Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

(повна назва факультету)

Кафедра комп’ютерних систем та мереж

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

(назва освітнього ступеня)

на тему: Клієнт-серверна інформаційна система ІТ компанії

Виконав: студент IV курсу, групи СІ-43
спеціальності 123 «Комп’ютерна інженерія»
(шифр і назва галузі знать, спеціальності)

_____ Копанецький А.Л.
(підпис) (прізвище та ініціали)

Керівник _____ Лупенко С.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Тиш Є.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Осухівська Г.М.
(підпис) (прізвище та ініціали)

Рецензент _____ Гащин Н.Б.
(підпис) (прізвище та ініціали)

Тернопіль

2022

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*

(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« ____ » _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр

за спеціальністю 123 «Комп'ютерна інжерерія»

студенту Копанецькому Артуру Любомировичу

1. Тема проекту Клієнт-серверна інформаційна система IT компанії

керівник роботи Лупенко Сергій Анатолійович д.т.н., професор кафедри КС

(прізвище, ім'я по батькові, науковий ступінь, вчене звання)

затверджені наказом ректора від «__» _____ 20___ р. № _____

2. Термін подання студентом завершеної роботи 23 червня 2022р.

3. Вихідні дані до роботи Розробка клієнт-серверної інформаційної системи для IT компанії

4. Зміст роботи (перелік питань, які потрібно розробити)

1. Розробка клієнт-серверної програми, як в перспективі може залучити покупців

2. Інструментальні засоби реалізації проекту

3. Практична реалізація

4. Безпека життєдіяльності та охорони праці

5. Перелік графічного матеріалу (з точки зазначенням обов'язкових креслень)

1. Блок схема авторизації на сайті

2. Логічна модель БД

3.Сторінки веб-сайту

4.Структурна схема проекту

5.Структурна схема веб-сайту

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Безпека життєдіяльності, основи хорони праці	Лазарюк В. В.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		*Виконано*
2.	Здійснення аналітичного огляду літератури за заданою темою.		*Виконано*
3.	Аналіз та дослідження ринку клієнт-серверних програм та веб-сервісів		*Виконано*
4.	Вибір середовища для розробки та допоміжних програм.		*Виконано*
5.	Аналіз особливостей проектування, та розробка застосунку для ІТ компанії.		*Виконано*
6.	Практична реалізація об’єкта проектування		*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»		*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони праці»		*Виконано*
9.	Оформлення пояснювальної записки дипломного проекту згідно з вимогами.		*Виконано*
10.	Нормоконтроль		*Виконано*
11.	Перевірка на плагіат		*Виконано*
12.	Попередній захист кваліфікаційної роботи		*Виконано*
13.	Захист кваліфікаційної роботи		*Виконано*

Студент _____ *Копанецький А.Л*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Лупенко.С.А*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Клієнт-серверна інформаційна система ІТ компанії // Кваліфікаційна робота освітнього рівня «Бакалавр» // Копанецький Артур Любомирович // Тернопільскький національний технічний університет імені Іівнв Полюя, факультет комп`ютерно-інформаційних систем і програмної інженерії, кафедра комп`ютерних систем та мереж, група Сіс-43 // Тернопіль, 2022 //С.-53, рис.-23, додатки 3, бібліогр.15.

Ключові слова: ІТ, клієнт-сервер, Java, інтерфейс, SOAP, багаторівнева архітектура, framework.

Робота складається із вступу, чотирьох розділів, висновку, списку використаних джерел і додатків.

У вступі обґрунтовується актуальність теми та формулюються задачі подальшого дослідження. У першому розділі даної дипломної роботи наведено загальний аналіз клуєнт серверної архітектури, та необхідність їх у ІТ компаній. У другому розділі наведено опис програмних продуктів і сервісів, які були задіяні у проекті, та які використовуються у популярих ІТ компаніях. Третій розділ присвячений опису лістингу застосунку та опис повного його функціоналу. Четверний розділ присвячений опису питань із безпеки життєдіяльності, такі як аварія з викидом радіоактивних речовин, та долікарська допомога при обмороженні. У висновку містяться поради по вдосконаленню застосунку, наведено висновки по роботі, а також по тестуванні та оптимізації.

Результатом дипломного проектування є повнофункціональний сайт, призначена для ІТ компаній, таких як сервісні центри, для змоги розповсюджеггя своїх товарів, та для відслідковування та управління клієнтською базою.

ABSTRACT

Client-server information system of the IT company // Qualification work of the educational level "Bachelor" // Kopanetsky Artur Lyubomirovich // Ternopil National Technical University named after Iivn Polyuya, Faculty of Computer Information Systems and Software Engineering, Department of Computer Systems and Networks, group Sis-43 // Ternopil, 2022 //S.-53, fig.-23, appendices 3, bibliogr.15.

Keywords: IT, client-server, Java, interface, SOAP, multilevel architecture, framework.

The work consists of an introduction, four chapters, a conclusion, a list of sources and appendices. The introduction substantiates the relevance of the topic and formulates the objectives of further research. The first section of this thesis presents a general analysis of client architecture clients, and their need for IT companies. The second section describes the software products and services involved in the project and used by popular IT companies. The third section is devoted to the description of the application listing and the description of its full functionality. The fourth section is devoted to the description of life safety issues, such as the accident with the release of radioactive substances, and pre-medical care for frostbite. The conclusion contains tips for improving the application, conclusions on the work, as well as testing and optimization. The result of the thesis is a full-featured site designed for IT companies, such as service centers, to distribute their products, and to track and manage customer base.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, ПОЗНАЧЕНЬ І ТЕРМІНІВ

IT – Information Technology. Інформаційні технології;

HTML – HyperText Markup Language. Мова розмітки веб сторінок

СУБД – Database Management System. Набір програм та комплексів для взаємопов’язання програм та доступу до них;

CSS – Cascading Style Sheets. Мова стилів, для задання візуального виду документа, написаному на HTML

JSP – JavaServer Pages. Використовується для динамічної генерації HTML та XML веб-сторінок;

API – Application Programming Interface. Програмний набір протоколів та інструментів для створення програмного забезпечення;

ПК – Персональний комп’ютер;

XML – eXtensible Markup Language. Розширена мова розмітки.

ВСТУП

Сьогодні суспільство розвивається дуже швидко, тому що людям необхідно не відставати від змін і обробляти величезні обсяги інформації з усього світу. Через це інформаційні технології сьогодні є одним із найпріоритетніших сфер для визначення майбутнього розвиток суспільства.

Інформаційні системи в частності клієнт-сервер та її архітектура набула популярності завдяки дуже швидкому розвитку інтернету-мережі та накопичення дуже великої кількості інформації на серверах базах даних. Архітектуру клієнт-сервер можна визначити як концепцію інформаційних мереж, де більшість інформації та ресурсів зосереджені на серверах, що обслуговують клієнтські запити.

Існує велика кількість веб-серверів, на яких розміщена та чи інша інформація, одним із прикладів таких є сервіс www, на якому розміщена та збережена велика кількість інформації. У найпростішому випадку ця інформація являє собою набір сторінок, які зберігаються на сервері як файли, розміщені за допомогою розмітки HTML. Ситуація в більшості випадків є складнішою тому, що більшість мережевих ресурсів на цьому етапі є динамічними, тобто їх не існує в заздалегідь готовому вигляді, а створюються вони безпосередньо в процесі обробки корисутвацьких запитів.

Головна задумка цієї архітектури є в тому, щоб зробити поділ веб-додатоку на декілька компонентів, кожен з яких здійснює певний набір послуг. Компоненти такого застосунку можуть бути розгорнуті на різних комп’ютерах, та виконувати функціонал сервера або клієнта. Це значно збільшує надійність, безпеку та продуктивність веб-додатків і усієї мережі.

На нинішньому етапі розвнитку для програмування модулів проміжневих ступенів застосовується серверна мова сценаріїв PHP, а для керування даними – система управління базами даних mysql. Одночасно засоби керування даними, так і middlewere-засоби можуть бути дуже різноманітними. Але для створення подібних серверних застосунків, може

використовуватись не тільки PHP, а і широко відомі мови програмування Java та Python, які набули популярності вже давно.

В рамках дипломного проекту створюється програмний код, на основі якого в подальшому будується клієнт-серверний застосунок у вигляді сайту. Все на даному сайті є оригінальним та було створено для цього проєкту.

Метою даної робити є створення сайту, backend якого написаний на мові програмування Java, а frontend на мовах css та html, на основі цього створюється невелика клієнт-серверний застосунок для IT компанії, яка може залучити потенційних клієнтів та працівників. Так само створений застосунок може бути основою для створення наступних програм, добавивши в неї ще більше функціоналу значно скоротивши час на їх створення приблизно на 40-50%.

Предметом дослідження є середовище програмування IntelliJ IDEA, а також компоненти цього середовища, MySql, PowerDesigner, Maven, Javafx.

Об’єктом дослідження є аналогічні продукти на сучасному ринку.

Досягнення мети цієї роботи передбачає собою розв’зання наступних завдань:

– огляд літератури по заданій темі;

– визначення та опис постановки завдання;

– побудова алгоритму реалізації програми;

– реалізація алгоритму програми в середовищі IntelliJ IDEA;

– опис користувацького інтерфейсу;

– опис інтерфейсу працівника (адміністратора);

– розробка корисного та зручного для користувача сайту;

– тестування та оптимізація.

Тому темою роботи є реалізація клієнт-серверного застосунку для додавання, редагування та видалення клієнтів, зміни їх тарифів тощо. Демонстрація системи відбуватиметься через інтерфейс користувача та працівника.

РОЗДІЛ 1 ОГЛЯД ОСНОВНОЇ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1 Визначення архітектури

Архітектура клієнт-сервер базується на двох компонентах: клієнті і сервері. Клієнт – це клієнтський комп'ютер, який надсилає запит серверу на надання інформації або виконання дії. Сервер – це великий потужний комп'ютер або апаратне забезпечення, призначене для виконання конкретних завдань виконання коду, виконання сервісних функцій на запити клієнтів, надання користувачам доступу до певних ресурсів, а також зберігання інформації та баз даних.Тип такої роботи полягає в тому, що користувач (клієнт) надсилає запит до серверу, який його обробляє, і передає користувачу уже завершений результат. Сервер може обслуговувати декілька клієнтів одночасно. Якщо одночасно надходить кілька запитів, вони стають у чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритет.

Запити з вищим пріоритетом мають виконуватися раніше

Функції, реалізовані сервером:

– зберігання, доступ, захист та резервне копіювання;

– обробка запитів клієнтів;

– надіслання результатів (відповіді) клієнту

Функції реалізовані на стороні клієнта:

– надати користувацький інтерфейс;

– формулювання запитів до сервера та його відправників;

– отримувати результати запиту та надсилати додаткові команди

Архітектура клієнт-сервер визначає принципи зв'язку між комп'ютерами, а правила та взаємодії визначаються в протоколі.

1.2 Опис клієнт-серверної моделі

Модель обчислень клієнт-сервер домінує в підході розподілених обчислень. Ця модель працює таким чином що розбиває програму на декілька різних частинок, після чого вона розміщується на різних платформах за для підвищення ефективності роботи Це означає те що программа яка керує даними знаходиться на сервері, а программа візуалізації даних знаходиться на пристрої користувача, як показано на рисунку 1.1 Даний розподіл може усунути наступні недоліки персональних комп'ютерів: мала обчислювальна потужність, надійність, та ізоляція між різними ПК.

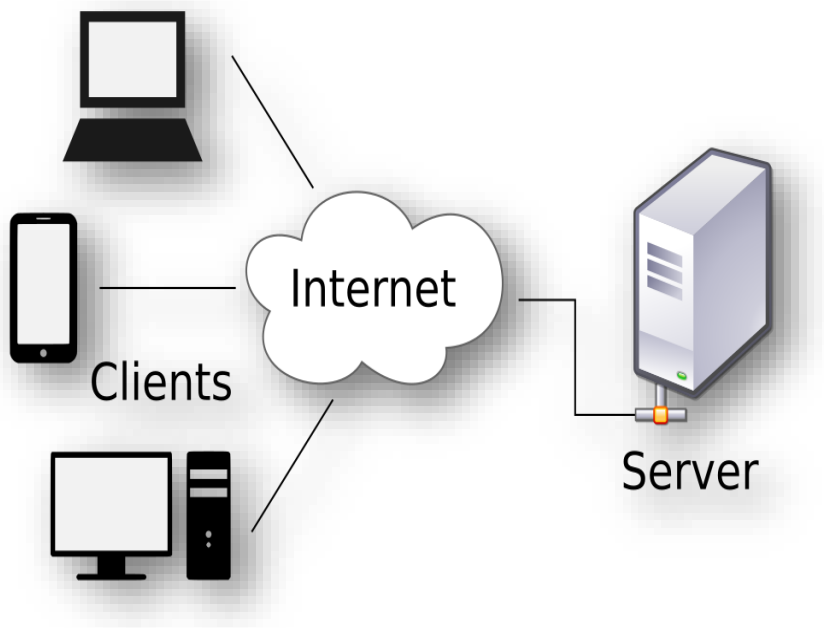

Рисунок 1.1 – Архітектура клієнт-сервер

Користувачу ПК звісно також потрібна велика потужність обчислювать та не малий спектр користувацьких функцій. В залежно від вжитої програми та програмного забезпечення розподілення обробки та аналіз даних може відрізнятися. Робота серверного устаткування полягає в тому щоб приймати запити від клієнтів та відсилати їм у відповідь очікуваний результат.

1.3 Класси клієнт-серверних застосунків

Розподіл роботи між сервером та клієнтом переважно виконується по різному у структурі клієнт-серверних застосунків.

Точний обсяг часу який знадобиться для обробки інформації залежить від характеру інформації, що міститься в базі даних, від наявності апаратного забезпечення, та від типу підтримуваних програм, яке може працювати разом, та від характеру використовуваних даних. На рисунках 1.2–1.5 показано схеми деяких основних категорій застосування. Між сервером та клієнтом можливі також і інші варіанти розподілу завдать. Ці декілька тип архітектур з різними можливими варіантами розподіленная завдань між серверною та клієнтською частинами можна побачити на рисунках, що зазначено вище.

1.3.1 Аналіз даних у хоста

Цей сценарій не є справжньою програмою клієнт-сервер, але він відноситься до традиційного середовища mainfraim, де вся або майже вся обробка даних відбувається на mainfraim. У більшості випадків у таких обчислювальних середовищах користувальницький інтерфейс має вигляд оригінального терміналу.

Але й тоді коли користувач (клієнт) використовує ПК, в обробці даних на хості його роль буде обмежуватить емуляцією інтерфейсу терміналу.

Така схема підходить, коли основна обчислювальна потужність зосереджена на стороні сервера. Як правило, в цьому випадку потужність клієнта набагато нижча, ніж необхідна для виконання завдання по створенню сервера[2].

Рисунок 1.2 – Аналіз даних на базі хоста

1.3.2 Аналіз даних на сервері

Найпростіша конфігурація клієнт-сервер полягає в тому, що клієнт відповідає лише за надання графічного інтерфейсу користувача, а майже вся обробка даних виконується на сервері. Це зменшує навантаження на серверну частину, особливо якщо логіка форми складна і вимагає великих системних ресурсів. Обробка даних дозволяє розмістити базу даних в різних вузлах комп'ютерної мережі декілька раз. Таким чином, кожен компонент бази даних розташовується за місцем наявності техніки та її обробки.

Рисунок 1.3 – Аналіз даних на базі сервера

1.3.3 Аналіз даних на основі клієнта

Цей сценарій показує протилежний підхід: майже вся обробка даних виконується на стороні клієнта, за винятком перевірки цілісності даних та іншої логіки, пов’язаної з обслуговуванням бази даних, яка переважно виконується на сервері.

Часто складна функціональність для роботи з базою даних знаходиться на стороні клієнта. Реалізація цієї схеми є найпоширенішою реалізацією архітектури клієнт-сервер. Це дозволяє користувачам використовувати програми, які відповідають їхнім локальним потребам.

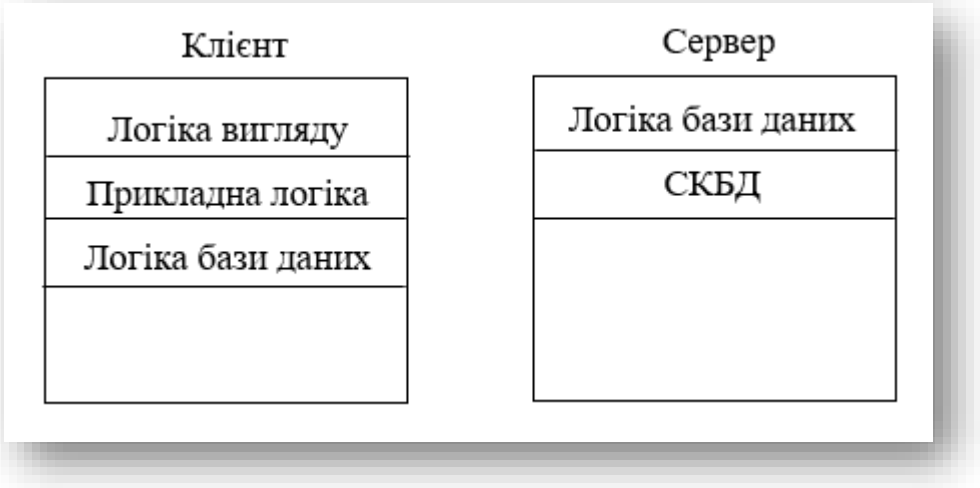

Рисунок 1.4 – Аналіз даних на базі клієнта

Обробка даних дозволяє розмістити базу даних кілька разів у різних вузлах комп’ютерної мережі. Таким чином, кожен компонент бази даних знаходиться на пристрої та де він обробляється.

1.3.4 Спільна обробка даних

У цій конфігурації обробка даних оптимізована з урахуванням переваг клієнта і сервера, а також того факту, що дані розподіляються.

Такі зміни набагато складніші в установці та обслуговуванні, але в довгостроковій перспективі вони забезпечують кращу продуктивність, а так же ефективність мережевих ресурсів, ніж інші методи реалізації архітектури клієнт-сервер.

Рисунки 1.4 і 1.5 відповідають конфігураціям, які дають велике навантаження на клієнтів, ці типи клієнтів називаються «товстими або жирними» клієнтами.

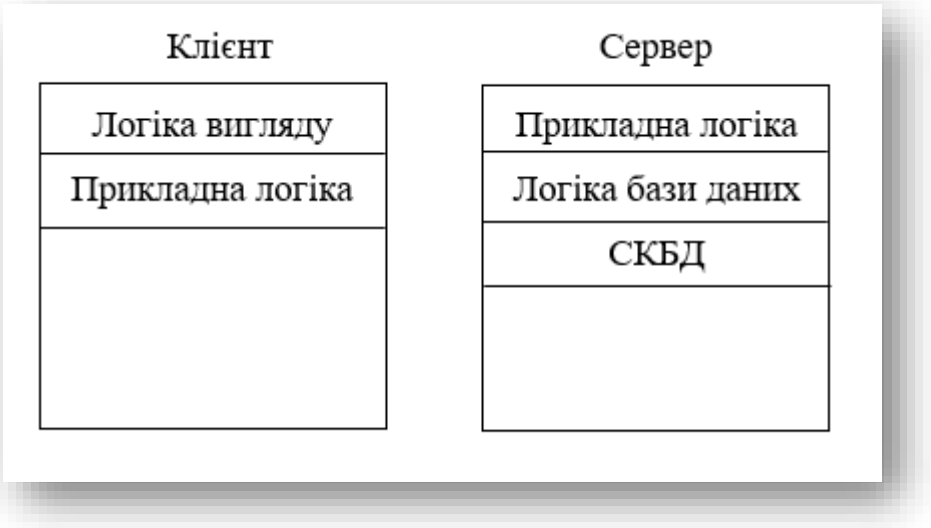

Рисунок 1.5 – Сумісна обрабка даних

Жирний або «Rich-клієнт» в архітектурі клієнт-сервер – це програма, яка забезпечує розширені функціональні можливості незалежно від центрального сервера. У цьому випадку сервер зазвичай є просто сховищем даних, і вся робота по обробці та рендерингу даних передається на машину клієнта.

1.4 Класифікація серверних архітектур

1.4.1 Трирівнева архітектура

У комп’ютерній техніці трирівнева архітектура, що є синонімом трирівневої архітектури, означає наступні програмні компоненти: клієнтський додаток (часто званий «тонкий клієнт» або термінал), який підключається до сервера додатків, який, у свою чергу, підключається до сервера бази даних.

У простій конфігурації фізичний сервер додатків можна об’єднати з сервером баз даних на одному комп’ютері, до якого через мережу під’єднано один або кілька терміналів.

У «правильній» конфігурації сервер бази даних знаходиться на виділеному комп’ютері (або кластері), до якого через мережу під’єднано один або кілька серверів додатків, а термінал – через мережу.

У порівнянні з архітектурою клієнт-сервер або файл-сервер, трирівнева архітектура може виділити наступні переваги:

– масштабованість;

– можливість налаштування;

– Ізоляція між рівнями дозволяє (шляхом правильного розгортання архітектури) швидко та легко налаштувати систему у разі збою або для планового обслуговування на одному з рівнів;

– висока безпека;

– висока надійність;

– низькі вимоги до швидкості каналу (мережі) між терміналом і сервером додатків ;

– через зниження вартості терміналу вимоги до експлуатаційних характеристик і технічні характеристики терміналу нижчі. Терміналом може бути не тільки комп’ютер, а й, наприклад, мобільний телефон.

Недоліки є перевагами. У порівнянні з архітектурами клієнт-сервер або файл-сервер, трирівневі архітектури можуть виявити такі недоліки:

– вища складність створення програми;

– важче розгорнути та керувати;

– високі вимоги до продуктивності до серверів додатків і серверів баз даних, отже, висока вартість серверного обладнання;

– швидкість каналу між сервером баз даних і сервером додатків дуже висока.

1.4.2 Дворівнева архітектура

Дворівнева архітектура використовується для систем клієнт-сервер, де сервер безпосередньо і повністю відповідає на запити клієнта, використовуючи лише власні ресурси. Тобто сервер не викликає сторонні мережеві програми, але не отримує доступу до сторонніх ресурсів для виконання будь-якої частини запиту.

Розташування компонентів на стороні клієнта або сервера визначає таку базову модель їх взаємодії в дворівневій архітектурі:

– сервер терміналів - розподілене представлення даних; Файловий сервер;

– доступ до віддалених баз даних і файлових ресурсів; сервер баз даних;

– віддалене представлення даних; Сервер додатків - віддалена програма.

Переваги цього підходу є очевидними:

– можливе централізоване управління програмними функціями;

– знизити вартість володіння системою, орендуючи сервери замість їх покупки;

– значне скорочення мережевого трафіку.

1.4.3 Баготорівнева архітектура

Багаторівнева архітектура клієнт-сервер є прямим продовженням поділу інтерфейсу користувача, компонентів обробки та програм на рівні даних.

Різні рівні взаємодіють відповідно до логічної організації програми. У багатьох бізнес-додатках розподілена обробка еквівалентна організації багатошарової архітектури програми клієнт-сервер. Такий розподіл називається вертикальним.

Основною особливостю цього підходу є розміщення логічно різних компонентів на різних машинах.

Прикладом є веб-сервер, який реплікується на кілька комп'ютерів у локальній мережі. Коли ви змінюєте веб-сторінку – зміни будуть надіслані на всі сервери. Сервер, на який надсилатимуться вхідні запити, вибирається на основі каруселі. Ця форма розповсюдження використовується для балансування навантаження на популярних серверах веб-сайтів.

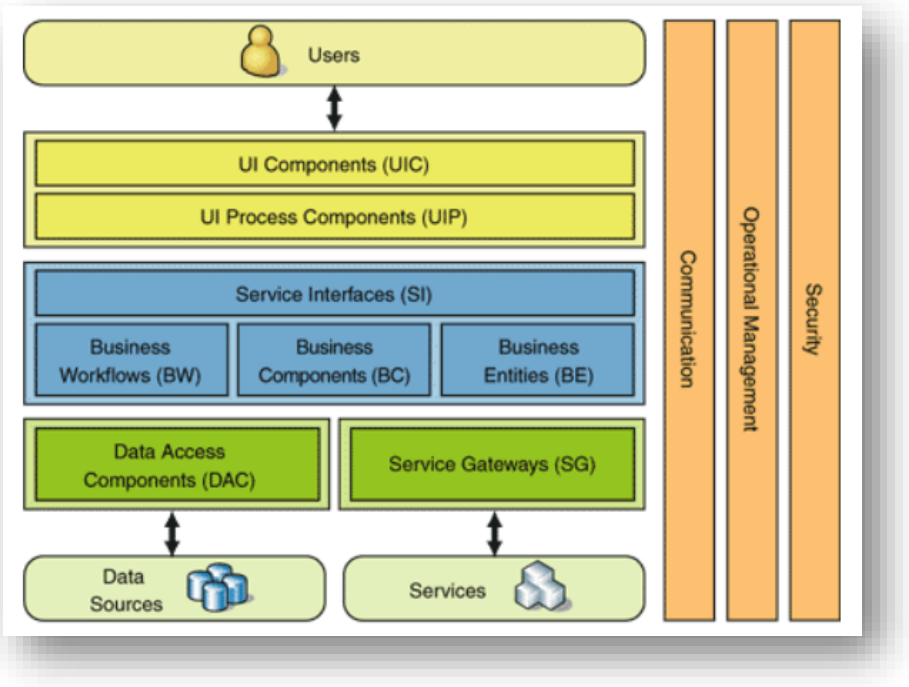

Рисунок 1.6 – Один із прикладів багатошарової архітектури

1.5 Клієнт-серверна взаємодія

Модель взаємодії клієнт–сервер насамперед визначається розподілом відповідальності між клієнтом і сервером. Логічно можна виділити три рівні операцій:

– рівень представлення даних, який по суті є інтерфейсом користувача, відповідає за представлення даних користувачеві та введення від нього команд керування;

– прикладний рівень, який реалізує основну логіку програми та необхідну обробку інформації;

– забезпечте рівень управління даними для зберігання та доступу до даних.

У мережі з виділеним файловим сервером серверна мережева операційна система встановлюється на виділений автономний ПК. Цей ПК стає сервером. Програмне забезпечення, встановлене на робочій станції, дозволяє їй спілкуватися з сервером.

Найпоширеніші мережеві операційні системи:

– Novel, компанія NetWare;

– Microsoft Windows NT;

– корпорація UNIX AT&T;

– Linux.

Мережева архітектура клієнт-сервер має наступні переваги:

– дозволяє організувати мережу з великою кількістю робочих місць;

– забезпечує централізоване управління обліковими записами користувачів, безпекою та доступом, спрощуючи управління мережею;

– ефективний доступ до мережевих ресурсів;

– користувачу потрібен пароль для входу в мережу та доступу до всіх ресурсів, на які поширюються права користувача.

Крім переваг мережевої архітектури клієнт-сервер, є багато недоліків:

– збій сервера призведе до збою в роботі мережі або, принаймні, до втрати мережевих ресурсів;

– потреба в кваліфікованих менеджерах;

– мають більш високу вартість мережі та мережевого обладнання.

РОЗДІЛ 2 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РЕАЛІЗАЦІЇ ДИПЛОМНОГО ПРОЕКТУ

2.1 Аналіз існуючих рішень на ринку клієнт-серверних програм

Ведення клієнтської бази на папері вже давно не може задовольнити запити сучасного бізнесу. Тому на етапі розвитку та формування бізнесу починається пошук потрібного програмного забезпечення.

Інформаційна система клієнт-сервер компанії – це платформа, яка дозволяє:

– управління клієнтською базою;

– облік клієнтів;

– імпорт та експорт клієнтської бази;

– управління клієнтською базою;

– встановлення тарифів на обслуговування;

– приймати та обробляти заявки від клієнтів;

– створювати таблиці з новими полями;

– організовувати взаємодію між клієнтами та адміністраторами.

Для того, щоб зрозуміти потреби у створенні нових рішень, далі будуть розглянені уже існуючі системи, показано їх переваги та недоліки, сфера використання і стиль інтерфейсу.

2.1.1 RemOnline

RemOnline – це хмарний сервіс, для якого потрібен лише Інтернет. Легкий у засвоєнні веб-додаток дозволяє всім співробітникам працювати комфортно та ефективно з першого дня, а керівники можуть контролювати їх роботу.

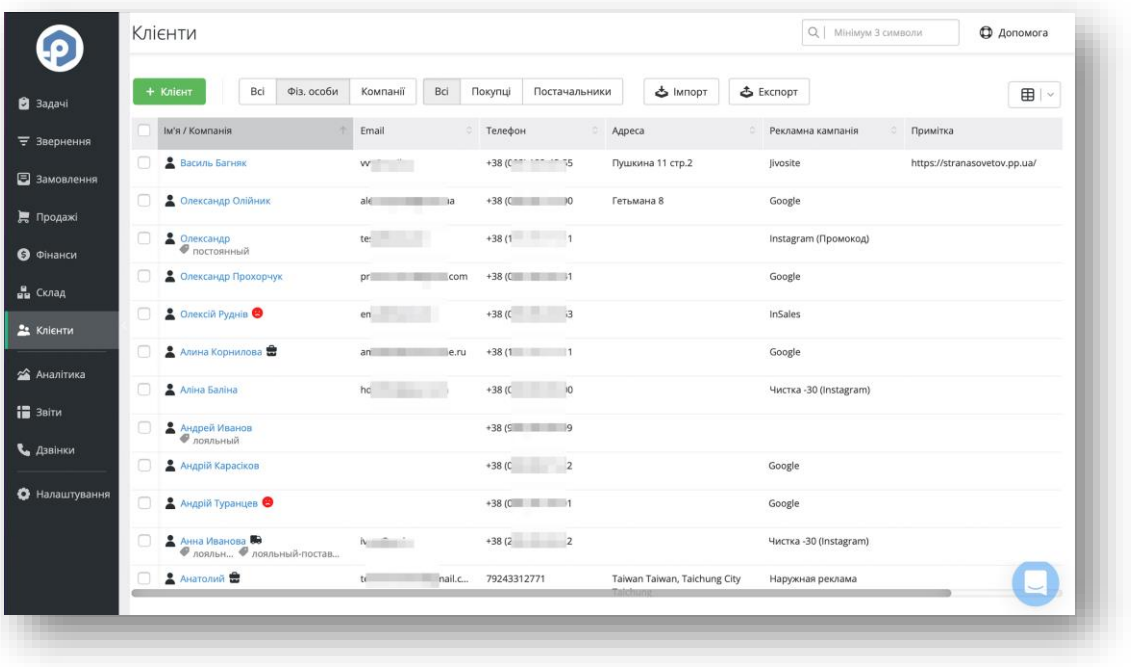

Рисунок 2.1 – Інтерфейс RemOnline

Переваги:

– швидкий старт і зрозумілий інтерфейс – новачкам не доведеться довго вчитися користуватися програмою;

– RemOnline не залежить від кількості та розташування майстерень - база даних клієнтів буде спільною для всіх;

– складні вкладки, розумний пошук і можливість створення фільтрів дозволяють швидко знаходити потрібну інформацію;

– форми та картки можна налаштувати, додавати, видаляти і вибирати те, що має відображатися;

– можна керувати своєю клієнтською базою за допомогою програми у будь якому місці.

Недоліки:

– платна, без відкритого вихідного коду;

– для розширення функціоналу або для виправлення помилок в системі може знадобитись допомога професіонала.

2.1.2 Opera hrs

HRS – найбільший готельний партнер Oracle у всьому світі, який обслуговує понад 10 000 клієнтів у 90 країнах. Визнана партнером Oracle, HRS пропонує широкий спектр інноваційних рішень, включаючи:

– управління готелями (PMS);

– POS, спа та дозвілля для гостей;

– бізнес-аналітика, платіжний шлюз;

– мобільні додатки, менеджер каналів;

– онлайн-бронювання;

– системи управління фінансами та персоналом;

– сканування та ідентифікація паспорта тощо.

Мета компанії – допомогти клієнтам стати сильнішими та ефективнішими, надаючи всі необхідні інструменти для побудови успішного, ефективного, прибуткового та стійкого бізнесу.

Інтегроване рішення Opera HRS базується на системі керування Opera RMS. Додаток налаштовується відповідно до ваших вимог і потреб. Це дозволяє швидко й легко отримувати точну інформацію про окремі готелі, спа тощо, в мережі, що складається з кількох об’єктів, об’єднаних в єдину базу даних Oracle.

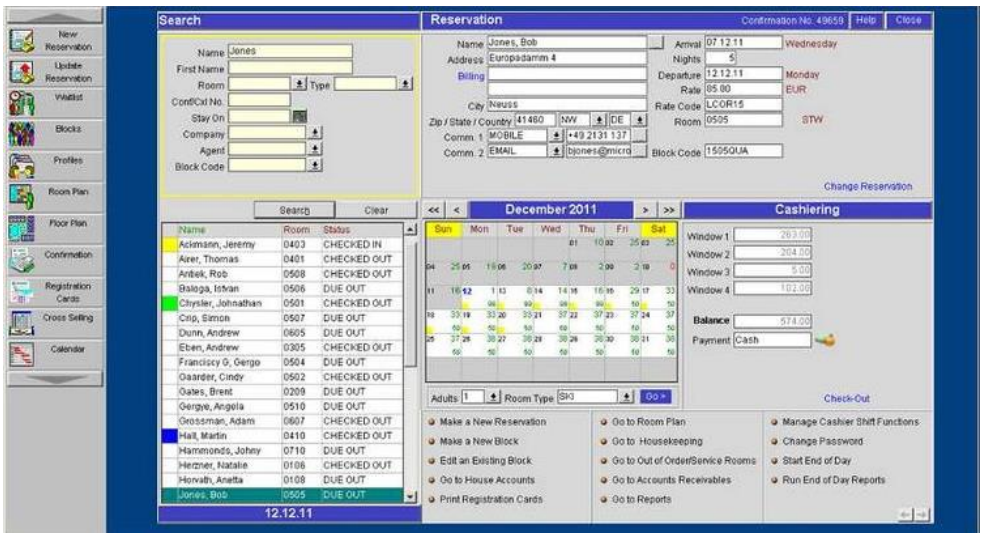

Рисунок 2.2 – Інтерфейс Opera hrs

2.2 Аналіз баз даних

База даних – це сукупність взаємопов'язаних і структурованих даних відповідно до типу метаданих. Метадані – це дані, які описують збережені дані. Метадані визначають, як дані зберігаються в базі даних.

Дані та метадані забезпечують середовище, в якому дані логічно впорядковані для легкого обслуговування та пошуку.

Метадані бази даних визначають структуру, в якій дані логічно організовані. Не всі бази даних мають однакову структуру. Існує багато різних моделей даних. Найчастіше реалізуються три такі моделі: ієрархічна, мережева та реляційна.

В нашому проекті була використана реляційна база даних, а саме MySql.

2.3 Характеристика середовищ розробки

Для виконання мною завдання дипломного проекту, були використані такі середовища:

– PowerDesigner – для проектування логічної та фізичної моделі бд (рис. 2.1 – 2.2);

– MySQL – система управління базою даних (СУБД);

– Maven – програма для автоматичного складання;

– Javafx – основна середа розробки за допомогою якої був розроблений застосунок.

– JSP – використовується для написання макета веб-сторінок, динамічно формується протягом серверного часу;

– Java Servlet API – для реалізації на сервері та роботи з клієнтом на основі запиту-відповіді;

– Apache Tomcat – основний сервер для проекту;

– Log4j – використовується як бібліотека для створення систем логування;

– Bootstrap – для кращого вигляду.

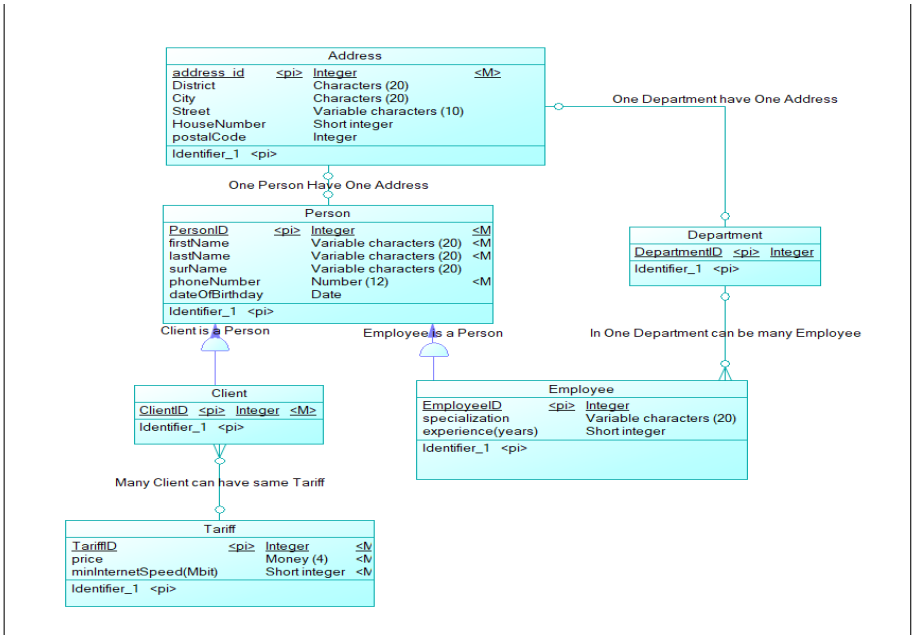

Рисунок 2.3 – Логічна модель БД.

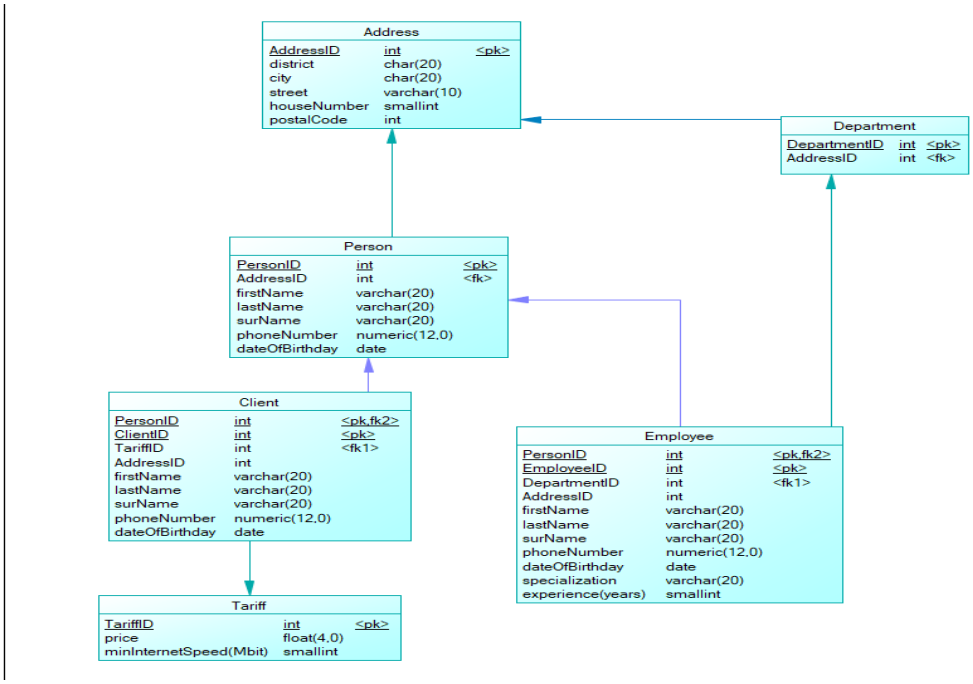

Рисунок 2.4 – Фізична модель БД.

2.3.1 Архітектура MySql

27 лютого 2008 року Sun Microsystems придбала MySQL AB за один мільярд доларів. 27 січня 2010 року Oracle Corporation викупила Sun Microsystems і додала mysql до своєї бази даних.

Функції сервера MySQL:

– простота установки та використання;

– підтримує необмежену кількість користувачів,

– які використовують базу даних одночасно;

– кількість рядків таблиці може досягати 50 мільйонів;

– швидкість виконання інструкцій висока;

– наявність простої та ефективної системи безпеки.

MySQL – це одна із безкоштовних систем керування реляційними базами даних. Дана система керування базою даних із відкритим кодом є створена як заміна комерційних систем. Зауважимо, що у вітчизняній літературі є синонімічні абревіатури СУБД і СКБД.

Зараз mysql є однією з найпоширеніших систем керування базами даних. Оскільки він має чудову підтримку різних мов програмування, він використовується для створення динамічних веб-сторінок. MySQL має в собі величезний набір функціоналу, та надає безпечне середовище для зберігання, підтримки та отримання даних[1].

Основні переваги MySQL полягають у наступному:

– Масштабованість. MySQL може підтримувати роботу дуже величезних баз даних, що підтверджує її впровадження в такі відому компанії як Associated Press, Yahoo!, HP, Google. Згідно з документацією, яка надається одразу ж із MySQL, деякі бази даних, які використовуються розробниками MySQL, зберігають до 50 мільйонів записів.

– Портативність. MySql має змогу працювати на всіх видах платформ та операційних системах таких як: Linux, Unix, Windows, OS/2, Solaris, Mac OC.

– Підключення. MySQL має мережеву структуру. Декілька користувачів можуть одночасно отримати доступ до MySQL з будь-якої точки Інтернету. MySql має обширний інтнрфейс для розробників (API), що дозволяє підключатися до MySql із програм, написаних C, C++, Perl, PHP, Jva, Python та багатьох інших.

– Безпека. MySQL є дуже безпечним через те, що має систему контролю доступу, яка шифрує данні під час передачі при необхідності.

– Зручність та ефективність в експлуатації, простота у встановленні та адмініструванні.

– Швидкість функціонування.

– Відкритий код.

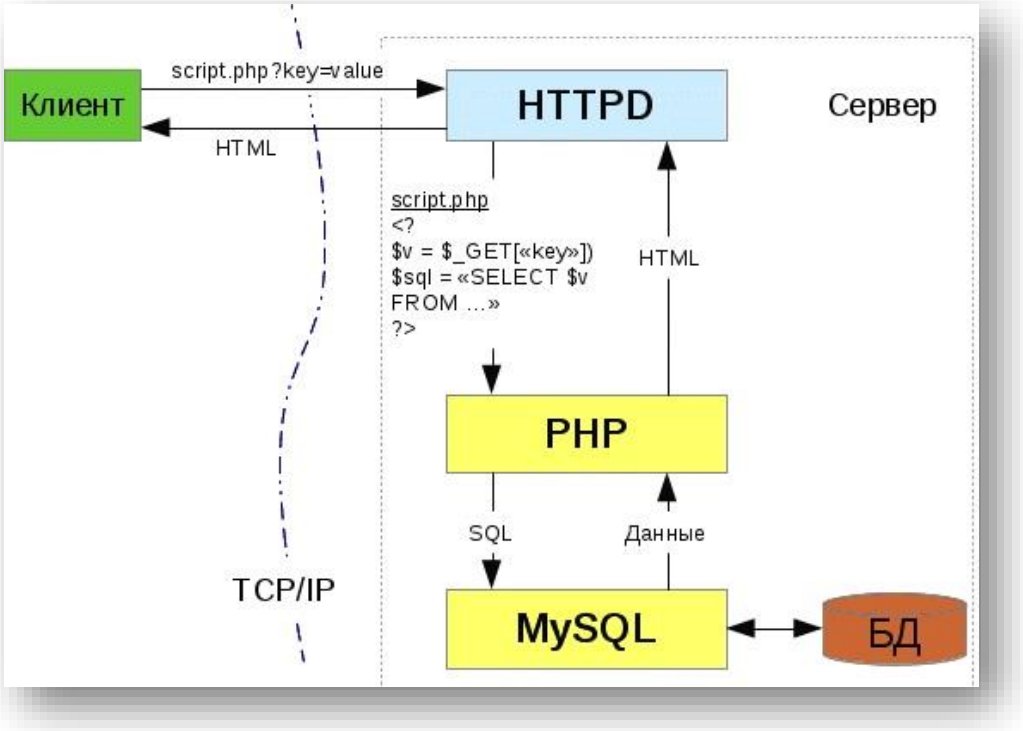

Рисунок 2.5 – Демонстрація роботи mysql

2.3.2 Огляд PowerDesigner

PowerDesigner – це засіб для сумісного моделювання в підприємстві, розроблений компанією Sybase і наразі належить SAP AG. PowerDesigner працює як рідна програма під Microsoft Windows і через плагін під Eclipse. PowerDesigner надає змогу проектувати програмне забезпечення з використанням архітектури, керованої моделі. PowerDesigner зберігає моделі у файлах з різними розширеннями, такими як .bpm, .cdm і .pdm.

Внутрішня структура файлу може бути у форматі XML або стиснутому двійковому форматі. PowerDesigner також може зберігати моделі в сховищах бази даних.

Станом на 2002 рік PowerDesigner займав 39% ринку серед інструментів моделювання даних. Ціна на PowerDesigner вербуються від 3000 до 7500 доларів на місце программіста.

PowerDesigner підтримує:

– моделювання бізнес-процесів із підтримкою BPMN;

– генерування коду на мовах: Java, Hibernate, C#, VB.NET, EJB3, NHibernate, JSF, WinForm;

– моделювання даних;

– моделювання сховища даних;

– формування звіту;

– репозиторій;

– фналіз вимог;

– моделювання об'єктів.

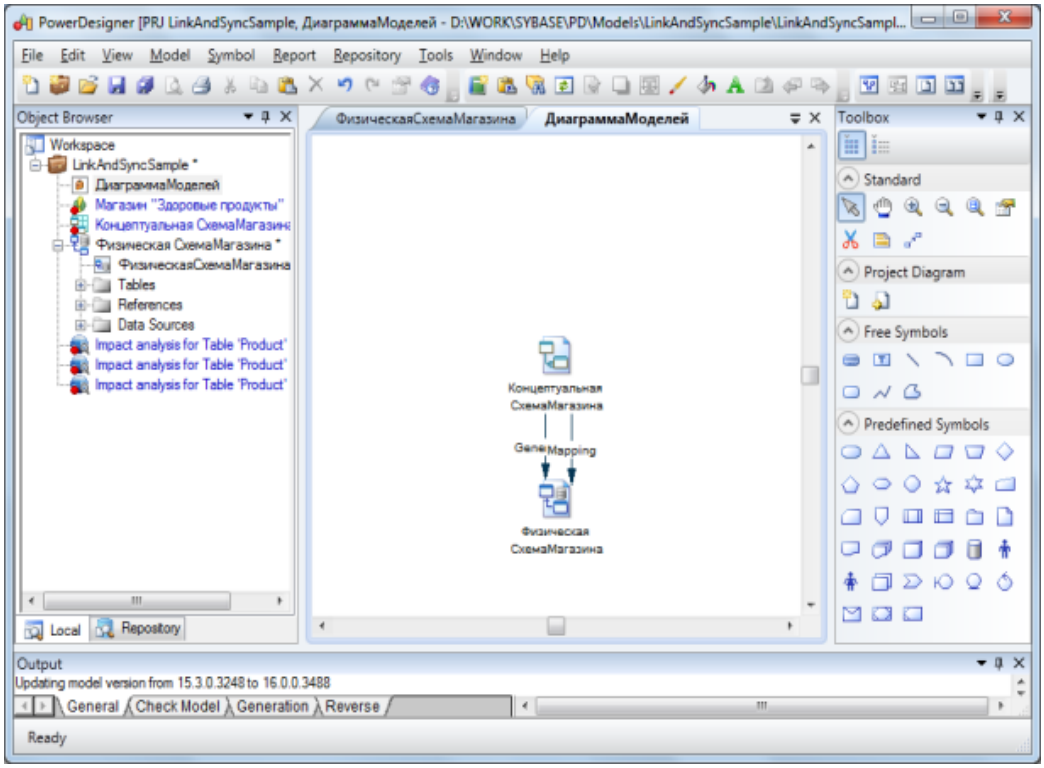

Рисунок 2.6 – Зовнішній вигляд PowerDesigner

2.3.3 Використання Apache Maven

Apache Maven – це програмний засіб автоматизації проектів, який спочатку використовувався для проектів Java, тепер використовується для керування та створення програм.

Створений він був у 2002 році Джейсоном ван Зілом. Він повністю відрізняється від Apache Ant за принципом роботи, виглядає простіше з точки зору налаштувань збірки, і надається у форматі XML. Файл XML описує проект, його порядок збірки, папки, підключення до зовнішніх модулів та необхідні плагіни. На сервері розміщено сервер із додатковими модулями та бібліотеками доповнень. Maven раніше був частиною проекту Jakarta.

Призначення та особливості Створення проекту Java рівня «Привіт, світ!» Ви можете використовувати командний рядок. Але чим складніше програмне забезпечення, яке розробляється, і чим більше використовується сторонніх бібліотек і ресурсів, тим складнішою буде команда компіляції. Maven прагне полегшити цю роботу. Однією з головних особливостей фреймворку є декларативний опис проекту. Це означає, що розробникам не потрібно зосереджуватися на кожному аспекті збірки – усі необхідні параметри налаштовані за замовчуванням. Вам потрібно внести зміни лише в тій мірі, в якій програміст хоче відхилитися від значень за замовчуванням.

Ще однією перевагою цього проекту є гнучке управління залежностями. Maven може завантажувати сторонні бібліотеки в свій локальний репозиторій, вибирати потрібну версію пакета, обробляти транспортні залежності. Maven забезпечує підтримку збірки не лише шляхом сортування файлів у цьому репозиторії, а й шляхом завантаження артефактів у кінці збірки. Локальний кеш завантажених артефактів діє як основний засіб синхронізації виводу проекту в локальній системі.

2.3.4 Огляд JavaFx

JavaFX – це платформа та набір інструментів для створення багатих Інтернет-додатків, здатних завантажувати медіа та вміст. Sun Microsystems дебютувала в травні 2007 року на Міжнародній конференції розробників JavaOne. JavaFX включає в себе набір утиліт, які дозволяють веб-розробникам і дизайнерам швидко створювати та надавати передові веб-додатки для настільних комп'ютерів, мобільних пристроїв, телевізорів та інших платформ. JavaFX складається з JavaFX Script і JavaFX Mobile. Починаючи з JavaFX версії 2.0, можна створювати програми JavaFX, повністю написані на Java. Для розробка відкрито велику кількість графічних та мультимедійних API які значно спрощують створення візуальних програм.

2.3.5 Огляд Apache Tomcat

Apache Tomcat – це контейнер сервлетів, розроблений компанією Apache Software Foundation. Повністю написаний мовою програмування Java, він реалізує сервлет Sun Microsystems і специфікації Java Server Pages, які є стандартами для розробки веб-додатків на Java.

Члени ASF та незалежні розробники волонтери підтримують розробку та обновлення Tomcat. Користувачі мають вільний доступ до вихідного коду Tomcat за ліцензією Apache. Перша версія Tomcat була 3.0.x.

Для того, щоб оптимізувати доступ перед Tomcat часто розміщують передній проксі сервер. Сам Tomcat написаний на Java і є ресурсомістким, тому найкраще полегшити його роботу з клієнтами, які в кінцевому підсумку працюють повільніше.

2.3.6 Огляд Log4j

Log4j – це бібліотека ведення журналів додатків Java, яка є частиною проекту журналу Apache. Log4j спочатку був розроблений як частина проекту Apache Jakarta, який відповідає за всі проекти Apache Java, але з тих пір став окремим і дуже популярним проектом ведення журналів. Це одна із найшвидших, найбезпечніших та гнучких структур журналів (API), вона написана на мові Java і поширюється під ліцензією Apache Software License. log4j – популярний пакет журналів, написаний на Java. log4j було перенесено на C, C++, C#, Perl, Python, Ruby та Eiffel.

log4 – являється легким у налаштуванні за допомогою зовнішніх файлів конфігурації. Він досліджує процес ведення журналу на основі пріоритету та надає механізми для переспрямування інформації журналу на різні місця призначення, такі як база даних, файл, консоль, системний журнал UNIX тощо.

2.3.7 Огляд Bootstrap

Bootstrap – це набір інструментів для створення веб-сайтів та веб-додатків, він є абсолютно безкоштовним, та включає в себе шаблони CSS та HTML для різних кнопок, таблиць, форм і інших компонентів інтерфейсу, а також розширюється з допомогою JavaScript. Це спрощує розробку динамічних веб-сайтів і додатків. Репозиторії з цією структурою є одними з найпопулярніших на GitHub. Серед них його використовують NASA та MSNBC.

Bootstrap – це фреймворк, розроблений Марком Отто та Джейкобом Торнтоном для забезпечення одноманітності інструментів у Twitter. До Bootstrap інтерфейси розроблялися з використанням різних бібліотек, що призводило до невідповідностей і складного обслуговування.

2.3.7 Огляд JSP

JSP – це технологія, яка дає змогу веб-розробникам динамічно створювати та розгортати HTML, XML та інші веб-сторінки. Робота над цією технологією розпочалась відносно недавно в 1997 році. Чуть пізніше було включено в Java EE 2012 – програмну платформу для програмування веб-додатків. Ця технологія дозволяє вставляти Java-код у статичний вміст сторінки. Бібліотека тегів jsp також можна застосовувати для встановлення їх у сторінки jsp. Сторінки компілюються компілятором jsp у сервлеті, які є класами Java, і запускаються на сервері. Сервлети також можуть бути написані розробниками без використання сторінок jsp. Ці технології доповнюють одна одну. JSP є високопродуктивною технологією, оскільки весь код сторінки перекладається в код Java сервлета за допомогою компілятора сторінок jsp, а потім компілюється у байт-код віртуальної машини Java.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

Цей проект містить в собі комплекс папок, які мають певну ієрархію, кожна папка містить в собі певні ресурси і програмний код. Всі ресурси відповідно зберігаються у папці під назвою resources, а програмний код у коренній папочці src.

Веб-контейнером було обрано популярну реалізацію специфікацій сервлетів Apache Tomcat 9, що дозволяє нам запускати створений додаток.

Клієнтська частина реалізована з використанням каскадних таблиць стилів CSS та мови розмітки HTML5 та фреймворку AngularJS.

3.1 Функціонал клієнтської частини

Шляхом надсилання клієнтом HTTP запитів відбувається комунікація клієнтської частини. Дизайн був виконаний у мінімалістичному стилі та не є перенасичений компонентами, це і робить його дуже простим у використанні, проте дуже функціональним.

Ось наприклад головна сторінка яка зустрічає користувача при відвідуванні сайту містить два поля для входу на свій вже створений акаунт, та кнопочку для створення акаунту, якщо такого у нього немає (рис. 3.1).

Ввівши свою електронну почту та пароль від свого уже створеного акаунту, та нажавши кнопочку Sign in, користувач потрапляє на головну сторінку сайту з доступними тарифами від інтернет провайдера, які той заздалегіть розмістив на сайті за допомогою адмінської панелі, тарифи. Доступні користувачу тарифи продемонстровано на рисунку 3.2.

Користувач в свою чергу може обрати любий із доступних йому тарифів на цій сторінці, після чого працівнику IT компанії прийде повідомлення, де в нього є змога прийняти або відхилити запрошений користувачем тарифний план (рис. 3.9).

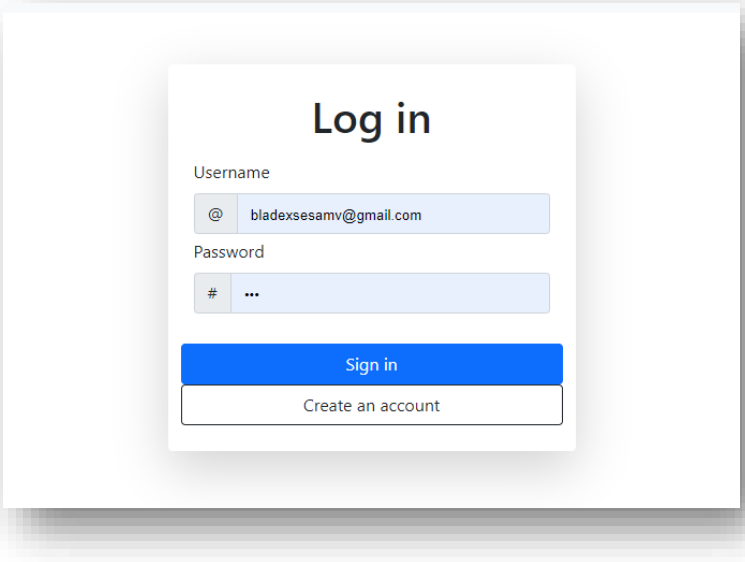

Рисунок 3.1 – Інтерфейс сторінки реєстрації

Така конструкція та логіка сайту дає змогу не заблукати користувачу на ньому, тому що все просто та зрозуміло. Даний підхід та мінімалістичний вигляд сайту знижує затримку при завантаженні сторінки, так як користувачу прийдеться завантажувати значно менший обсяг інформації та інтерфейсу.

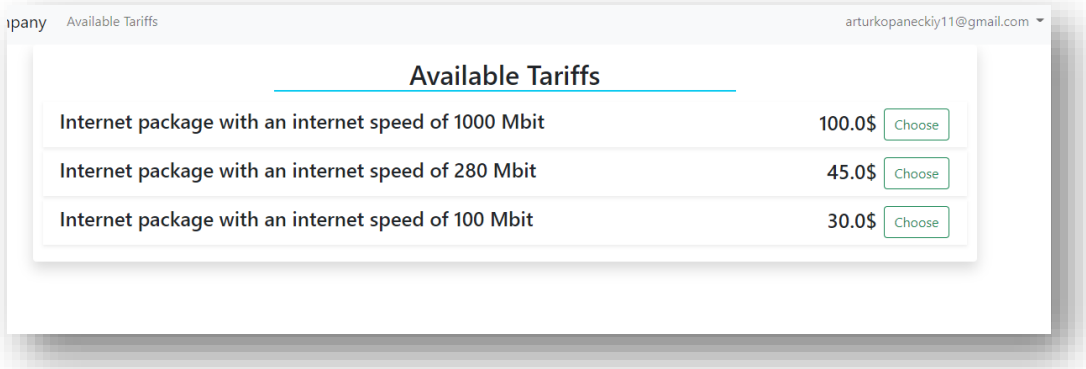

Рисунок 3.2 – Доступні користувачу тарифи у вкладці Available Tariffs

Базовий функціонал клієнтської частини сайту включає в себе:

– можливість перегляду всією вказаної інформації при реєстрації, а також її зміни при необхідності;

– обрання, зміна та відключення від тарифу;

– аерегляд схваленого тарифу адміністратором сайту, або заявку на перехід на новий тариф якщо він ще не був схвалений;

– змога зміни паролю, якщо користувачу це знадобиться.

Все це продемонстровано на рисунках 3.3 та 3.4

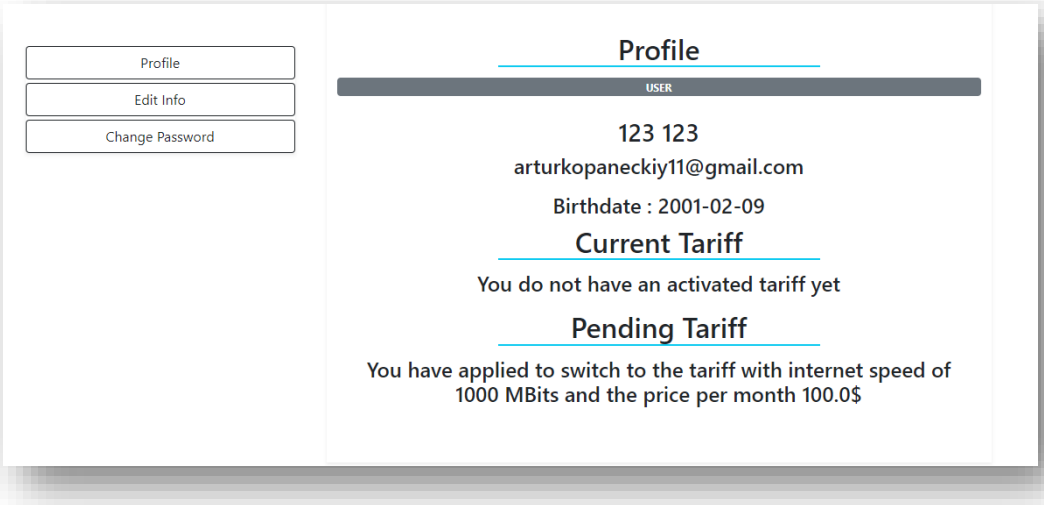

Рисунок 3.3 – Профіль користувача у вкладці Profile

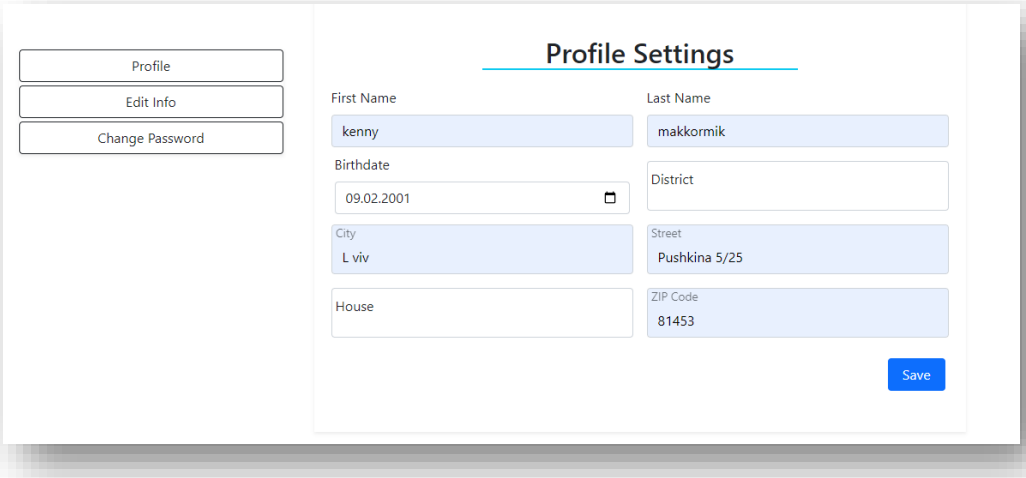

Рисунок 3.4 – Зміна користувачем інформації про себе

3.2 Функціонал адмінської частини

Сайт зі сторони працівника має такий же інтерфейс як із сторони клієнта, зроблено це для зменшення витрати часу на побудову нового інтерфейсу та зменшення витрат коштів на її обнову.

Щоб мати повний доступ до адміністрування сайту, потрібно зайти на нього з акаунту адміністратора, для цього в базі данних itcompany у полі під назвою role в таблиці клієнта потрібно виставити цифру – 1, після чого звичайному акаунту відкриється повний доступ до сайту.

При заході на сайт із такого акаунта, працівника також зустрічає сторінка із тарифами, проте працівник на відміну від клієнта має змогу редагувати тарифні плани. Редагування відбувається за допомогою натискання відповідних кнопок на панелі тарифів, як показано на рисунку 3.5.. Після натискання кнопочки під назвою Edit відбувається перехід на нову сторінку, де працівник може вказати нову ціну та швидкість інтернету для даного тарифного плану, це ж меню відкривається і при нажиманні кнопки New (рис. 3.6).

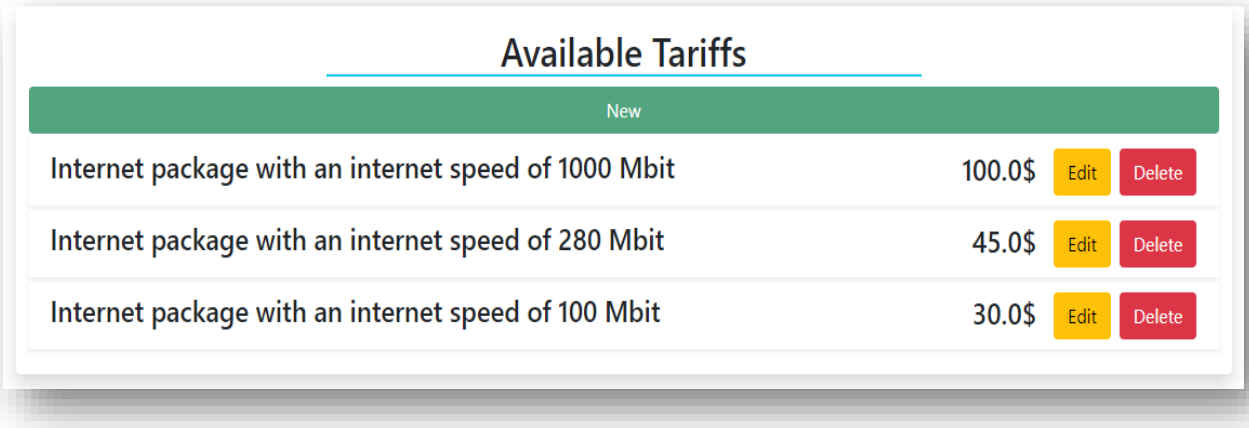

Рисунок 3.5 – Меню редагування тарифних планів

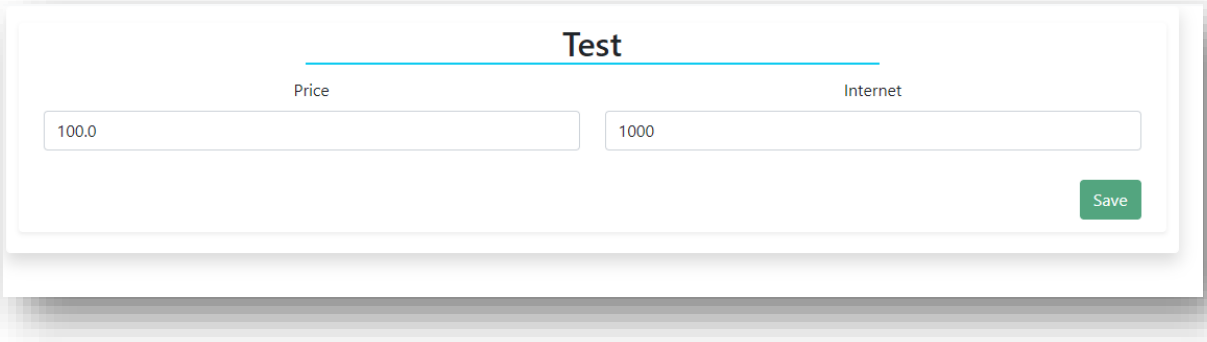

Рисунок 3.6 – Зміна працівником даних тарифу

Також працівника IT компанії зустрічають дві нові вкладки на шапці сайту під нозвою Users та Pandding Tariffs.

Users – демонструє усіх зареєстрованих клієнтів сайту та наступну інформацію про них: електронний адрес, ім’я, фамілія та дата народження.

Pandding Tariffs – показує тарифи які очікують підтвердження від працівника та детальну інформацію: ім’я та фамілія клієнта який надіслав запит, назва дії яку він в свою чергу хоче зробити, швидкість та ціна тарифу який обрав клієнт і дата звернення користувача.

Кнопка Edit у вкладці Users дає змогу працівнику (адміністратору) змінити всю інформацію про обраного клієнта, що той вказав при реєстрації та змінити роль клієнта із звичайного користувача на працівника. Також присутня кнопка Ban, для запису користувача у чорний список, якщо той в свою чергу порушив правила компанії або не оплатив завчасно ціну свого тарифного плану. Все це зображено на рисунках 3.7, 3.8.

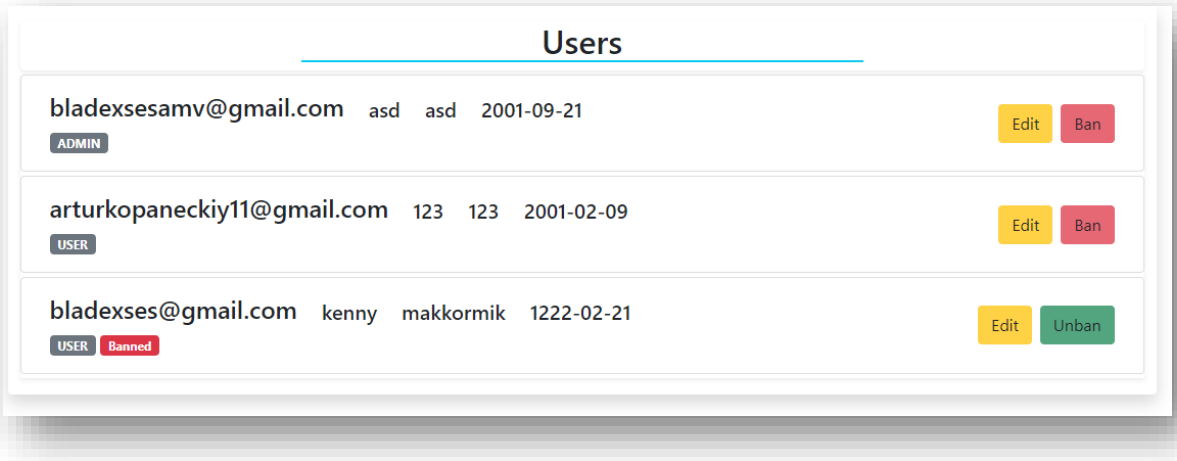

Рисунок 3.7 – Вкладка Users

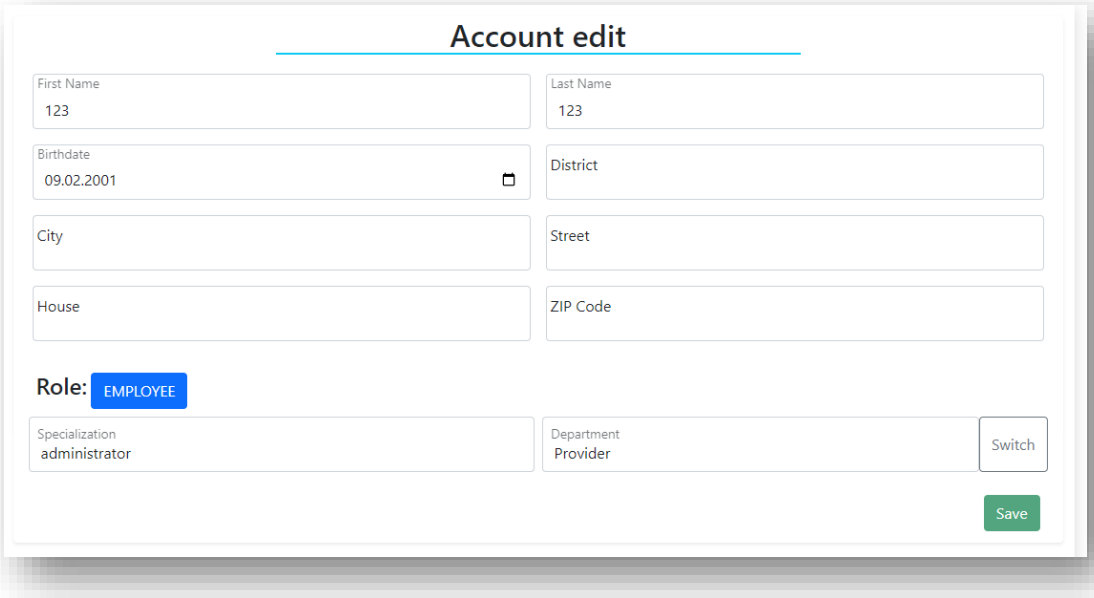

Рисунок 3.8 – Сторінка редагування даних користувача

В меню Pandding Tariffs працівник може віхилити або прийняти запит користувача на обраний тариф, якщо той йому поступив (рис. 3.9)

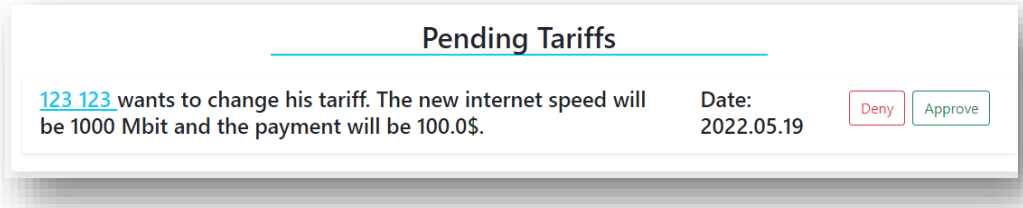

Рисунок 3.9 – Сторінка очікування запитів на тариф

3.3 Алгоритм реалізації клієнтської бази

У кожному застосунку подібного типу повинна бути реалізована клієнтська база із всіма потрібними таблицями та полями в них. Для даного проекту у MySql було створено базу даних під назвою “itcompany”, а у ній п’ять основних таблиць: accounts, addresses, departments, pending_tariff, tariffs.

```
/*==============================================================
*/
/* DBMS name:      MySQL 5.0
*/
/* Created on:     02.06.2022 16:44:23
*/
/*==============================================================
*/
drop database if exists itcompany;
create database itcompany;
use itcompany;

drop table if exists accounts;

drop table if exists addresses;

drop table if exists departments;

drop table if exists pending_tariff;

drop table if exists tariffs;
```

Рисунок 3.10 – Лістинг коду запиту до MySql для створення таблиць

Даний код є універсальним через те, є можливість редагувати змінні та назви полів таблиць. Цей код потрібно виконати всього лиш один раз при першому запуску застосунку, після чого буде створена база даних із всіма потрібними таблицями та полями, повторне її сторення не потрібне але і ні до якої проблеми це не призведе.

```
*/
/*==============================================================
*/
create table accounts
(
   accountId            int not null auto_increment,
   tariffId             int,
   addressId            int,
   departmentId         int,
   email                varchar(50) unique not null,
   password             varchar(50) not null,
   role                 smallint not null,
   firstName            varchar(50),
   lastName             varchar(50),
   surname              varchar(50),
   birthdate            date,
   register_date        date not null,
   specialization       varchar(100),
   banned               bool default False,
   primary key (accountId)
);
```

Рисунок 3.11 – Лістинг коду створення першої таблиці accounts для користувачів

Цей код є дуже зрозумілим, а назва полів дає змогу зрозуміти яким чином буде використовуватись данна таблиця.

Поля accountId, addressId, email, password, firstName, lastName, surname, birthdate, register_date, створюються і заповнюються одразу як тільки користувач зареєструється на сайті, окрім полів tariffId, departmentId, role, specialization, banned, вони будуть заповнені при взаємодії користувача із сервером або адміністратором. При зміні поля role між значеннями 0 та 1 буде змінюватись і роль користувача на сайті, між звичайним користувачем та адміністратором. DepartmentId та specialization в свою чергу відповідають за

призначення працівнику певного робочого місця, будь то технічний супорт або інженер тестувальник.

```
/*==============================================================
*/
/* Table: addresses
*/
/*==============================================================
*/
create table addresses
(
   addressId            int not null auto_increment,
   departmentId         int,
   accountId            int,
   District             char(100),
   City                 char(100),
   Street               varchar(100),
   HouseNumber          smallint,
   postalCode           int,
   primary key (addressId)
);
```

Рисунок 3.12 – Лістинг коду створення таблиці для адрес користувачів

Адресна інформація користувачів не заповнюється як тільки той зареєструється на сайті, проте в нього є змога заповнити її пізніше у спеціально відділеній вкладці Profile settings, або дочекатись поки адміністратор заповнить її власноруч.

```
/*==============================================================
*/
/* Table: departments
*/
/*==============================================================
*/
create table departments
(
   departmentId         int not null auto_increment,
   addressId            int,
   name                 varchar(256),
   primary key (departmentId)
);
```

Рисунок 3.13 – Лістинг коду створення таблиці робочих відділів для працівників

```
/* Table: pending_tariff
*//*==========================================================


create table pending_tariff
(
   tariffId             int not null,
   accountId            int not null,
   submission_time            timestamp,
   primary key (accountId)
);
```

Рисунок 3.14 – Лістинг коду створення таблиці очікування тарифів на підтвердження

В коді рисунку 3.14 заповнюються 3 поля після того як працівник підтвердив запит клієнта на підключення тарифного плану. TariffId відповідає за номер тарифного плану на рисунку 3.15, accountId за індифікатор користувача у лістингу 3.12 і submission_time за час коли був прийнятий запит.

```
/*==========================================================
*/
/* Table: tariffs
*/
/*==========================================================
*/
create table tariffs
(
   tariffId             int not null auto_increment,
   price                numeric(8,0) not null,
   internet_speed       smallint not null,
   primary key (tariffId)
);
```

Рисунок 3.15 – Лістинг коду створення таблиці тарифних планів

З цими трьома таблицями взаємодіє працівник (адміністратор).

На рисунку 3.14 продемонстровано створення таблиці для працівників, що дозволяє назначати їх на різні посади та робочі місця за допомогою меню керування на сайті, під час чого можна заповнити поля наступною інформацією: відділ в якому працює працівник, його індифікатор та назва посади.

У лістингу 3.15 створюється сам тарифний план, доступ до створення має тільки користувач з адмін правами, тарифний план може містити в собі

порядковий номер або індифікатор який створюється автоматично, ціна значення якої не може становити 0, та швидкість інтернет підключення яке.

3.4 Використання Log4g в програмному коді

Створення логера за допомогою модуля log3g є доволі таки простим. Для зрозуміння та більше легшого пояснення я продемонструю його код на рисунку 3.16

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration name="testConfiguration"
               status="debug">
    <Properties>
        <Property
name="logdir">D:\DEV\IdeaProjects\TestingApp\logs</Property>
        <Property name="layout">%d [%t] %-5p %c{2} - %m%n</Property>
    </Properties>
    <Appenders>
        <Console name="STDOUT" target = "SYSTEM_OUT">
            <ThresholdFilter level="trace" onMatch="ACCEPT"
onMismatch="DENY"/>
            <PatternLayout pattern="${layout}"/>
        </Console>
        <File name="RolFile"
              fileName="${logdir}/localhost.log"
              filePattern="${logdir}/localhost.%d{yyyy-MM-dd}-%i.log">
            <PatternLayout pattern="${layout}"/>
            <ThresholdFilter level="debug" onMatch="ACCEPT"
onMismatch="DENY"/>
            <Policies>
                <TimeBasedTriggeringPolicy />
                <SizeBasedTriggeringPolicy size="1 MB" />
            </Policies>
            <DefaultRolloverStrategy max="10" />
        </File>
    </Appenders>
    <Loggers>
        <Root level="trace" additivity="false">
            <AppenderRef ref="STDOUT" />
            <AppenderRef ref="RolFile" />
        </Root>
    </Loggers>
</Configuration>
```

Рисунок 3.16 – Лістинг коду логера log3j xml

Цей конфігураційний файл говорить про те, що потрібно виводити всі повідомлення із классів в консоль на запис їх в окремі файли для збереження

та подальшого використання. При чому записуватись будут всі повідомлення аж до рівня Debug, хоча для більш точно логування можна вказати конкретні рівні у полі root.

Атрибутам name в елементі logger було задано ім’я файлу та классу, для яких виставляється окремі рівні логування appender.

При взаэмодії всіх користувачів із сайтом всі дії будуть записуватись та відображатись в наступному вигляді: точна дата виконаної дії в плоть до мілісекунд, рівень логування, файл або назва класу до якого відноситься дія, та саме повідомлення на місці якого може бути помилка або заздалегіть зазначене відповідь на дію користувача.

Прикладом може слугувати відповідь на запуск сервера: 20-Jun-2022 03:11:47.907 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [4015] milliseconds.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Аварії з викидом радіоактивних речовин

За даними світової статистники, що фіксується в головній службі небезпечних ситуацій, великі аварії на підприємствах та об’єктах різних типів, де лінійні розміри зон ураження досягають декілька сотень або навіть тисяч метрів, є досить рідкісними. Проте тим не менш, у світі в середньому за рік відбувається близько 2 – 3 подібних аварій.

Аварії із загибелью понад 25 осіб і числом поранених більше 100 реєструється в середньому раз на 2,5 роки.

У цілому, як вважають фахівці, спостерігається неухильне зростання кількості пропислових і енергетичних аварій, викликане, з одного боку, збільшенням кількості небезпечних об’єктів, з іншого боку, зростанням питомої щільності населення в зонах розвитку промислових і енергетичних об’єктів.

Найбільша в історії людстава радіаційна катастрофа на Чорнобильській АЕС сталась 26 квітня у 1986 році. Кілька років після катастрофи всі офіційни джерела СРСР повідомляли, що жертвами Чорнобиля стали тільки 33 людини – в основному пожежники, які брали участь в наймерших роботах. Потім почали з’являтись окремі повідомлення про те, що від променевої хвороби загинуло кілька десятків ліквідаторі, а захворіли тисячі. Про жертви серед місцевого населення не говорилось взагалі. Режим секретності з питань аварай ЧАЕС, який існував до 1991 року, не дозволяв відтворити об’єктивну картину масштабів ураження населення.

За сучасними уявленнями, аварія на ЧАЕС має серйозні наслідки пролонгованої дії, в тому числі такі, що можуть виявлятись на генетичному рівні в окремих груп персоналу АЕС, ліквідаторів і населення, яке проживає поблизу зони аварії.

У результаті вибуху четвертого реактора Чорнобильської атомної електростанції стався величезний викид радіоактивних речовин в атмосферу. Ці радіоактивні опади випали в основному в межах євро-азіатського континенту, але особливо у великих кількостях на значних територіях Білорусі, Російської Федерації та України.

За оцінками, протягом 1986–1987 рр. до ліквідації наслідків аварії було залучено понад 350000 чоловік-«ліквідаторів» з числа військовослужбовців, працівників АЕС, місцевої міліції та пожежних служб. Досить високі дози радіації отримали близько 240000 чоловік під час проведення робіт з ліквідації наслідків аварії в межах 30-кілометрової зони, виконання робіт з консервації аварійного 4-го блоку АЕС – будівництва «Саркофагу», очищення дахів АЕС, створення системи захисту водних об'єктів.

У даний час приблизно п'ять мільйонів людей проживають в районах Білорусі, Російської Федерації та України, де рівні радіоактивного забруднення грунтів цезієм перевищують 37 кБк/м2. З них приблизно 270000 людей продовжують жити в районах, які класифікувалися повноважними органами як зони посиленого контролю ,де зараження 137Cs перевищує 555 кБк/м2.

Можна припустити збільшення кількості випадків смерті від раку протягом усього життя серед осіб, які зазнали впливу радіації в результаті аварії. У зв'язку з тим, що в даний час неможливо визначити, які конкретні випадки раку були викликані радіацією, кількість таких випадків смерті можна оцінити лише статистично на основі використання інформації та проекцій, отриманих під час досліджень на людях, які вижили після вибухів атомних бомб[4, с. 202, 210].

4.2 Долікарська допомога при обмороженні

Гіпертермія – це патологічний стан організму, виникає внаслідок значного порушення терморегуляції або дії зовнішнього тепла.

Замерзання – це загальне патологічне переохолодження організму, зумовлене поступовим зниженням температури тіла під впливом охолодження дії зовнішнього середовища, при недостатній захисній терморегуляторній функції організму. В основі замерзання лежить порушення терморегуляції організму. Загальна гіпотермія викликає зниження усіх видів обміну, в результаті чого створюються умови, за яких тепловіддача значно перевищує теплоутворення.

Залежно від глибини ураження тканини розрізняються на чотири ступеня відмороження(рис.4.1):

– Перший ступінь – шкіра постраждалого блідого кольору, незначно набрякла, чутливість знижена або повністю відсутня. Підвищується чутливість до холоду, що може зберігатись 2-3 місяці і більше;

– Другий ступінь – у ділянці відмороження утворюється міхури, наповнені прозорою або білою рідиною. До цього характерні підвищення температури тіла, остуда;

– Третій ступінь – омертвіння шкіри: з’являються міхурі, наповнені рідиною темно-червоного або темно-бурого кольору як результат обмороження глибокої дерми. Характерний розвиток інтоксикації – остуда, потовиділення, значне погіршення самопочуття, апатія;

– Четвертий ступінь – поява міхурів, наповнених чорною рідиною. У постраждалого є ознаки шоку. Розвиток набряку відбувається через 1 або 2 год. Набряк, як правило, поширюється на проксимальні відділи кінцівок. Потім розвивається муміфікація, рідше – волога гангрена.

Рисунок 4.1 – Стадії обмороження

Дії при наданні першої медичної допомоги відрізняються в залежності від ступеню обмороження, стану загального охолодження організму людини, його віку та наявних хвороб.

При обмороженні постраждалого в першу чергу потрібно перенести у тепле приміщення, після чого приступити до поступового зігрівання. Застосовують пасивне зігрівання: знімаючи мокрий та весь холодний одяг, поступово обсушуючи шкіру, накривання теплими ковдрами. Цей метод ефективний при легкій гіпотермії. Обкладання постраждалого грілками або поступове занурення у теплу воду призведе до руху відносно холодної крові по всьому тілу з подальшим охолодженням і порушень у життєвоважливих органах, тому краще обійтись без цього.

При гіпотермії тяжкого ступеня, коли температура тіла нижче 30 градусів, потрібно застосувати активне зовнішнє зігрівання, що включає в себе: ковдри з підігрівом, гарячі ванни.

Розмерзання, розігрівання тканин та відновлення кровообігу повинно поширюватись у зворотньому напрямку (від центра до периферії) під дією тепла власного тіла та крові. Передчасне розігрівання тканини на периферії без відновлення кровообігу веде до їх загибелі. Тому для ушкоджених холодом тканин потрібно створити умови термоса, шляхом накладання термоізоляційних пов’язок від кисті до плечових суглобів та ступні до

кульшових суглобів, обгорнувши їх поліетиленовою плівкою, поверх плівки накласти товстий шар вати чи шерстяних тканин і добре забинтувати марлевими бинтами. При цьому забезпечивши постраждалому часте пиття теплої води та якнайшвидшу госпіталізацію до найближчого лікувального закладу[6, с. 10, 24].

При виникненні таких ситуацій потрібно запам’ятати, що:

– не можна дозволяти постраждалому з обмороженими ногами або пальцями йти, його потрібно переносити на носилках;

– не можна відігрівати тканини, якщо є можливість їх повторного обмороження;

– не потрібно відігрівати постраждалого біля вогню, або втирати сніг у шкіру.

Щорічно до лікувальних закладів України надходять постраждалі з різними стадіями обмороження. За спостереженнями лікарів кількість таких хворих збільшується одразу в декілька разів у період різких похолодань. Щоб уникнути переохолодження й обмороження, фахівці рекомендують більше про це дізнаватись та дотримуватись основних правил поведінки в умовах низьких температур[7, с. 148, 166].

ВИСНОВКИ

Під час виконання дипломного проекту, отримано веб-сайт, який готовий для використання користувачами. Розробка задовольняє всі вимоги, які поставлені на етапі постановки завдання.

Було проведено дослідження компонентів програмного середовища Intellij IDEA, яке використовуваллсь при створенні застосунку.

Клієнт-серверна інформаційна система представляє собою написання демонстраційного веб-сайту який має корпоративне направлення, що маже залучити потенційних клієнтів, які находяться в пошуку найкращого провайдера з відносно низькими цінами та швидким інтернетом.

Перевагою цього веб-сайту є те, що він досить простий у використанні та має обширний функціонал. Але як і люба програма та веб-сайт може бути покращена за рахунок допрацювання.

Написання дипломної роботи дало багато досвіду в розробці програмного продукту, вивчення нового матеріалу, вдосконалення вміння та навичок роботи із програмуванням в середовишы Intellij IDEA.

Використання засобів програмування дало змогу справитись з поставленою задачею. У середовищі програмування Intellij IDEA є всі необхідні інструменти та компоненти для створння повноцінної програми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Phillip T.Rawles, Julie R.Mariga – Wiley December 17, 1998 668 page – ISBN – 978-0471296546
2. Кириченко А.В., Дубовик Е.В., web на практиці, CSS, HTML. м.Київ 2021. – 432с. ISBN 978-5-94387-271-6.
3. С. М. Плохій, Чорнобиль, Історія ядерної катастрофи 2019. - 396 с. – ISBN 617-7013-99-9
4. Плачкова С.Г., Плачков І.В. Енергетика: історія, сучастність і майбутнє – м. Київ 2013 р. с. 301– ISBN 978-966-8163-18-0
5. Віктор Бар’яхтяр., Катастрофи на АЕС та атомна енергетика. – м.Київ, 2014. – 36 с. – ISSN 1819-7329.
6. В.О. Крилюк, Домедична допомога Медичний посібник – 2014. – 84 с. – ISSN 1681-2751
7. Т. В. Петриченко Перша медична допомога – м.Київ, 2015. – 272 с. – ISBN 978-617-505-359-1
8. Віктор Оліфер, Наталія Оліфер., Комп'ютерні мережі Принципи, технології, протоколи. – м.Київ, 2016. – 992 с. – ISBN 978-5-496-01967-5;
9. Берон Шварц MySql по максимуму, 3-е изд. – м.Київ. 2018. 864с. – ISBN 978-5-4461-0696-7
10. Джозеф Албахарі, Бен Албахарі. Java. Кишеньковий довідник. – м.Львів. 2017. – 224 с. – ISBN 978-5-9909446-1-6
11. ЦСО. Тарифи на доступ до Інтернет. [Інтернет ресурс] – Режим доступу: http://cso.com.ua/index.php?text=internetprice (Дата звернення 18.04.2022 року);
12. IntelliJ IDEA 2019. [Інтернет ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/idea/download/other.html/ (Дата звернення 18.04.2022 року);

13. Початок роботи з IntelliJ IDEA. [Інтернет ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/idea/ (Дата звернення 18.04.2022 року);

14. Enterprise System Architectures Mark Goodyear – Published September 28, 1999 002 pages. – ISBN 9780849398360

15. Tom Laszwski, Prakash Nauduri Service Enablement of Client/Server Application – Published 26 September 2011 238 pages. – ISBN 9980646398378

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

«Затверджую»

завідувач кафедри КС

_____ Осухівська Г.М.

"____ "_____ 2022 р.

КЛІЄНТ-СЕРВЕРНА ІНФОРМАЦІЙНА СИСТЕМА ІТ КОМПАНІЇ

ТЕХНІЧНЕ ЗАВДАННЯ

на __6__ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»Спеціальність

123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»

Керівник дипломного проєкту

_____ к.т.н., Лупенко С. А.

«____»_____ 2022 р.

«ВИКОНАВЕЦЬ»

Студент групи СІс-43

_____ Копанецький А.Л.

«____»_____ 2022 р

Тернопіль 2022

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: клієнт-серверна інформаційна система ІТ компанії

Умовне позначення кваліфікаційної роботи: КС КРБ 123.393.00.00

1.2 Виконавень

Студент групи СІс-43, факультету комп'ютерно-інформаційних систем і програмної інженерії, кафедри комп'ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Копанецький Артур Любомирович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (No 4/7-180 від 23.03.2022 р.)

1.4 Планові терміни початку та завершнення роботи

Плановий термін початку виконання кваліфікаційної роботи – 24.03.2022р. Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2022 р.

1.5 Порядок оформлення та пред'явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД,ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

Призначенням програмного продукту є представлення клієнтам можливості вибору найзручніших тарифів для інтернет. Веб застосунок базується на клієнтській частині, сервері та базі даних.

До основних задач, які повинен виконувати веб застосунок це:

- Реєстрація в системі;
- Створення профілю користувача;
- Збереження його даних в базі даних;
- Перегляд інших профілів;
- Редагування профілю;
- Вибір тарифного плану;
- Одобрення тарифного плану працівником.

2.1 Мета створення платформи

Метою створення платформи є розробка застосунку на базі клієнт серверної архітектури, з використанням біліотек, фреймворків та сервлетів. Веб застосунок розроблений у вигляді окремого об’єкту та забезпечує можливість управління клієнтською базою, здійснення вибору та переключенням між тарифними планами ті їх редагуванням.

2.2 Характеристики об’єкту

2.3.1 Основні задачі та функції об’єкту

Основною задачею даного застосунку є можливість авторизації користувача в системі, створення та заповнення профілю користувача, його редагування власноруч або за допомогою працівника, вибір підходящого тарифного плану на інтернет, які були заздалегіть виставлені працівниками.

Щоб досягнути цих завдань необхідно необхідно вирішити певні питання, які полягають у взаємозвязку клієнтської, серверної частин та бази даних.

Доступ до даних користувачів та сервера повинен бути у зареєстрованих користувачів під акаунтом адміністратора для забезпечення безпеки даних.

3 Вимоги до системи

3.1 Вимоги до платформи

Під час відправлення запиту від клієнта на сервер, повинна здійснюватись відповідь в залежності від обраної дії користувача або адміністратора.

3.1.1 Вимоги до структури та функціонування системи

Структура сайту складається з наступного:

- Клієнт сервер;
- База даних;
- Веб сервер;
- Створення акаунту;
- Збереження даних користувачів;
- Управління даними клієнтів адміністратором;
- Редагування тарифних планів для користувачів;

– Обрання тарифного плану користувачем;

– Схвалення або відхилення адміністратором обраного тарифу користувачем.

3.1.2 Вимоги по діагностуванню

Діагностика та профілактика програмних і апаратних частин сайту відбувається за допомогою системного адміністратора.

3.1.3 Перспективи розвитку програмного продукту

До перспектив належать:

– Реалізація захищенного доступу до особистої інформації

– Оптимізація та розширення бази даних

– Можлива адаптація для мобільних пристроїв та реалізаця в комп’ютерну програму;

– Розширення функціоналу адміністрування та добавлення нових видів тарифних планів.

3.1.4 Вимоги до надійності веб-застосунку

На сайті повинна бути реалізована веріфікована авторизація користувачів. До цього потрібно зберігати дані користувачів окремо від остальних, та надавити доступ до них тільки адміністраторам та особам яким вона належить.

3.1.5 Вимоги до апаратного забезпечення

Мінімальні вимоги що до апаратного забезпечення сервера:

– процесор: Intel Xeon e5 2440 2.4 Ghz 6 core;
– графічна карта: Nvidia GeForce 740;
– оперативна пам’ять об’ємом в 32 GB DDR4 2333;
– накопичувач: 4TB Gb HHD;

Мінімальні вимого що до апаратного забезпечення клієнта:

– процесор: Intel Pentium Gold;
– графічна карта: AMD hd 4670;
– оперативна пам’ять об’ємом 2000 mb.

3.1.6 Вимоги до програмного забезпечення

Програмне забезпечення веб-сервера: Linux, Git, Ubuntu, MySql, Apache tomcat;

Програмне забезпечення для клієнт сервера: Linux, Ubuntu, Git, JSP, Bootstrap, Maven;

Використаний редактор коду Intell IDEA.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ з/п	Назва ступенів роботи	Термін виконання етапів роботи
1.	Ознайомлення з завданням до кваліфікаційної роботи	
2.	Провести огляд літературних джерел по темі випускної бакалаврської роботи	
3.	Провести порівняльну характеристику програмних продуктів	
4.	Розробити функціональну схему роботи об'єкта проектування	
	Розробка моделі бази даних	
5.	Практична реалізація об’єкта проектування	
6.	Описати та розробити організаційні заходи спрямовані на поліпшення стану охорони праці	
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	
8.	Виконання завдання до підрозділу «Основи хорони	
9.	Оформлення кваліфікаційної роботи	
10.	Нормоконтроль	
11.	Перевірка на плагіат	
12.	Попередній захист кваліфікаційної роботи	
13.	Захист кваліфікаційної роботи	

Додаток Б

Лістинг шаблона логера

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration name="testConfiguration"
               status="debug">
    <Properties>
        <Property
name="logdir">D:\DEV\IdeaProjects\TestingApp\logs</Property>
        <Property name="layout">%d [%t] %-5p %c{2} -
%m%n</Property>
    </Properties>
    <Appenders>
        <Console name="STDOUT" target = "SYSTEM_OUT">
            <ThresholdFilter level="trace" onMatch="ACCEPT"
onMismatch="DENY"/>
            <PatternLayout pattern="${layout}"/>
        </Console>
        <File name="RolFile"
              fileName="${logdir}/localhost.log"
              filePattern="${logdir}/localhost.%d{yyyy-MM-dd}-
%i.log">
            <PatternLayout pattern="${layout}"/>
            <ThresholdFilter level="debug" onMatch="ACCEPT"
onMismatch="DENY"/>
            <Policies>
                <TimeBasedTriggeringPolicy />
                <SizeBasedTriggeringPolicy size="1 MB" />
            </Policies>
            <DefaultRolloverStrategy max="10" />
        </File>
    </Appenders>
    <Loggers>
        <Root level="trace" additivity="false">
            <AppenderRef ref="STDOUT" />
            <AppenderRef ref="RolFile" />
        </Root>
    </Loggers>
</Configuration>
```

Додаток В

Лістинг шаблона створення бази даних

```
/*==============================================================
*/
/* DBMS name:      MySQL 5.0
*/
/* Created on:     02.06.2022 16:44:23
*/
/*==============================================================
*/
drop database if exists itcompany;
create database itcompany;
use itcompany;

drop table if exists accounts;

drop table if exists addresses;

drop table if exists departments;

drop table if exists pending_tariff;

drop table if exists tariffs;

/*==============================================================
*/
/* Table: accounts
*/
/*==============================================================
*/
create table accounts
(
   accountId            int not null auto_increment,
   tariffId             int,
   addressId            int,
   departmentId         int,
   email                varchar(50) unique not null,
   password             varchar(50) not null,
   role                 smallint not null,
   firstName            varchar(50),
   lastName             varchar(50),
   surname              varchar(50),
   birthdate            date,
   register_date        date not null,
   specialization       varchar(100),
   banned               bool default False,
   primary key (accountId)
);
```

```
/*==============================================================
*/
/* Table: addresses
*/
/*==============================================================
*/
create table addresses
(
   addressId            int not null auto_increment,
   departmentId         int,
   accountId            int,
   District             char(100),
   City                 char(100),
   Street               varchar(100),
   HouseNumber          smallint,
   postalCode           int,
   primary key (addressId)
);

/*==============================================================
*/
/* Table: departments
*/
/*==============================================================
*/
create table departments
(
   departmentId         int not null auto_increment,
   addressId            int,
   name                 varchar(256),
   primary key (departmentId)
);

/*==============================================================
*/
/* Table: pending_tariff
*/
/*==============================================================
*/
create table pending_tariff
(
   tariffId             int not null,
   accountId            int not null,
   submission_time            timestamp,
   primary key (accountId)
);

/*==============================================================
*/
/* Table: tariffs
*/
/*==============================================================
```

```
*/
create table tariffs
(
   tariffId             int not null auto_increment,
   price                numeric(8,0) not null,
   internet_speed       smallint not null,
   primary key (tariffId)
);

alter table accounts add constraint "FK_in one department works
many employee" foreign key (departmentId)
      references departments (departmentId) on delete set null
on update cascade;

alter table accounts add constraint "FK_one account have one
address" foreign key (addressId)
      references addresses (addressId) on delete set null on
update cascade;

alter table accounts add constraint "FK_one tarrif can be in
many people" foreign key (tariffId)
      references tariffs (tariffId) on delete set null on update
cascade;

alter table addresses add constraint "FK_One Department have One
Address" foreign key (departmentId)
      references departments (departmentId) on delete cascade on
update cascade;

alter table addresses add constraint "FK_one account have one
address2" foreign key (accountId)
      references accounts (accountId) on delete cascade on
update cascade;

alter table departments add constraint "FK_One Department have
One Address2" foreign key (addressId)
      references addresses (addressId) on delete set null on
update cascade;

alter table pending_tariff add constraint FK_pending_tariff
foreign key (tariffId)
      references tariffs (tariffId) on delete cascade on update
cascade;

alter table pending_tariff add constraint FK_pending_tariff2
foreign key (accountId)
      references accounts (accountId) on delete cascade on
update cascade;
```