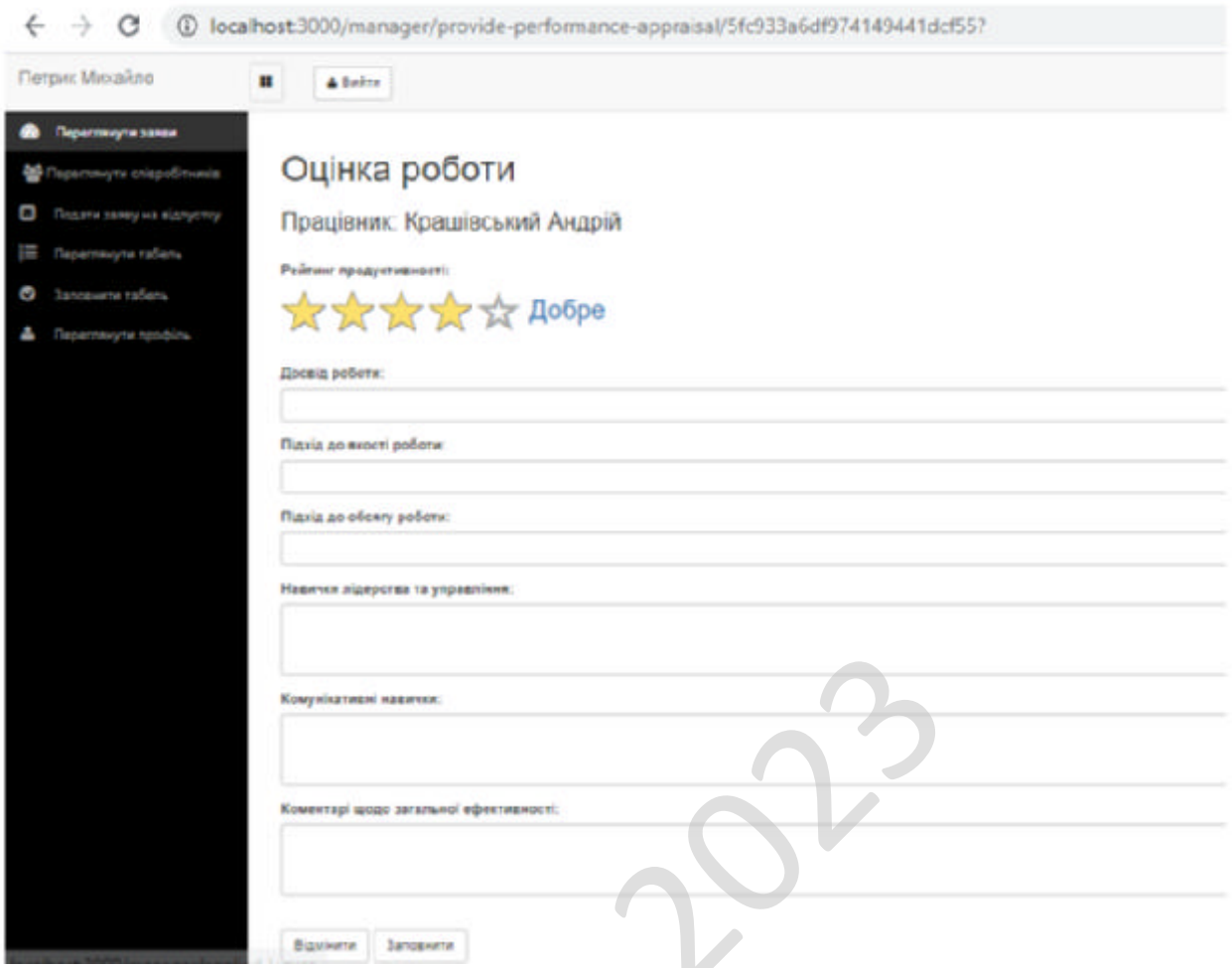


Рисунок 5.8 – Форма оцінки результативності роботи працівників

Особлива увага в системі приділяється питанням обліку відпрацьованого часу і заробітної плати. Кожен співробітник зможе заповнити свій графік, переглянути свою історію, побачити поточну зарплату, переглянути поточний профіль співробітника (включаючи освітню та робочу історію), переглянути всі проекти в організації, побачити інших співробітників, які спільно з ними працюють. , подати заяву на відпустку та переглянути статус відповідної заяви на відпустку.

На рисунку 5.9 зображено сторінку з даними про робочий час працівника за день.

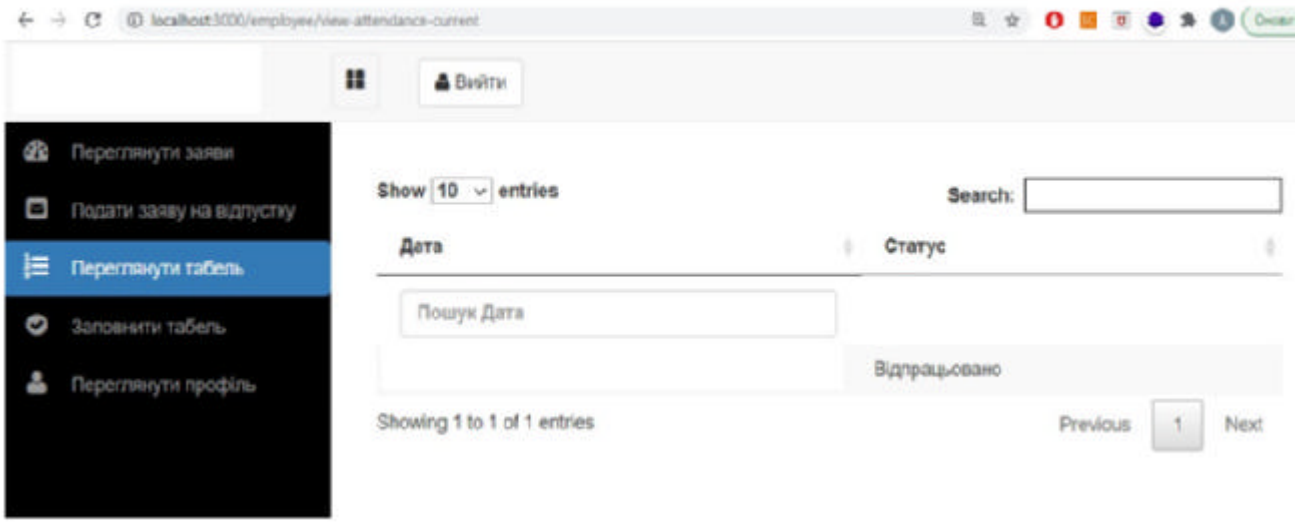


Рисунок 5.9 – Сторінка перегляду відпрацьованого часу працівником

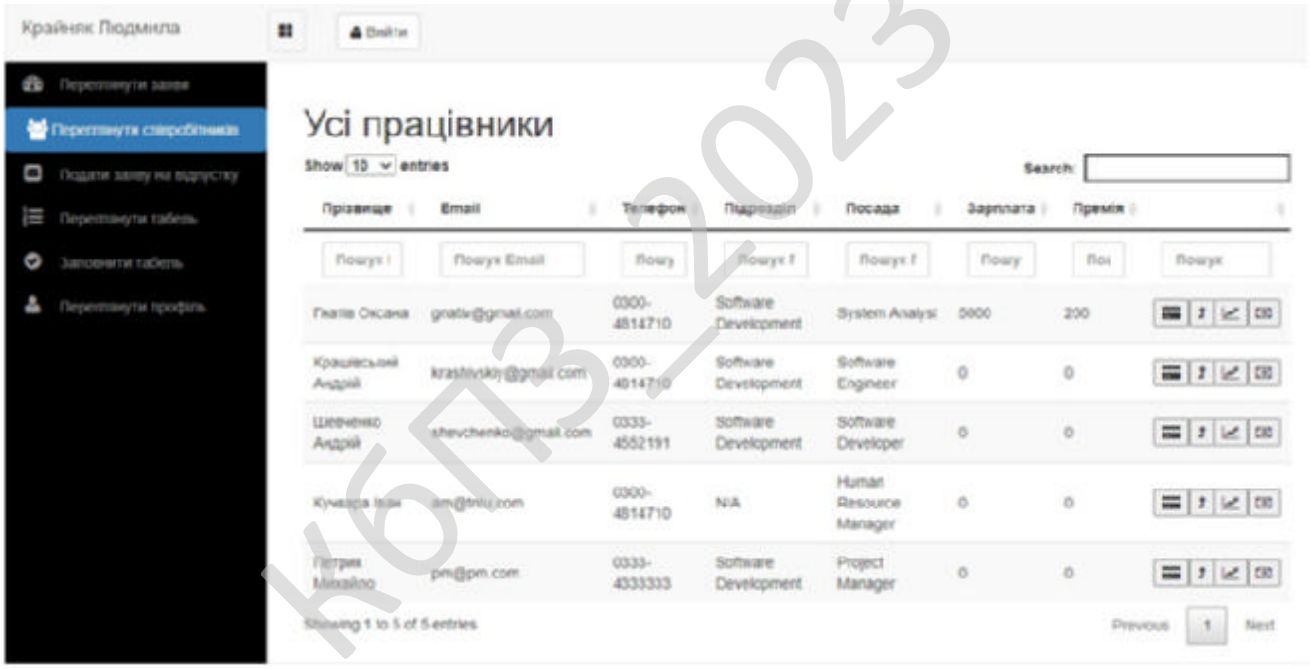


Рисунок 5.10 – Сторінка обліку оплати праці співробітників

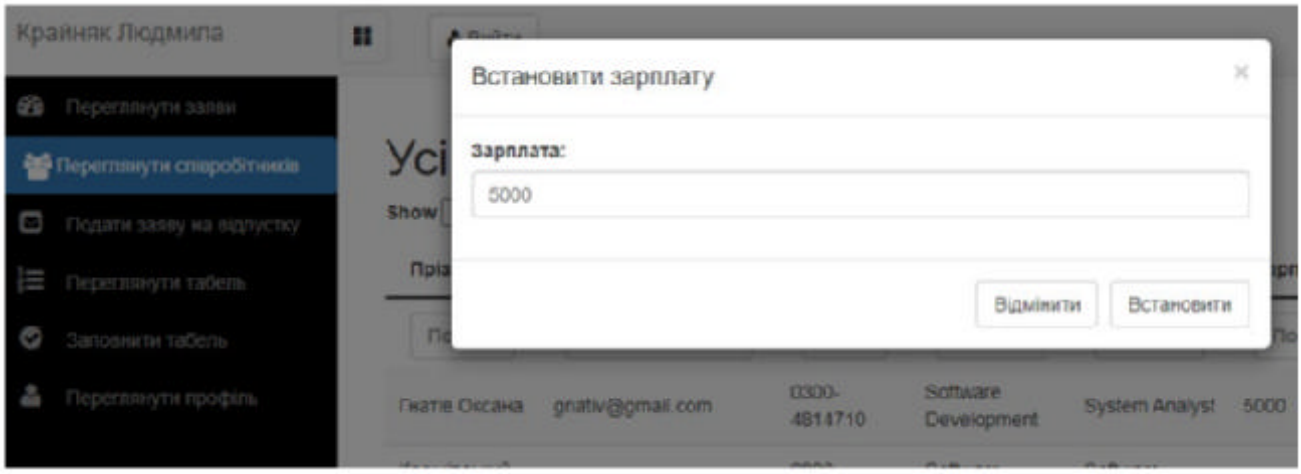


Рисунок 5.11 – Сторінка встановлення зарплати для окремого працівника

Система передбачає можливість створення бонусів для працівників за рахунок відповідних премій. Відповідно до пропозицій, поданих керівниками проекту, ці премії мають диференційований характер. На рисунку 5.12 наведено форму визначення премії працівнику.

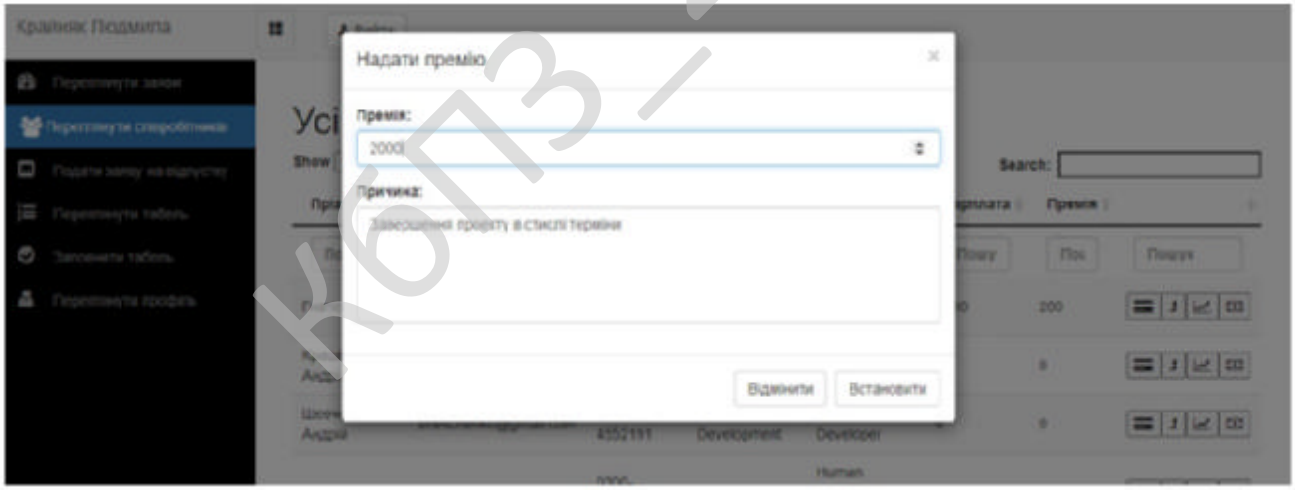


Рисунок 5.12 – Сторінка встановлення премії для окремого працівника

Виходячи з вищесказаного, можна підсумувати напрацьоване інформаційна система дозволяє:

- впровадження окремих облікових записів для всіх адміністраторів і співробітників;

- різні види даних і чіткий доступ до даних для учасників на основі їхніх привілеїв;

- реєстрація співробітників/адміністраторів;

- управління робочим часом усіх працівників та їх заробітком.

- ведення обліку навчального та трудового стажу працівника;

- управління поточними розподіленими проектами співробітників всередині організації.

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення, яке призначено для системи аналізу застосування NOSQL баз даних в сучасних ІС.

Метою розробки є дослідження та програмна реалізація системи аналізу застосування NoSQL баз даних в сучасних ІС, оцінці можливостей, переваг та обмежень цих технологій, а також виявленні їхнього впливу на якість та швидкість обробки даних в різноманітних сценаріях використання

Об'єктом дослідження є дослідження та програмна реалізація системи аналізу застосування NoSQL баз даних в сучасних ІС, оцінці можливостей, переваг та обмежень цих технологій, а також виявленні їхнього впливу на якість та швидкість обробки даних в різноманітних сценаріях використання.

Предметом дослідження є самі NoSQL бази даних та їхнє застосування в інформаційних системах. Методи та програмні засоби реалізації інформаційної системи.

Методи дослідження базуються на методах зберігання даних, методах математичної статистики, методах теорії масового обслуговування, методах розробки програмного забезпечення, теорії алгоритмів та об'єктно-орієнтованого проектування.

Наукова новизна отриманих результатів.

У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– удосконалено метод вибору NoSQL БД для роботи з сучасними ІС;

– розроблено інформаційну систему на основі NOSQL БД, яка має більш широкі можливості та швидкість доступу до даних на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація застосування NoSQL баз даних в інформаційній системі.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 - Початкові данні

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	40
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	_	2
29. Досвід роботи з програмними інструментами розробки (1-6)	_	3
30. Розробка ПО для декількох серверів одночасно (1-6)	_	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	_	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B , \tag{7.1}$$

де A - коефіцієнт Боема, A=2,45; Size - загальний об'єм відлагодженого програмного коду, тис. рядків; В - показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

де *Wi* - сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1,01+0,001(2,43+3,64+3,38+3,95+2,73) = 1,026$$

$$T_{ном} = 2,45 \cdot 2,7^{1,026} = 6,78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПV_j, \tag{7.3}$$

де *ПVj* - добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6,78 \cdot (0,88 \cdot 0,93 \cdot 0,88 \cdot 0,91 \cdot 0,95 \cdot 1 \cdot 1 \cdot 0,87 \cdot 1,22 \cdot 1,16 \cdot 1,1 \cdot 1,1 \cdot$$
$$\cdot 1,12 \cdot 1,1 \cdot 1 \cdot 1,1 \cdot 1,1) = 9,37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0,3 C T_{уточн}^{0,33+0,2(B-1,01)} S, \tag{7.4}$$

де С - визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S - коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0,3 \cdot 2,66 \cdot 9,37^{0,33+0,2(1,026-1,01)} \cdot 44 = 74 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 - Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	74	Ф 7.1-7.4
Впровадження	13	Д13
Всього	115	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де Fpq - плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{115 \cdot 1}{60\text{-}5} = 2 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 - Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	11	990	16,5
Монітор	60	11	660	11
Клавіатура	30	11	330	5,5
Маніпулятор «мишка»	30	11	330	5,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	3	90	1,5
Кабельні господарства ЛВС на 1 м. п.	2,5	420	1050	17,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	65,16

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{65,16 \cdot 3}{1,2} = 162,9 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=162,9/(60·8)=0,35 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів – електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 - Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців:

Таблиця 7.5 - Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	25000	75000
Продакт-менеджер	0,25	18000	13500
Інженер-програміст	2,0	25000	150000
Інженер-електронщик	0,35	18000	18900
Інженер-системотехнік	0,25	18000	13500
Адміністратор мережі	0,5	18000	27000
Системний програміст	0,25	18000	13500
Дизайнер WEB	0,25	18000	13500
Інженер-верстальник	0,25	18000	13500
Бухгалтер-економіст	0,5	18966	28449
Всього за період розробки	$R_{сп}$=5,6	-	$Ф_{роб}$=366849

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{366849}{5,6 \cdot 60} = 1092 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sy – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пектораль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми Компбест за 06.11.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core™ i3 10105 (BX8070110105) 1200, 4 ядра, 8 потоків, 3.7 GHz, 4.4 GHz, TDP - 65 Вт, 14nm, BOX	-
Системна плата	ASUS PRIME H510M-K сокет - 1200, DDR4, 3200 MHz, LAN - 1 Гбіт/с, D-Sub (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Intel UHD Graphics 630	-
Жорсткий диск	SSD M.2 2280 512GB LEVEN (JP600PCIE512GB) Серія - JP600, 512 GB, 3D TLC NAND, M.2, PCI Express 3.0 x4	-
Оперативна пам’ять	DDR4 8GB 3200 MHz Fury Beast Black Kingston Fury (ex.HyperX) (KF432C16BB/8)	-
Блок живлення	Gamemax 500W (GM-500B) ATX 12V v2.3, 500 Вт, 20+4 pin, CPU - 4+4pin, GPU - 1x6 pin, SATA - 3, Peripheral - 2, +12V1 - 20A, 1x120 мм, 150 x 140 x 86 мм	-
Корпус	Vinga CS210B, Miditower, ATX, Micro - ATX, Mini - ITX, Слотів расширення - 7	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO	220
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 - Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 1092 \cdot 115/40 = 3140 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 3140 \cdot 10 \cdot 0{,}01 = 314 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(3140+314) = 760 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 3140 \cdot 15 \cdot 0{,}01 = 471 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм $n_{міс}$ приймаємо 1/6 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки $N_м$ =3 міс:

$$З_{М1} = Ц_п \cdot N_м \, n_{міс}. \quad (7.16)$$

$$З_{М1} = 210 \cdot 3 \cdot 1/6 = 105 \text{ грн}.$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 10 примірників:

$$З_{М2} = \sum Ц_{д}.,\quad (7.17)$$

де Цд – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 49,2 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 49,2 грн/шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

Згідно виданих норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з}.,\quad (7.18)$$

де: $Ц_{з}$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_{М} = (105 + 492 + 1702)/40=57 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом (Нп = 15%) від основної зарплати виконавців

$$О_{п} = З_{о} \cdot Н_{п} \cdot 0{,}01,\quad (7.19)$$

де Нп - норматив витрат на освоєння нових мов програмування, %

$$О_{п} = 3140 \cdot 15 \cdot 0{,}01=471 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_{е}$ = 40 прим.)

$$А_{м} = \frac{А_{р} \cdot N_{міс}}{N_{е} \cdot 12},\quad (7.20)$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_{м} = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_{п} = З_{о} + З_{д} + С_{оц} + Г_{осп} + З_{м} + О_{п} + А_{м}.\quad (7.21)$$

$$С_{п} = 3140+314+760+471+57+471+876 = 6089 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 6089 = 3349 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	3140
2. Додаткова зарплата виконавців	$З_д$	314
3. Відрахування на соціальні потреби	$С_{оц}$	790
4. Загальногосподарські витрати	$Г_{осп}$	471
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	471
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	6119
9. Плановий прибуток	$П_р$	3349
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	9468
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1893,6
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	12817

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 - Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	12817
Всього капітальних витрат	–	12817

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 - Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на оплату праці	$З_р$	53690	33550
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	3204
Всього витрат за рік	$І$	53690	36754

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг - заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 400 годин на рік до 250 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 400 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 53680 \; грн.$$

до

$$З_{р\,нов} = 250 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 33550 \; грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

Після впровадження програмного продукту витрати на електроенергію не змінюються і складають Зел баз = Зел нов

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 - Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	12817	–	3204,25
Всього відрахувань	-	–	12817	–	3204,25

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (9468 - 6119) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 98925 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1596075}{(9468 - 6119) \cdot 40 \cdot 12 / 3} = 2,98 \text{ років}.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (53690 - 36754) - 0,25 \cdot 12817 = 13732 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{К_н - К_б}{І_б - І_н} \qquad (7.28)$$

$$T_{сп} = \frac{12817}{53690 - 36754} = 0,76 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 - Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	6119
3. Ціна розробленої програми	Грн.	9468
4. Плановий прибуток від реалізації розробленої програми	Грн.	3349
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	133960
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	98925
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,98
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9893
11.Величина економічного ефекту у користувача програмної продукції	Грн.	13732
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,76

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Аналіз умов праці програміста

Інтернет відіграє важливу роль у житті сучасної людини. Кожного дня мільйони людей використовують Інтернет для пошуку необхідної інформації, спілкуванні у соціальних мережах, перегляду новин. Багато людей користуються Інтернетом у професійних цілях, оскільки завдяки Інтернету з'явилося багато нових професій. Тому для веб-розробника так важливо розробити зручний інтерфейс для зручного сприйняття інформації, та необхідний функціонал, який буде відповідати необхідним вимогам та навантаженням. Все це вимагає багато часу та великого навантаження з боку розробників.

Тому так важливо слідкувати за умовами праці, в яких відбувається робочий процес. Оскільки захворювання можуть бути спричинені надмірним фізичним або розумовим навантаженням, через велику нервово-емоційну напругу, або через виробниче середовище. В даному розділі магістерської роботи проведемо аналіз основних чинників при роботі програміста.

При роботі за комп'ютером, розробник має велике зорове навантаження, тому йому необхідне належне освітлення приміщення. Якщо в приміщенні недостатньо природного освітлення, потрібно використовувати спеціальні світильники. Також оскільки розробник значний час працює з електричними приборами є можливість, бути ураженим електричним струмом, тому потрібно дотримуватись всіх необхідних норм. Серед основних чиників, які впливають на розробників під час трудової діяльності можна виділити[46]:

1. Рівень освітлення в приміщенні.
2. Температура, вологість в приміщенні.
3. Рівень шуму на робочому місці.

4 .Напруга в електричному ланцюзі, електричні показники.

Розберемо кожний з чиників окремо, проаналізуємо які повинні бути стандарти кожного з чиників, відповідно до правил з охорони праці.

Рівень штучного освітлення

Головним документом для встановлення норм необхідних показників освітлення є ДБН В.2.5-28:2018 «Природне та штучне освітлення» [47].

Сьогодні найбільш розповсюдженішими є світлодіодні ламп. В середньому світловіддача від таких ламп знаходиться на рівні 80-120 Лм/Вт [48]. Джерелом живлення прийнято вважати електричну мережу у 220В. А освітленіть робочого приміщення повина бути Е =300-500 Лк, оскільки робота програміста відноситься до робіт середньої точності з присвоєнням розряду зорових робіт IV[49].

Рівень сітла повинен бути достатнім, щоб працівник міг працювати без навантаження на зір. Це залежить від системи освітлення, кількості світильниув, їх типу та розміщення у приміщенні. Допустиме значення освітленості робочої поверхні приймається Е = 400 лк [50].

Для покращення освітлення комп'ютерній лабораторії будуть використовуватися світлодіодні лампи, а саме FL-LED T8-900 світловий потік яких F=1500лм.

Мікроклімат робочої зони: температура, відносна вологості, швидкість руху повітря

Головним документом для встановлення норм мікроклімату робочої зони є ДСН 3.3. 6.042 -99 «Державні санітарні норми мікроклімату виробничих приміщень» [51].

Праця програміста за важкістю відноситься до легкої фізичної роботи категорії Іа [49]. Де вказано, що в приміщенні, я кому знаходиться компютерне обладнення, повнині бути встановлені певні норми, оскільки офісна техніка є джерелом тепловиділень, що може спричинити підвищення температури. Серед

потимальних параметрів для роботи встановлена температура 23 – 25^{0}С і вологість на рівні 40 – 60% в залежності ві періоду року.

Для підтримки комфортної температури можна використовувати як організаційні методи, наприклад розпорядок дня, так і технічне обладнення, наприклад кондиціонери, вентіляцію. Як правило в холодний період часу використовуються додаткове опалення для підтримання комфортної температури, а в літку встановлюються кондеціонери.

Рівень шуму на робочому місці

Як правило при використанні великої кількості компютерів в одному приміщенні, через гудіння, рівень шуму має значення більше норми. Допустима норма становить менше 50 дБ [52].

Гучний шум негативно впливає на умови праці та організм людини. Якщо шум триває тривалий час цу може спричинии головні болі, біль у вухах, підвищення стомлюваності, зниження концентрації та уваги. Такі симптоми можуть викликати стресові ситуації у людини. Все це шкодить продуктивності працініка та його стану здоровья.

Щоб встановити необхідний рівень шуму, використовують додаткову звукоізоляцію. Для цього найчатіше використовуються мати та плити із скляного та мінерального волокна, м'які плити з деревних стружок, картон, гуму, утеплений лінолеум, а також заміна вікон на звукоізолюючі.

8.2 Заходи профілактики при роботі з комп’ютерною технікою

Санітарно-гігієнічні норми є важливим критерієм при роботі в приміщенні. Від них залежить здоровє працівників, їх рівнень працездатності, втомлюваність. Щоб всього цього уникнути потрібно стежити за нормами на робочому місці.

Якщо говорити про електробезпеку в приміщенні, то в приміщенні необхідно устаткування розподільних щитів спеціальними розетками з

заземлюючими контактами, повині бути заземлені всі прилади і пристрої, час від часу повинна проводитися перевірка всіх приладів, щорічна здача іспитів з охорони праці.

Для оптимальних показників мікроклімату та освітленості потрібні використовувати дефлектор, для організації вентиляції, та повітрообміну. Та перевірку освітленості в приміщенні згідно відділом охорони праці, щоб відповідати нормам для зорової роботи .

Ще однією проблемою з якою часто зустрічаються програмісти є мала рухливість та повний сидячий робочий день. Тому рекомендується час від часу робити невеликі перерви, під час обіднього перериву вживати їжу не на робочому місці. У окремих випадках, коли при дотриманні всіх санітарних норм, працівник все одно себе погано почуває, дозволяється індивідуальний підхід для обмеження роботи з обчислювальними пристроями. Тривалість роботи за комп'ютером не повинна безперервно тривати більше 4 годин.

Для зменшення зорового та нервово-емоційного навантаження, та поліпшення мозкової діяльності рекомендується робити перерви для психологічного та фізичного розвантаження.

В приміщеннях також подині бути протипожежне обладнання, та інструкція у разі надзвичайних ситуаціях. Повина бути особа, яка відповідає за пожежну безпеку, перевіряє обладнання, та системи протипожежного захисту а також щорічне проведення інструктажів серед працівників.

Автоматична пожежна сигналізація повинна відповідати вимогам ДБН ДБН В.2.5-56:2014, яке вимагає використання вогнестійких кабелів та автоматичну роботу системи оповіщення та евакуації людей у випадку надзвичайної ситуації [53].

При перевірці, приміщення повинно відповідати всім нормам пожежної безпеки. Це виконується за допомогою перевірки пожежної охорони та техніки, проведенню інструктажу і своєчасне інформування пожежної охорони про несправність пожежної техніки, впровадження систем протипожежного

захисту. Організаційні та технічні заходи, спрямовані на попередження виникнення пожежі, обмеження поширення вогню та успішної евакуації людей.

8.3 Розрахнок занулення глухозаземленої нейтралі

Занулення, як основний засіб захисту, застосовується в електроустановках до 1 кВ з глухозаземленою нейтраллю трансформатора або генератора [54]. Початкові дані для розрахунку занулення глухозаземленої нейтралі трансформатора виробничого приміщення:

Загальна потужність: P = 5 кВт.

Кількість електродвигунів: m = 10

Потужність освітлювальних приладів: Po = 3 кВт.

Довжина магістрального кабеля: LM =70 м.

Довжина розгалудження: l=22м.

Лінійна напруга U=380 В.

Фазна напруга Uф=220 В.

Визначаємо силу номінального струму електроустановки:

$$Iном = Imax = (P * 1000) / (\sqrt{3} * Uл * cosф) \quad (8.1)$$

$$Iном = Imax = (5 * 1000)/(\sqrt{3} * 380 * 0,85) = 8,9 А$$

де: P − номінальна сумарна потужність електроприладів, кВт;

Uл − лінійна напруга, В;

cosф − коефіцієнт потужності, приймається в залежності від типу електрообладнання в межах 0,8..0,87.

Визначаємо силу пускового струму електродвигуна:

$$Iпус = 5 * Iном \quad (8.2)$$

$$Іпус = 5 * 8,9 = 44,5 А$$

Визначаємо номінальну силу струму апарата захисту:

$$Ін = Іпус/b \quad (8.3)$$

$$Ін = 44,5 /2,5 = 17,8 А$$

b − коефіцієнт пуску електродвигуна − для легких умов пуску − 2,5..3.

Вибираємо запобіжник ПН 2-100 з плавкою вставкою Іном = 50 А. Визначаємо найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання:

$$Іктіп = Ін * К \quad (8.4)$$

$$Іктіп = 50 * 3 = 150 А$$

Ін−номінальний струм апарата захисту;

К − коефіцієнт надійності;

Знаходимо переріз провода або кабеля розгалуження з умови допустимого нагрівання:

$$Ідоп = Івст/а \quad (8.5)$$

$$Ідоп = 50 / 3 = 16,6 А$$

Вибираємо площу перерізу 10 мм^2 (Sф) при числі проводів і = 4 розташований у повітрі. Визначаємо максимальний робочий струм:

$$Iроб = Ко(Кз*(Р*1000)/(\sqrt{3}*Uл*cosф)*m+Кз*(Ро*1000)/(\sqrt{3}*Uл*cosф)) \quad (8.6)$$

$$Iроб = 0,75*(0,85*((5*1000)/(1,73*380*0,85)*10+(3*1000)/(1,73*380*0,85))=60,4 \text{ А}$$

Ко − коефіцієнт одночасності роботи групи електроприймачів;

Ко = 0.7...0.8; Кз = 0.8...0.9;

Кз – коефіцієнт завантажених електродвигунів;

Ро =3кВт − потужність освітлювальної мережі;

Визначається струм короткочасного перевантаження магістрального кабеля:

$$Iпер = Ко(Кз*(Р*1000)/(\sqrt{3}*Uл*cosф)*n+Кз*(Ро*1000)/(\sqrt{3}*Uл*cosф))+Iпус \quad (8.7)$$

$$Iпер = 0,75*(0,85 *(5*1000)/(1,73*380*0,85)*9+0,85*(30*1000)/(1,73*380 * 0,85)) + 44,5 = 130,0643 \text{ А}$$

Струм спрацювання електромагнітного розчеплювача додатково перевіряемо по максимальному струму перевантаження лінії:

$$Iспр >= 1,25 * Iпер \quad (8.7)$$

$$Iспр >=1,25 * 130,0643 = 162,5803 \text{ А}$$

Приймаємо Iспр = 162,5803 А. Вимикач : А3714Б.

Вибираемо площу перерізу Sф магістрального кабеля (провідника) по Iдоп. Sф = 70mm^2 , − кабель АВРГ прокладений в землі, i=3 (число проводів). Вибрану площу перерізу перевіряємо для автоматів з електромагнітним розчеплювачем:

$$Iдоп >= Iспр/4,5 \quad (8.8)$$

$$Iдоп >= 162/4,5 = 36 А$$

Проводимо узгодження з номінальним струмом автомата:

$$Iдоп = Iспр/3 \quad (8.9)$$

$$Iдоп = 162/3 = 54 А$$

Значення 36 і 54 А. менше ніж Imax= 60,4 А., значить площа перерізу кабеля вибрана вірно. Визначаємо потужність трансформатора:

$$Nтр = ((Кп * Рном)/cosф) \quad (8.10)$$

$$Nтр = (0,7 * 53)/0,8 = 46,375 кВ*А$$

Рном − сумарна потужність електроприймачів, кВт;

Cosф − середній коефіцієнт потужності електроприймачів (0,8);

Кп − коефіцієнт попиту (0,7);

Одержане значення потужності трансформатора округляємо до ближчого стандартного значення. Визначаємо опір трансформатора Z_T. Вибираємо трансформатор на 40 кВА (Z_T =0,562 Ом). визначаємо орієнтовно площу перерізу провідника. Для магістрального кабеля:

$$Sn_1 >= 0,5 * Sф \quad (8.11)$$

$$Sn_1 >= 0,5 * Sф = 0,5*70=35 мм^2$$

Визначаємо для розгалуження:

$$Sn_2 >= 0{,}5*10 = 5 \text{ мм}^2$$

Округляємо ці значення до найближчих більших 35 мм^2 (Sn1), і 6 мм^2 (Sn2). Визначаємо активний і індуктивний опір фазного і нульового захисного провідників на дільницях 1 і 2:

$$Rф = p * (Lм/Sф_1) + p * (L/Sф_2) \qquad (8.12)$$

$$Rф = 0{,}028 * (70/70) + 0{,}028 * (22/10) = 0{,}0896 \text{ Ом}$$

$$Rn = p * (Lm/Sn_1) + p * (L/Sn_2) \qquad (8.13)$$

$$Rn = 0{,}028 * (70/35) + 0{,}028 * (22/6) = 0{,}1586 \text{ Ом}$$

Для окремо проложених нульових провідників иого приимають рівним 0,6 Ом/км. При прокладці кабелем, або в сталєвих трубах індуктивним опором нехтують.

Знаходимо дійсне значення (модуль) струма однофазного короткого замикання:

$$Iкр = Uф/((Zт/3) + \sqrt{((Rф + Rn)^2 + (Xф + Xn + X'n^2))} \qquad (8.14)$$

$$Iкр = 220/((0{,}562/3) + \sqrt{(0{,}0896 + 0{,}1586)^2}) = 418{,}18 \text{ А}$$

Визначення максимальноі напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$Uктах = Iкр * Zн \quad (8.15)$$

$$Uкmax = 418,18 * 0,1586 = 66,32 В$$

$Z_Н$ − повний опір нульового провідника.

Умова не виконується. Необхіно збільшити перерізи Sn 1 та Sn 2 до Sф1 та Sф2 і зробити перерахунок:

$$Rn = 0,028 * (70/70) + 0,028 * (22/10) = 0,0896 \text{ Ом},$$

$$Iкр = 220/((0,562/3) + \sqrt{(0,0896 + 0,0896)^2}) = 600,21 А,$$

$$Uкmax = 600,21 * 0,0896 = 53,77 В.$$

Умова не виконується, необхідно або замінити запобіжник з плавкою вставкою на автоматичний вимикач із струмовим реле, що дає можливість зменшити час замикання на копус і підвищити допустиму напругу на корпусі або застосувати повторне заземлення нульового захисного провідника. Повторне заземлення нульового захисного провідника:

$$Rn=(Uдоп * Ro)/((Iкр * Zн)−Uдоп) \quad (8.15)$$

$$Rn= 36 * 4/((600,21 * 0,0896) - 36) = 8,09 Ом.$$

8.4 Висновки

Існує багато факторів, які можуть вплинути на роботу розробника, через некомфортні або навіть небезпечні умови праці, які знаходяться в приміщенні. Серед основних причин виділяється: недостатній рівень світла, гучний шум, високий рівень навантаження, умови мікроклімату. Під час дослідження теми,

були переглянуті можливі шкідливі та небезпечні ситуації, які можуть виникнути на робочому місці та спообби їх уникнення та ліквідації.

В результаті можна зробити висновки, які умови повині бути для продуктивної роботи працівника, як повино бути організоване приміщення і його робоче місце. За допомогою проведених обчислень, можна становити необхідні для комфортної роботи умови.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання магістерської роботи, призначено для порівняння та виявлення кращих технологій відповідно до поставлених сучасних вимог у галуз веб-розробки, а саме застосування різних типів БД при розробці додатків.

У магістерській роботі представлені дані з яких можна зробити висновки, які технології сьогодні використовуються, у яких сферах, та для його функціоналу призначені бази даних різних типів. Які можливості та для яких проектів краще використовувати класичні реляційні а коли ООБД.

Для отримання результатів були проведені наступні дослідження:

– Було створено односторінковий додаток засобами NodeJS.

– Було розробленно та протестовано функціонал REST API та веб-сокети за допомогою AngularJS.

– Було створенно сервер засобами NODEJS, з підтримкою бази даних MongoDB.

– Було створено шаблони, які можна надалі використовувати при створенні нових додатків.

Розроблені під час виконання магістерської роботи додатки дозволяють чітко зрозуміти для яких проектів краще підходять реляційні а коли NOSQL бази даних, які їх переваги та недоліки.

Розроблені додатки підтримують функціонал, який використовується на більшості сайтів в Інтернеті, серед основних можливостей це додавання медіа файлів, можливість надсилання e-mail повідомлень, використання сокетів для спілкування в чаті, авторизація, динамічний інтерфейс. За допомогою фреймворків була продемонстрована технологія односторінкових додатків, яка надала можливість розробити швидкий та зручний для сприйняття інтерфейс.

При розробці використовувалися мови JavaScript, та TypeScript що дало можливість порівняти на скільки сильно відрізняються фреймворки Angular та AngularJS та їх основні концепції. З цього можна зробити висновки скільки займає час розробки, надійність та можливість подальшого підтримання продукту.

В результаті розроблені додатки відповідають поставленим вимогам технічного завдання, та мають можливість на подальше вдосконалення. Створені додатки можуть бути впровадженні у виробництво.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 13732 грн. Враховуючи вартість розробки та необхідне обладнання, строк окуплення становить 0,76 років.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бахрушин В. Є. Методи аналізу даних: Навч. посібник / В. Є. Бахрушин. – Запоріжжя: КПУ, 2011. – 268 с

2. Шумейко А. А. Интеллектуальный анализ данных (Введение в Data Mining) / А. А. Шумейко, С. Л. Сотник. – Днепропетровск: Белая Е. А., 2015. – 212 с.

3. http://www.nbuv.gov.ua/eb/ep.html - Національна бібліотека України імені В.І.Вернадського

4. Node.js v14.0.0 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/api/

5. М. Кантелон , М. Хартер, Т. Головайчук, Н. Райлих // “Node.js в действии”.

6. Янг А., Мек Б., Кантелон М. // “Node.js в действии. 2-е издание”.

7. John Resig, Bear Bibeault, Josip Maras // “Secrets of theJavaScript Ninja”.

8. Express [Електронний ресурс] – Режим доступу до ресурсу: http://expressjs.com/

9. Фреймворк AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS

10. AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://angularjs.org/

11. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/

12. React.js [Електронний ресурс] – Режим доступу до ресурсу: https://https://uk.reactjs.org/

13. AngularJS MVC [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

14. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/

15. Angular 2 [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/

16. Односторінковий застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Односторінковий_застосунок

18. Build Node.js Apps [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

19. REST [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/REST

20. What is REST [Електронний ресурс] – Режим доступу до ресурсу: https://restfulapi.net/

22.Веб приложение [Електронний ресурс] – Режим доступу до ресурсу: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/

23.Мова розмітки гіпертексту [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML

24. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/HTML

25. Каскадні таблиці стилів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS

26. Стек MEAN [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MEAN_(веброзробка)

27. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

28. Веб-технології для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web

29. CORS, XSS [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/maleta/cors-xss-and-csrf-with-examples-in-10-minutes-35k3

30. AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AJAX

31. Fetch API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API

32. AngularJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/angular/default.asp

33. jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JQuery

34. Основи Web-технологій [Електронний ресурс] – Режим доступу до ресурсу: https://pidruchniki.com/1243020547796/informatika/web-tehnologiyi_pidpriyemstvah

35. npm [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/

36. Install MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/administration/install-on-linux/

37. Как установить Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/node-js-ubuntu-18-04-ru

38. Linux [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linux

39. npm-audit [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/cli/audit

40. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

41. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

42. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/products/compass

43. Postman [Електронний ресурс] – Режим доступу до ресурсу: https://www.postman.com/

44. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

45. SPA (Single-page application) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

46. Охорона праці [Електронний ресурс]: реферат – Режим доступу до ресурсу: https://revolution.allbest.ru/life/00468031_0.html

47. Природне і штучне освітлення ДБН В.2.5-28:2018: державні будівельні норми України [Електронний ресурс] / Ю. Громадський, С. Облакевич, М.Громадський, Г. Фаренюк, Є. Фаренюк, О. Підгорний, О. Сергейчук, Є.Рейцен, В. Єгорченков, Л. Коваль, Д. Радомцев, В. Злоба, Н. Кучеренко, Г.Кожушко, О. Гончар, О. Козенко, Б. Шабашкевіч, Ю. Добровольський, В.Акіменко, С. Гозак, А. Яригін, В. Назаренко, В. Мартиросова, В. Сорокін, Є.Пугачов - Київ 2018 - Режим доступу до ресурсу: https://ledeffect.com.ua/images/__branding/dbn2018.pdf

48. Розрахунок світлодіодного освітлення кімнати [Електронний ресурс] – Режим доступу до ресурсу: https://luxled.biz.ua/rozrahynok-svitlodoidnogo-osvitlennja-kimnatu-v-kvarturi-abo-bydunky

49. Охорона праці, охорона праці та безпека в надзвичайних ситуаціях : метод. вказ. до викон. розділів у дипломних роботах / [укл. В.М. Челябієва, О.Л. Гуменюк] - Чернігів ЧДТУ 2013 - [Електронний ресурс] – Режим доступу до ресурсу: http://ir.stu.cn.ua/bitstream/handle/123456789/12461/Охорона праці та безпека. в надзв. ситуац;метод.вказ..pdf?sequence=1&isAllowed=y

50. Освітленість робочих місць [Електронний ресурс] – Режим доступу до ресурсу:https://ua-referat.com/Освітленість_робочих_місць_сучасні_підходи_до_вимірів_і_оцінки

51. Температурний режим праці [Електронний ресурс] – Режим доступу до ресурсу: http://poltava.medprof.org.ua/poltava/zakhist-trudovikh-ta-socialno-ekonomichnikh-prav-pracivnikiv-galuzi/pravova-dopomoga/temperaturnii-rezhim-praci-jakim-vin-maje-buti/

52. Шум. Методи захисту від його дії : метод. вказ. до лабораторної роботи / [укл. В. І. Шмирко, С. М. Журавель] — Запоріжжя: ЗНТУ, 2014. - 14 с.

[Електронний ресурс] – Режим доступу до ресурсу: https://zp.edu.ua/sites/default/files/konf/oop_shum-2014.pdf

53. Системи протипожежного захисту ДБН В.2.5-56:2014 / Б. Платкевич, В. Носач, В. Федюк, В. Мусійчук, В. Євстіфєєв, Г. Дубінський, В. Сокол, А.Бушиленко, В. Дунюшкін, Р. Уханський, С. Пономарьов, В. Приймаченко, А.Приймаченко, С. Пітайчук, Н. Морозова, І. Колосов, О. Лагода, П. Мізін, В.Савченко, М. Федорович, П. Шаповалов, Л. Фесенко — Київ 2015 — [Електронний ресурс] – Режим доступу до ресурсу: http://kbu.org.ua/assets/app/documents/dbn2/98.1. ДБН В.2.5-56~2014. Системи протипожежного захисту.pdf

54. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ IBM−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с.[Електронний ресурс] – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

55. The Top JavaScript Frameworks [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/

56. JavaScript Frameworks 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/migrate-mongo

57. Web Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/web-technology

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку та програмну реалізацію додатку з використанням NoSQL баз даних.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 35-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою розробки є дослідження та програмна реалізація додатку з використанням NoSQL баз даних. В результаті роботи розглянуто методи управління реплікаціями в NoSQL системах, розроблено програмну реалізацію системи автоматизації HR-процесів, яка представляє собою веб-додаток з використанням Node.js та MongoDB, Node Express, який є веб-додатком для Node.js, а інтерфейс створений за допомогою HTML, Bootstrap, CSS та JS

4 Джерела розробки

Джерелом цієї магістерської роботи є відносна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– Розробку додатку;

– систему підключення та тестування баз даних ;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10/11.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище NodeJS та СУБД MongoDB

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці та техніки безпеки в магістерській роботі повинен бути розглянутий аналіз умов праці програміста та розрахунок захисного заземлення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 107 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської дипломної роботи. Постановка задачі на виконання бакалаврської дипломної роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень бакалаврської дипломної роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

11.1 Подання магістерської дипломної роботи на попередній захист 10.12.2023 р.

1.2 Подання магістерської роботи на захист 10.01.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Босько В.В.

Дослідження та програмна реалізація застосування NoSQL баз даних в інформаційній системі

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 17

Літера: РП

Кропивницький – 2023

```
var express = require('express'); var router = express.Router();
var passport = require('passport'); var User = require('../models/user');
var Project = require('../models/project'); var csrf = require('csurf');
var csrfProtection = csrf();
var config_passport = require('../config/passport.js'); var moment =
require('moment');
var Leave = require('../models/leave');
var Attendance = require('../models/attendance');



router.use('/', isLoggedIn, function isAuthenticated(req, res, next)
{
next();
});
/**
*     Description:
*     Displays home page to the admin
*
*
*     Known Bugs: None
*/
router.get('/', function viewHome(req, res, next) {
res.render('Admin/adminHome', {
title: 'Admin Home', csrfToken: req.csrfToken(),
userName: req.session.user.name
});
});
/**
*     Description:
*     First it gets attributes of the logged in admin from the User Schema.
*     Attributes are get with the help of id of logged in admin stored in
session.
*/
router.get('/view-profile', function viewProfile(req, res, next) {
User.findById(req.session.user._id, function getUser(err, user)
{
if (err) {
console.log(err);
```

```
}
res.render('Admin/viewProfile', { title: 'Profile',
csrfToken: req.csrfToken(), employee: user,
moment: moment,
userName: req.session.user.name
});
});

});
/**
*     Description:
*     Sorts the list of employees in User Schema.

*     Known Bugs: None
*/

router.get('/view-all-employees', function viewAllEmployees(req, res, next) {
var userChunks = []; var chunkSize = 3;
//find is asynchronous function
User.find({$or: [{type: 'employee'}, {type: 'project_manager'},
{type: 'accounts_manager'}]}).sort({_id: -1}).exec(function getUsers(err, docs)
{
for (var i = 0; i < docs.length; i++) { userChunks.push(docs[i]);
}
res.render('Admin/viewAllEmployee', { title: 'All Employees', csrfToken:
req.csrfToken(), users: userChunks,
userName: req.session.user.name
});
});



});
/**
*     Description:
*     Displays add employee form to the admin.
*/
```

```
router.get('/add-employee', function addEmployee(req, res, next) { var messages
= req.flash('error');

var newUser = new User(); res.render('Admin/addEmployee', {









});









});


title: 'Add Employee', csrfToken: req.csrfToken(), user: config_passport.User,
messages: messages,

hasErrors: messages.length > 0, userName: req.session.user.name




/**

*     Description:

*     First it gets the id of the given employee from the parameters.

*/

router.get('/all-employee-projects/:id', function getAllEmployeePojects(req,
res, next) {

var employeeId = req.params.id; var projectChunks = [];


//find is asynchronous function Project.find({employeeID:
employeeId}).sort({_id: -

1}).exec(function findProjectOfEmployee(err, docs) { var hasProject = 0;
```

```
if (docs.length > 0) { hasProject = 1;
}
for (var i = 0; i < docs.length; i++) { projectChunks.push(docs[i]);
}
User.findById(employeeId, function getUser(err, user) { if (err) {
console.log(err);
}
res.render('Admin/employeeAllProjects', { title: 'List Of Employee Projects',
hasProject: hasProject,
projects: projectChunks, csrfToken: req.csrfToken(), user: user,
userName: req.session.user.name
});
/**
*/
});
});
});
router.get('/leave-applications', function getLeaveApplications(req, res, next)
{
var leaveChunks = [];
var employeeChunks = []; var temp;
//find is asynchronous function
Leave.find({}).sort({_id: -1}).exec(function findAllLeaves(err, docs) {
var hasLeave = 0;
if (docs.length > 0) { hasLeave = 1;
```

```
}
for (var i = 0; i < docs.length; i++) { leaveChunks.push(docs[i])
}
for (var i = 0; i < leaveChunks.length; i++) {
User.findById(leaveChunks[i].applicantID, function getUser(err, user) {
if (err) {
console.log(err);
}
employeeChunks.push(user);
})
}




seconds


// call the rest of the code and have it execute after 3
setTimeout(render_view, 900); function render_view() {
res.render('Admin/allApplications', { title: 'List Of Leave Applications',
csrfToken: req.csrfToken(), hasLeave: hasLeave,
leaves: leaveChunks,
employees: employeeChunks, moment: moment, userName:


req.session.user.name
});









});


}
});



/**
*     Description:
```

```
*/

router.get('/respond-application/:leave_id/:employee_id', function
respondApplication(req, res, next) {

var leaveID = req.params.leave_id;

var employeeID = req.params.employee_id; Leave.findById(leaveID, function
getLeave(err, leave) {

if (err) {

console.log(err);

}


User.findById(employeeID, function getUser(err, user) { if (err) {

console.log(err);

}

res.render('Admin/applicationResponse', { title: 'Respond Leave Application',
csrfToken: req.csrfToken(),

leave: leave, employee: user,

moment: moment, userName: req.session.user.name

});




})




});




});


/**

*     Description:

*/

router.get('/employee-profile/:id', function getEmployeeProfile(req, res, next)
{

var employeeId = req.params.id; User.findById(employeeId, function getUser(err,
user) {

if (err) {

console.log(err);

}
```

```
res.render('Admin/employeeProfile', { title: 'Employee Profile', employee: user,

csrfToken: req.csrfToken(), moment: moment,

userName: req.session.user.name

});


});

});


/**

*     Description:

*/

router.get('/edit-employee/:id', function editEmployee(req, res, next) {

var employeeId = req.params.id; User.findById(employeeId, function getUser(err,
user) {

if (err) {


res.redirect('/admin/');

}

res.render('Admin/editEmployee', { title: 'Edit Employee', csrfToken:
req.csrfToken(), employee: user,

moment: moment, message: '',

userName: req.session.user.name

});




});

});


/**

*     Description:


*

*     Known Bugs: None

*/

router.get('/edit-employee-project/:id', function editEmployeeProject(req, res,
next) {

var projectId = req.params.id;
```

```
Project.findById(projectId, function getProject(err, project) { if (err) {

console.log(err);

}

res.render('Admin/editProject', { title: 'Edit Employee', csrfToken:
req.csrfToken(), project: project,

moment: moment, message: '',

userName: req.session.user.name

});




});


});

/**

*     Description:

*     Gets the id of the employee from parameters.

*     Displays the add employee project form to the admin.

*

*     Known Bugs: None

*/

router.get('/add-employee-project/:id', function addEmployeeProject(req, res,
next) {


var employeeId = req.params.id; User.findById(employeeId, function getUser(err,
user) {

if (err) {

res.redirect('/admin/');

}

res.render('Admin/addProject', { title: 'Add Employee Project', csrfToken:
req.csrfToken(), employee: user,

moment: moment, message: '',

userName: req.session.user.name

});

});


});
```

```
/**
*     Description:
*     First finds project in the Project Schema with the help of id from the
parameters.
*     Gets the Employee of the project.
*
*     Known Bugs: None
*/
router.get('/employee-project-info/:id', function viewEmployeeProjectInfo(req,
res, next) {
var projectId = req.params.id;
Project.findById(projectId, function getProject(err, project) { if (err) {
console.log(err);



user) {


}
User.findById(project.employeeID, function getUser(err,
if (err) {
console.log(err);


}
res.render('Admin/projectInfo', {
title: 'Employee Project Information', project: project,
employee: user, moment: moment, message: '',
userName: req.session.user.name, csrfToken: req.csrfToken()
});
})
});


});


/**
*     Description:
*     Redirects admin to the employee profile page.
*     Known Bugs: None
```

```
*/
router.get('/redirect-employee-profile', function viewEmployeeProfile(req, res,
next) {
var employeeId = req.user.id;
User.findById(employeeId, function getUser(err, user) { if (err) {
console.log(err);
}
res.redirect('/admin/employee-profile/' + employeeId);
});

});
/**
*     Description:
*     Displays the admin its own attendance sheet
*     Known Bugs: None
*/
router.post('/view-attendance', function viewAttendance(req, res, next) {
var attendanceChunks = []; Attendance.find({
employeeID: req.session.user._id, month: req.body.month,
year: req.body.year
}).sort({_id: -1}).exec(function viewAttendanceSheet(err, docs)
{
var found = 0;
if (docs.length > 0) { found = 1;
}
for (var i = 0; i < docs.length; i++) { attendanceChunks.push(docs[i]);
}
res.render('Admin/viewAttendanceSheet', { title: 'Attendance Sheet',
month: req.body.month, csrfToken: req.csrfToken(), found: found,
attendance: attendanceChunks, userName: req.session.user.name, moment: moment
});
});

});
/**
*     Description:
*     Known Bugs: None
```

```
*/

router.get('/view-attendance-current', function
viewCurrentlyMarkedAttendance(req, res, next) {

var attendanceChunks = [];

Attendance.find({

employeeID: req.session.user._id, month: new Date().getMonth() + 1, year: new
Date().getFullYear()

}).sort({_id: -1}).exec(function getAttendanceSheet(err, docs) { var found = 0;

if (docs.length > 0) { found = 1;

}

for (var i = 0; i < docs.length; i++) { attendanceChunks.push(docs[i]);

}

res.render('Admin/viewAttendanceSheet', { title: 'Attendance Sheet',

month: new Date().getMonth() + 1, csrfToken: req.csrfToken(), found: found,

attendance: attendanceChunks, moment: moment,

userName: req.session.user.name

});

});


});

/**

*     Description:

*     Displays the attendance sheet of the given employee to the admin.

*/

router.get('/view-employee-attendance/:id', function viewEmployeeAttendance(req,
res, next) {

var attendanceChunks = [];

Attendance.find({employeeID: req.params.id}).sort({_id: - 1}).exec(function
getAttendanceSheet(err, docs) {

var found = 0;

if (docs.length > 0) { found = 1;

}

for (var i = 0; i < docs.length; i++) { attendanceChunks.push(docs[i]);

}


User.findById(req.params.id, function getUser(err, user) {

res.render('Admin/employeeAttendanceSheet', { title: 'Employee Attendance
Sheet', month: req.body.month,
```

```
csrfToken: req.csrfToken(), found: found,
attendance: attendanceChunks, moment: moment,
userName: req.session.user.name
,
'employee_name': user.name
})
});
});



});
/**
*     Description:

*/
router.post('/add-employee', passport.authenticate('local.add- employee', {
successRedirect: '/admin/redirect-employee-profile', failureRedirect:
'/admin/add-employee', failureFlash: true,
}));
/**
*     Description:

*/
router.post('/respond-application', function respondApplication(req, res) {
Leave.findById(req.body.leave_id, function getLeave(err, leave)
{
leave.adminResponse = req.body.status; leave.save(function saveLeave(err) {
if (err) {
console.log(err);
}
res.redirect('/admin/leave-applications');
})
})


});
router.post('/edit-employee/:id', function editEmployee(req, res) { var
employeeId = req.params.id;
```

```
var newUser = new User(); newUser.email = req.body.email;

if (req.body.designation == "Accounts Manager") { newUser.type =
"accounts_manager";

}

else if (req.body.designation == "Project Manager") { newUser.type =
"project_manager";

}

else {

newUser.type = "employee";

}

newUser.name = req.body.name,

newUser.dateOfBirth = new Date(req.body.DOB), newUser.contactNumber =
req.body.number, newUser.department = req.body.department;

newUser.Skills = req.body['skills[]']; newUser.designation =
req.body.designation;


User.findById(employeeId, function getUser(err, user) { if (err) {

res.redirect('/admin/');

}

if (user.email != req.body.email) {

User.findOne({'email': req.body.email}, function getUser(err, user) {

if (err) {

res.redirect('/admin/');

}

if (user) {

res.render('Admin/editEmployee', { title: 'Edit Employee', csrfToken:
req.csrfToken(), employee: newUser,

moment: moment,

message: 'Email is already in use', userName: req.session.user.name

});

}

});

}

user.email = req.body.email;

if (req.body.designation == "Accounts Manager") { user.type =
"accounts_manager";

}
```

```
else if (req.body.designation == "Project Manager") { user.type =
"project_manager";

}

else {

user.type = "employee";

}

user.name = req.body.name,

user.dateOfBirth = new Date(req.body.DOB), user.contactNumber = req.body.number,


user.department = req.body.department; user.Skills = req.body['skills[]'];
user.designation = req.body.designation;


user.save(function saveUser(err) { if (err) {

console.log(error);

}

res.redirect('/admin/employee-profile/' + employeeId);


});

});


});

router.post('/add-employee-project/:id', function addEmployeeProject(req, res) {

var newProject = new Project(); newProject.employeeID = req.params.id;
newProject.title = req.body.title; newProject.type = req.body.type;

newProject.startDate = new Date(req.body.start_date), newProject.endDate = new
Date(req.body.end_date), newProject.description = req.body.description,
newProject.status = req.body.status;

newProject.save(function saveProject(err) { if (err) {

console.log(err);

}

res.redirect('/admin/employee-project-info/' + newProject._id);

});

});

router.post('/edit-employee-project/:id', function editEmployeeProject(req, res)
{

var projectId = req.params.id; var newProject = new Project();

Project.findById(projectId, function (err, project) { if (err) {

console.log(err);

}
```

```
project.title = req.body.title; project.type = req.body.type;
project.startDate = new Date(req.body.start_date), project.endDate = new
Date(req.body.end_date), project.description = req.body.description,
project.status = req.body.status;
project.save(function saveProject(err) { if (err) {
console.log(err);




projectId);
});


}
res.redirect('/admin/employee-project-info/' +


});
});



router.post('/delete-employee/:id', function deleteEmployee(req, res) {
var id = req.params.id;
User.findByIdAndRemove({_id: id}, function deleteUser(err) { if (err) {
console.log('unable to delete employee');
}
else {
res.redirect('/admin/view-all-employees');




});


}
});

router.post('/mark-attendance', function markAttendance(req, res, next) {

Attendance.find({
```

```
employeeID: req.session.user._id, date: new Date().getDate(), month: new
Date().getMonth() + 1, year: new Date().getFullYear()

}, function getAttendance(err, docs) { var found = 0;

if (docs.length > 0) { found = 1;

}

else {

var newAttendance = new Attendance(); newAttendance.employeeID =
req.session.user._id; newAttendance.year = new Date().getFullYear();
newAttendance.month = new Date().getMonth() + 1; newAttendance.date = new
Date().getDate(); newAttendance.present = 1; newAttendance.save(function
saveAttendance(err) {

if (err) {

console.log(err);

}

});

}

res.redirect('/admin/view-attendance-current');

});


});

module.exports = router;

function isLoggedIn(req, res, next) { if (req.isAuthenticated()) {

return next();

}

res.redirect('/');

}



function notLoggedIn(req, res, next) { if (!req.isAuthenticated()) {

return next();

}

res.redirect('/');

}
```