Національний університет «Полтавська політехніка імені Юрія Кондратюка»\_

(повне найменування вищого навчального закладу)

\_Навчально-науковий інститут інформаційних технологій і робототехніки\_

(повна назва інституту)

\_Кафедра комп’ютерних та інформаційних технологій і систем\_

(повна назва кафедри)

**Пояснювальна записка**

**до дипломного проєкту (роботи)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_бакалавра\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(освітньо-кваліфікаційний рівень)

на тему

«Розроблення web-додатку для списку задач

на базі бібліотеки React (Front-end)»

Виконав: студент \_\_4\_\_ курсу, групи 401-ТН

спеціальності

\_\_\_\_\_\_\_\_\_\_\_\_122 Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва спеціальності)

\_\_\_\_\_\_\_\_\_\_\_Удовицький Д. О.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_Гафіяк А.М.\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Полтава – 2021 року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА**

**ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**спеціальність 122 «****Комп’ютерні науки»**

**на тему**

**«****Розроблення web-додатку для списку задач на базі бібліотеки React (Front-end)»**

##### Студента групи 402 – ТН Удовицького Данила Олеговича

##### 

Керівник роботи

доктор педагогічних наук,

доцент Гафіяк А. М.

Завідувач кафедри

кандидат технічних наук,

доцент Головко Г. В.

Полтава – 2021

**РЕФЕРАТ**

Кваліфікаційна робота бакалавра: 69с., 21 малюнків, 4 додатоки, 9 джерел.

**Об’єкт дослідження**: діяльність підприємства з обслуговування систем безпеки на об’єктах підприємства.

**Мета роботи**: розробка автоматизованої інформаційної системи обліку діяльності підприємства з обслуговування систем безпеки на об’єктах. Методи проектування та розробка веб-серісу обліку діяльності підприємства з обслуговування обслуговування систем безпеки на об’єктах підприємства.

**Ключові слова**: підприємство, обслуговування систем безпеки, замовлення, облік, ремонт.

**ABSTRACT**

Bachelor's qualification work: 69p., 21 drawings, 4 appendices, 9 sources.

Object of research: the activity of the enterprise on maintenance of security systems at the objects of the enterprise.

Purpose: development of an automated information system for accounting the activities of the enterprise for the maintenance of security systems at facilities. Methods of designing and developing a web series of accounting for the activities of the enterprise for the maintenance of security systems at the facilities of the enterprise.

Key words: enterprise, maintenance of security systems, order, accounting, repair.

**ЗМІСТ**

Зміст..........................................................................................................................3

ВСТУП..................................................................................................................... 4

РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА.................................................................. 6

1.1 Огляд існуючих рішень .......................................................................... 6

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА .....................................................................8

2.1 Моделювання системи…………………………….................................8

2.2 Модель бази даних ………………………............................................13

2.3 Вибір платформи................................................................................... 15

2.4 Вибір та обґрунтування використання системного комплексного пакета XAMPP в якості web-сервера………...………………………………....18

2.5 Розробка інтерфейсу ..............................................................................19

2.6 Програмне рішення ................................................................................23

2.7 Реалізація програмного продукту………………………..……………27

ВИСНОВКИ...........................................................................................................32

СПИСОК ЛІТЕРАТУРИ.......................................................................................33

ДОДАТОК А (ТЕХНІЧНЕ ЗАВДАННЯ).......................................................... 34

ДОДАТОК Б (ІНТЕРФЕЙС КОРИСТУВАЧА СИСТЕМИ) ......................... 36

ДОДАТОК В (ВИХІДНІ КОДИ СИСТЕМИ)................................................... 38

ДОДАТОК Г (ДОВІДКА ПРО ВИКОРИСТАННЯ) ........................................ 65

**ВСТУП**

На сьогоднішній день інформаційні системи обліку обслуговування систем безпеки існують виключно для підтримки техніки персонального використання. Підприємства України, які мають обширну систему безпеки, яка розподілена по великій території на сьогодні не використовують централізованих систем віддаленої підтримки. В таких підприємствах та установах використовуються три схеми обслуговування систем безпеки,а саме:

* шляхом письмового складання замовлень до підприємства, що обслуговує,
* повідомлення відповідного працівника за допомогою електронної пошти,
* усне повідомлення працівника по телефонному зв’язку, яке у кінцевому випадку також передбачає складання письмових звітів.

Отже впровадження подібних локалізованих систем є досить доцільним рішенням на сьогоднішній день.

Така система необхідна для підприємства так як існує велика кількість об’єктів, які обладнанні системами безпеки. Під час експлуатації систем безпеки виникає необхідність її обслуговувати,оновлювати та ремонтувати тому виникає потреба в централізованому сервісі з можливістю віддаленого прийому.

Метою роботи є розроблення інформаційної системи обліку діяльності підприємства з обслуговування систем безпеки об’єктів підприємства, яка буде функціонувати на базі ТОВ «СП»Охорона». Так система повинна автоматизувати збір замовлень та облік робіт підприємства. Система створює сприятливі та комфортні умови роботи будь-яких користувачів – керівника підприємства обслуговування систем безпеки, робітників та користувачів, яким необхідно подати заявку на обслуговування або ремонт.

Об’єктом дослідження являється діяльність підприємства з обслуговування систем безпеки на об’єктах підприємства.

Організація подібної системи можлива шляхом організації веб-сервісу, який буде надавати наступний основний функціонал:

* приймання замовлень;
* розподіл адміністратором замовлень між працівниками;
* отримання працівниками замовлень в реальному часі;
* складання звіту та обліку замінених деталей працівником;
* складання статистики роботи та швидкий перегляд виконаної роботи адміністратором.

Впровадження системи насамперед сприятиме наступним факторам:

* зменшення часу проходження замовлень на обслуговування та ремонт систем безпеки;
* облік та моніторинг діяльності співробітників підприємства.

**РОЗДІЛ 1**

**ТЕОРЕТИЧНА ЧАСТИНА**

**1.1 Огляд існуючих рішень**

З кожним роком суттєво зростає потік наукової інформації, швидко змінюються інженерні рішення. Впроваджуються при монтажі, ремонті і реконструкції систем безпеки нове обладнання, технології, комплексні методи безпеки, нові форми організації праці.

Тому розробка АІС для обслуговування систем безпеки є дуже корисною для встановлення взаємозв’язку між замовником послуг та підприємством, керівником підприємства та персоналом що надає ці послуги.

Оскільки на сьогодні подібні системи не існує на підприємстві, я провела огляд саме таких систем.

Найбільші системи обслуговування та підтримки систем безпеки направлені на комерційну діяльність ,а їх локалізація охоплює цілі міста. Такі системи спрямовані на надання послуг технічного обслуговування за визначену плату, або по гарантії, якщо така система відноситься до певної фірми систем безпеки. Усі подібні системи надають наступні послуги :

* подання заявок на обслуговування(Розглядається тільки адреса замовника);
* падання майстра,який буде виконувати замовлення при виконанні умов згоди про надання подібних послуг(оплата або гарантійний ремонт);
* усунення проблем не тільки з обладнанням, а й різноманітну підтримки систем безпеки (відновлення системи безпеки, налаштування програмного забезпечення, та ін.).

Впровадження подібних систем централізовано на підприємстві або інших установах які обширно використовують системи безпеки, тільки частково вирішить проблеми таких установ. Подібна віддалена підтримка не веде облік виконаної роботи для звітності на підприємстві, як наприклад перелік замінених або доданих деталей, облік програмного забезпечення, яке було встановлене або відновлене. Такий облік вкрай необхідно вести на підприємствах для підрахування коштів на нове обладнання, контролю резервів обладнання.

Розглянувши функціонал існуючих систем, я створила таблицю, яка описує функціонал кожної конкретної системи у порівнянні з іншими.

Таблиця 1.1 – Функціонал існуючих систем.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Система | Подача замовлення | Замовлення майстра | Облік виконаної роботи | Облік деталей | Формування звітів | Контроль статусу виконання |
| ТОВ«СП»Охорона» |  |  |  |  |  |  |
| ПП «Явір 2000» |  |  |  |  |  |  |
| ТОВ Венбест |  |  |  |  |  |  |
| ПП Дозор |  |  |  |  |  |  |

Отже існуючі системи не надають достатню кількість функціонування для розробки даної дипломної роботи. Тому виконавши аналіз необхідних функцій роботи моєї інформаційної системи я розширила функціонал існуючих систем та привела усе функціонування до необхідного подання у ТОВ «СП»Охорона».

**РОЗДІЛ 2**

**ПРОЕКТНА ЧАСТИНА**

**2.1 Моделювання системи**

**2.1.1 Модель функціонування. IDEF0**– FunctionModeling – методологія функціонального моделювання. За допомогою наглядної графічного моделі IDEF0 досліджувана система постає перед розробником та аналітикам у вигляді набору взаємозалежних функцій(функціональних блоків – у термінах IDEF0). Як правило, моделювання засобами IDEF0 є першим етапом вивчення будь-якої системи.

Така діаграма явно демонструє функції, які виконує розроблена мною інформаційна система обліку діяльності сервісного центру обслуговування комп’ютерної техніки. Контекстна та декомпозована діаграма зображені на рисунок 2.1 та рисунок 2.2.

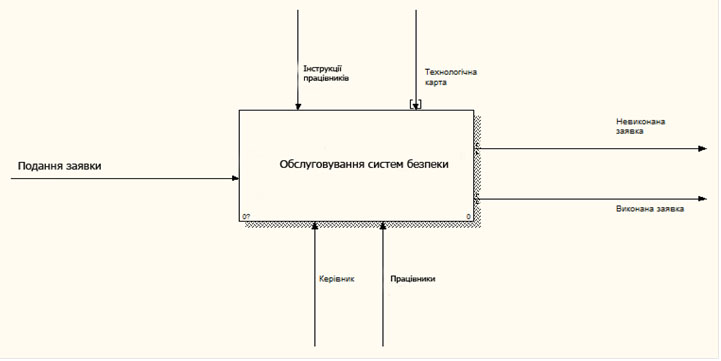


Рисунок 2.1 – Контекстна діаграма.

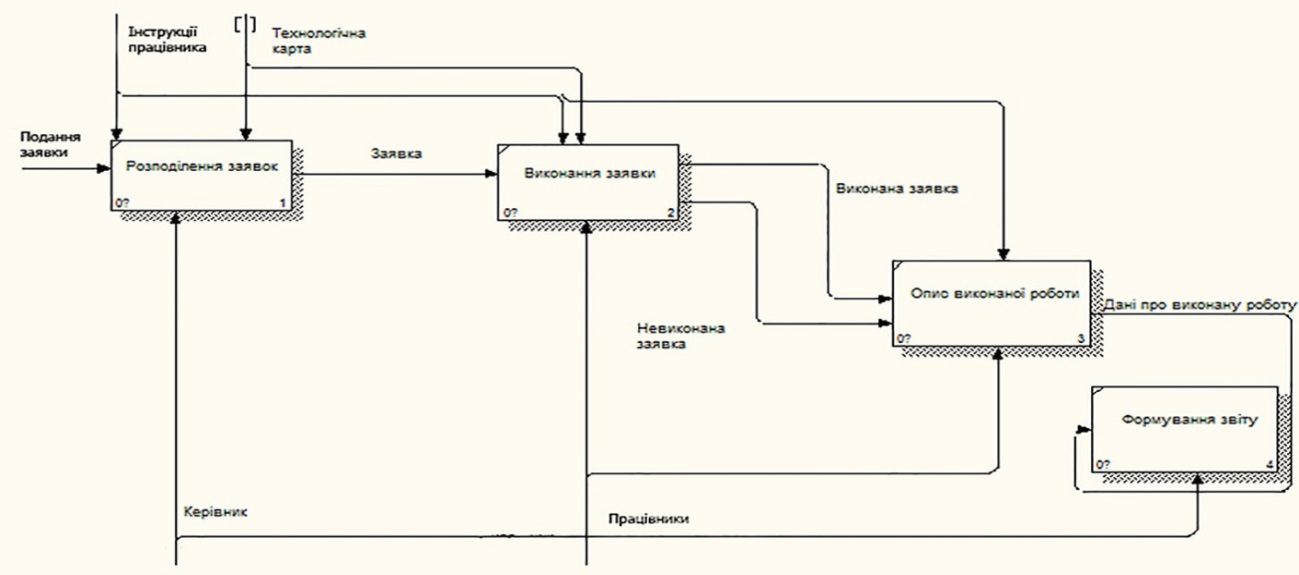


Рисунок 2.2 – Діаграма декомпозиції.

**2.1.2. Модель потоків даних. DFD** –Data Flow Diagrams – діаграми потоків даних. Так називається методологія графічного структурного аналізу, що описує зовнішні по відношенню до системи джерела і адресати даних, логічні функції, потоки даних та сховища даних, до яких здійснюється доступ. Діаграма потоків даних (dataflowdiagram, DFD) – один з основних інструментів структурного аналізу та проектування інформаційних систем, що існували до широкого розповсюдження UML. Незважаючи на що має місце в сучасних умовах зміщення акцентів від структурного до об'єктно-орієнтованого підходу до аналізу та проектування систем, «старовинні» структурні нотації як і раніше широко і ефективно використовуються як в бізнес-аналізі, так і в аналізі інформаційних систем. Побудована мною діаграма показує потік даних в середині центру обслуговування комп’ютерної техніки.



Рисунок 2.3 – Діаграма потоків даних.

**2.1.3. Діаграми варіантів використання.** Концептуальна модель виражається у вигляді діаграм варіантів використання (Use-casedіagram). Цей тип діаграм служить для проведення ітераційного циклу загальної постановки завдання разом із замовником. Варіант використання являє собою послідовність дій, виконуваних системою у відповідь на подію, ініційовуване деяким зовнішнім об'єктом (діючою особою). Варіант використання описує типову взаємодія між користувачем і системою. У найпростішому випадку варіант використання визначається в процесі обговорення з користувачем тих функцій, які він хотів реалізувати. Ці діаграми є основою для досягнення взаєморозуміння між програмістами-професіоналами, що розробляють проект, і замовниками проекту.

На схемі зображено процес зв’язку обслуговуючого персоналу з користувачами, які залишили замовлення на обслуговування.

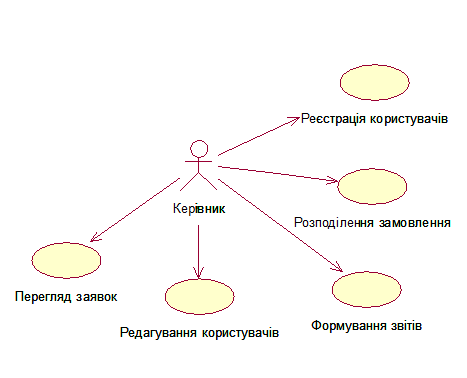


Рисунок 2.4 – Діаграма використання(Керівник)

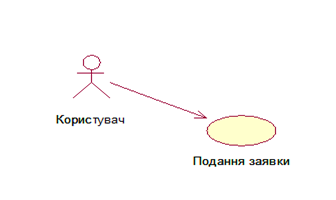


Рисунок 2.5 – Діаграма використання (Користувач)

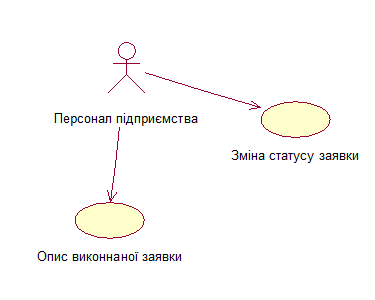


Рисунок 2.6 – Діаграма використання (Персонал підприємства)

**2.1.4. Діаграма активності (actіvіty dіagram).** Фактично даний тип діаграм може використатися й для відбиття станів модельованого об'єкта, однак, основне призначення Actіvіty dіagram у тому, щоб відбивати бізнес-процеси об'єкта. Цей тип діаграм дозволяє показати не тільки послідовність процесів, але й розгалуження й навіть синхронізацію процесів. Він дозволяє проектувати алгоритми поводження об'єктів будь-якої складності.

Для моделювання процесу виконання операцій у мові UML використаються діаграми діяльності. Застосовувана в них графічна нотація багато в чому схожа на нотацію діаграми станів, оскільки на цих діаграмах також присутні позначення станів і переходів. Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід у наступний стан виконується тільки при завершенні цієї операції.

Таким чином, діаграми діяльності можна вважати окремим випадком діаграм станів. Вони дозволяють реалізувати в мові UML із процедурного й синхронного керування, обумовленого завершенням внутрішніх діяльностей і дій. Основним напрямком використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно представити алгоритми їхнього виконання.

У контексті мови UML діяльність (actіvіty) являє собою сукупність окремих обчислень, виконуваних автоматом, що приводять до деякого результату або дії (actіon). На діаграмі діяльності відображається логіка й послідовність переходів від однієї діяльності до іншої, а увага аналітика фокусується на результатах. Результат діяльності може привести до зміни стану системи або поверненню деякого значення.

Початок Отримання замовлення Розподілення замовлень Відправлене замовлення Виконання замовлення Зміна статусу замовлення Виконане замовлення Опис виконаного замовлення Кінець

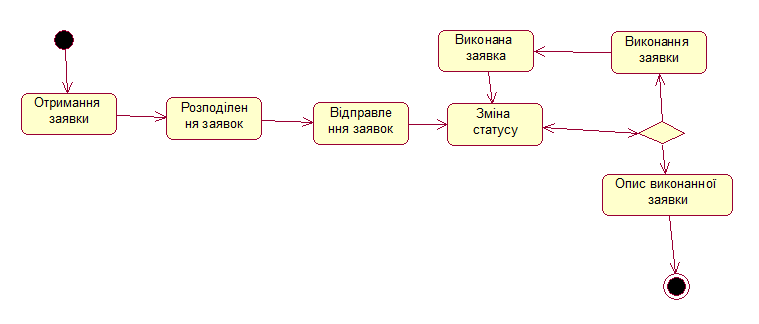


Рисунок 2.7 – Діаграма активності

**2.1.5. Діаграма взаємодії.** Діаграми взаємодії є моделями, що описують поводження взаємодіючих об'єктів. Як правило, діаграма взаємодії охоплює поводження тільки одного варіанта використання. На такій діаграмі відображається ряд об'єктів і ті повідомлення, якими вони обмінюються між собою в рамках одного варіанта використання.

Існує два види діаграм взаємодії: діаграми послідовності (sequence dіagrams) і кооперативні, або співробітництва (collaboratіon dіagrams). Діаграми послідовності визначають тимчасову послідовність переданих повідомлень, порядок, вид і тип повідомлення, що відбуваються в рамках варіанта використання. Діаграми послідовності й кооперативні є різними поглядами на ті самі процеси, тому RatіonalRose дозволяє створити з діаграми послідовності діаграму Кооперації й навпаки, а також робить автоматичну синхронізацію цих діаграм.

На діаграмі послідовності взаємодія зображується у вигляді двовимірної схеми: вертикальне (час) і горизонтальне (об'єкти, що беруть участь у взаємодії). Істотна тільки послідовність повідомлень, однак тимчасова вісь може служити реальною метрикою виміру активності об'єкта. Прямокутники на вертикальних лініях показують "час життя" об'єкта. Лінії зі стрілками й написами назв методів означають виклик методу в об'єкта.

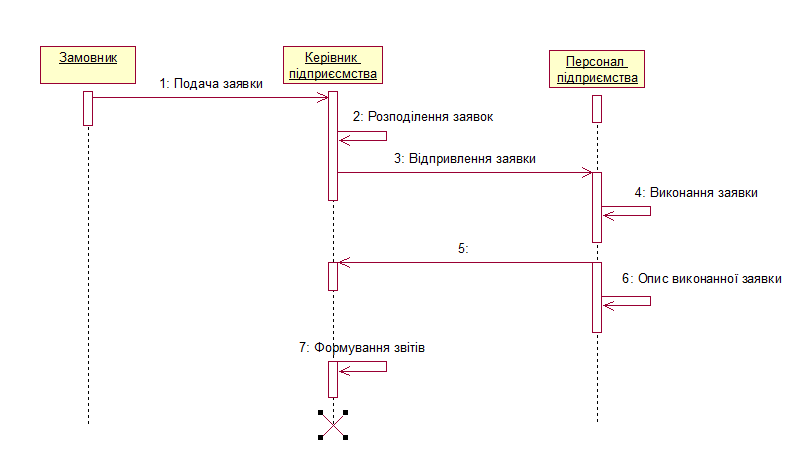


Рисунок 2.8 – Діаграма взаємодії

**2.2 Модель бази даних**

**2.2.1 ER –діаграма. Модель "сутність-зв'язок" (ER-модель)**(*entity-relationship diagram*)-модельданих, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель - це мета-модель даних, тобто засіб опису моделей даних.

Модель "сутність-зв'язок" ґрунтується на якійсь важливій семантичній інформації про реальний світ і призначена для логічного представлення даних. Вона визначає значення даних в контексті їх взаємозв'язку з іншими даними. Важливим є той факт, що з моделі "сутність-зв'язок" можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найбільш загальною.

Будь-який фрагмент наочної області може бути представлений як безліч сутностей, між якими існує деяка безліч зв'язків.

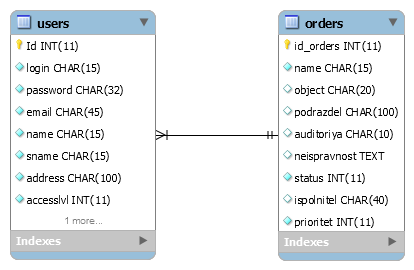


Рисунок 2.9 – Модель "сутність-зв'язок"

Така модель організації бази даних найбільше підходить до даної системи.

Дана організаціє дає змогу проводити деякі зміни в відомостях користувача системи, при подачі заявки. Якщо на момент подання замовлення на обслуговування обладнання систем безпеки змінився підрозділ замовника, він може вказати інший. Тому в деяких таблицях дублюються окремі поля, що є забезпечує можливість зміни відомостей, а не статичне їх прикріплення до реєстраційних даних.

Основним кодуванням бази даних є utf8, збір utf8\_general\_ci. Вибір такого кодування був зумовлений необхідністю використання кириличних символів для відображення більшості текстової інформації, що й дозволяє вибране кодування.

Довжина поля паролю становить 32 символи, така довжина була вибрана через необхідність зберігати в ньому пароль захешований системою md5, яка кодує саме до розміру в 32 символи.

**2.3 Вибір платформи**

Розробка інформаційної системи обліку діяльності сервісного центру з обслуговування обладнання систем безпеки передбачає впровадження її на базі локального сервера. Тому вибір платформи та програмного забезпечення зумовлений тим, яку платформу та ПЗ використовує локальний сервер підприємства. Тому для розроблення системи я обрала наступні програмні продукти.

**Apache HTTP-сервер**-відкритий веб-сервер Інтернет для UNIX-подібних,Microsoft Windows, Novell NetWare та інших операційних систем. На сьогодні є найуживанішим Веб сервером мережі Інтернет.

Apache розроблюється та підтримується спільнотою розробників відкритого програмного забезпечення під керівництвом Apach Software Foundation.

Web-сервер Apache є самостійним, некомерційним, вільно розповсюджуваним продуктом. Продукт підтримує безліч можливостей, багато з яких реалізовані як скомпільовані модулі, які розширюють основні функціональні можливості. Вони різняться від серверної підтримки мов програмування до схем аутентифікації.

Існують інтерфейси для підтримки мов програмування Perl, Python, Tcl PHP.

Функції віртуального хостингу дозволяють одній інсталяції Apache обслуговувати різні веб-сайти. Тому використовуючи функцію віртуального хостингу, розроблену систему обліку роботи центру обслуговування обладнання систем безпеки можливо впровадити використовуючи вже встановлений та налагоджений Apache.

Apache передусім використовується для передачі через HTTP статичних та динамічних веб-сторінок.. Багато веб-застосунків спроектовано, зважаючи на середовище і можливості, які надає цей веб-сервер.Apache зіграв ключову роль у початковому зростанні всесвітньої павутини, і продовжує бути найпопулярнішим у світі веб-сервером, де-факто платформою, на яку орієнтуються інші веб-сервери.

**PHP** (PHP:Hypertext Preprocessor- PHP:гіпертекстовий препроцесор)-скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java,. NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP-проект відкритого програмного забезпечення. PHP інтерпретується веб-сервером в HTML-код, який передається на сторону клієнта.

На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати РНР для генерування і JavaScript-кодів які виконуються вже на стороні клієнта.

PHP — мова, яка може бути вбудована безпосередньо в html-код сторінок, які, в свою чергу коректно будуть оброблені PHP -інтерпретатором. Механізм РНР просто починає виконувати код після першої екрануючої послідовності (<?) і продовжує виконання до того моменту, коли він зустріне парну екрануючу послідовність (?>).

Дуже важливою особливістю мови PHP для розробки мною системи є наявність інтерфейсів до багатьох баз даних. В PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, Hyperware, Informix, InterBase, Sybase. Через стандарт відкритого інтерфейсу зв'язку з базами даних (Open Database Connectivity Standard — ODBC) можна підключатися до всіх баз даних, до яких існує драйвер.

Дана скрипкова мова програмування встановлюється на сервер Apache та є його розширенням для роботи з мовою PHP. Для виконання моєї дипломної роботи я використовувала версію 5.2 мови PHP.

**MySQL** -вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Для некомерційного використання MySQL є безкоштовним. Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн.;
* висока швидкість виконання команд;
* наявність простої і ефективної системи безпеки.

Для реалізації інформаційної системи обліку діяльності сервісного центру з обслуговування обладнання систем безпеки, я використовувала останній реліз мови SQL , а саме версію 5.7.

Усі перелічені програмні засоби та платформа PHP використовуються для функціонування локального серверу підприємства. Тому я вирішила виконувати свою роботу з їх використанням. Це було зумовлено тим,що зникає необхідність встановлення нового програмного забезпечення на сервер , його конфігурування та внесення змін в використання вже існуючого локального серверу. Також такий вибір було зроблено тому що дані продукти є повністю безкоштовними та усувають проблему з адмініструванням ліцензій для використання. Крім того ці продукти на сьогоднішній день є безумовними лідерами в розробці веб-продуктів, тому вони мають бездоганну відмово стійкість, вбудовану систему захисту даних та високу швидкодію.

**2.4 Вибір та обґрунтування використання системного комплексного пакета XAMPP в якості web-сервера.**

XAMPP -набір дистрибутивів (Apache, PHP, MySQL, Perl і т.д.) і програмна оболонка, призначені для створення і налагодження сайтів (веб-додатків, іншого динамічного вмісту інтернет-сторінок) на локальному ПК (без необхідності підключення до мережі Інтернет) під управлінням ОС Windows.

Використання локального web - сервера зумовлене необхідністю дотримання корпоративної таємниці на підприємстві. Працездатність мною розробленої інформаційної системи неможливо зробити при роботі вдома – так як технології SSI (Server-Side Includes - Включення на стороні сервера), CGI(Common Gateway Interface - Загальний шлюзовий інтерфейс) і РНР вимагають використання сервера. Для вирішення цієї проблеми і був використаний .

Головна особливість XAMPP – повністю безкоштовний і простий в установці дистрибутив Apache, що містить MariaDB, PHP і Perl. XAMPP створений з відкритим вихідним кодом, щоб бути неймовірно простим в установці і у використанні. На думку фахівців web-програмування, емулятор веб-сервера XAMPP підходить для створення власного хостингу, а також дозволяє реалізовувати бази даних у власних сайтах, використовуючи в системі управління базами даних MySQL.

Розглянемо основні конкурентні переваги XAMPP:

* безкоштовність.
* простота в установці та використанні.
* постійне оновлення продукту;
* швидкість роботи (при зміні шаблону або додавання нових функцій зміни відбуваються моментально);
* економічна економія (не потрібно замовляти і засмічувати платний хостинг базами даних для перевірки роботи сайту).

**2.5 Розробка інтерфейсу**

Однією з найважливіших етапів розроблення ПЗ являється розробка інтерфейсу користувача. Від розробленого інтерфейсу залежить те, як зручно буде користуватися системою,а отже й її популярність або вживаність. Системою будуть користуватися виключно у випадку, коли користувач добре розуміє що саме він по винен робити щоб отримати бажаний йому результат. Тобто інтерфейс повинен бути зрозумілим на інтуїтивному рівні користувача.

Сьогодні, коли говорять про екранний інтерфейс користувача, звичайно мають на увазі графічний інтерфейс (далі GUI). Текстовим інтерфейсом зараз майже не користуються.

Проектуючи інтерфейс сайту, я враховувала те, як будуть на нього реагувати потенційні відвідувачі . Визначивши головні функції системи, я зробила упор на них та розділив по типам користувачів. Тому зайшовши до сайту користувачі будуть бачити різний функціонал в залежності від того, до якого типу вони входять :Адміністратор,співробітник, користувач. Такий спосіб розподілення позбавить користувачів від збентеження, який пункт меню йому необхідно вибрати, тому що кожен користувач буде отримувати тільки список тих дій, які йому необхідно виконати.

**2.5.1 Принципи проектування інтерфейсу користувача.** Найбільш ефективним підходом до проектування інтерфейсу користувача є його розроблення шляхом моделюванням функцій користувача. На початку процесу прототипування створюються паперові макети інтерфейсу, потім розробляються екранні форми, які моделюють взаємодію з користувачем. Бажано, щоб кінцеві користувачі приймали активну участь у процесі проектування інтерфейсу. В одних випадках користувачі допоможуть оцінити інтерфейс. В інших – допоможуть знайти помилки у його розроблені.

Розробники інтерфейсів завжди повинні враховувати фізичні й розумові здібності людей, які будуть працювати з ПЗ. Люди спроможні запам’ятовувати досить обмежений об’єм інформації і на короткий термін, і роблять помилки, якщо приходиться вводити вручну великі об’єми даних або працювати у напружених умовах. Фізичні можливості людей можуть суттєво розрізнятися, тому при проектуванні інтерфейсів користувача необхідно постійно пам’ятати про це.

Основою принципів проектування інтерфейсів користувача є людські спроможності. У табл. 2.1 подані основні принципи, які застосовуються при проектуванні будь-яких інтерфейсів GUI.

Основною вимогою в розробці інтерфейсу є доступність та зрозумілість для усіх користувачів. Тому я вибрала найпопулярнішу модель побудови інтерфейсу, яка містить в собі 2 колонки (меню та основний блок з інформацією).

|  |  |
| --- | --- |
|  |  |

Рисунок 2.10 – Модель побудови інтерфейсу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Адміністратор | Співробітник | Користувач |

Рисунок 2.11 – Меню в залежності від типу користувача

Колонка «меню» вміщує в себе усі функції, які може використовувати кожен користувач в залежності від його типу.

Таблиця 2.1 – Принципи проектування GUI

|  |  |
| --- | --- |
| Принцип | Опис |
| Врахування знань користувача | В інтерфейсі необхідно використовувати терміни й поняття, які взяті із досвіду майбутніх користувачів системи |
| Узгодженість | Інтерфейс повинен бути узгодженим у тому розумінні, що однотипні (але різні) операції треба виконувати одним і тим самим чином |
| Мінімум несподіванок | Поведінка системи повинна бути прогнозована |
| Здатність до поновлення | Інтерфейс повинен мати засоби, які дозволяють користувачу поновлювати дані після помилкових дій |
| Порадник користувача | Інтерфейс повинен надавати необхідну інформацію контекстно-залежною довідкою |
| Врахування різнорідності користувачів | В інтерфейсі повинні бути засоби для зручної взаємодії з користувачем, який має різний рівень кваліфікації й різні можливості |

**2.5.2 Використання в GUI кольорів.** Правильне використання кольорів робить інтерфейс користувача більш зручним для розуміння й керування. Разом з тим використання кольорів може бути неправильним, що візуально неприємно і навіть провокує помилки. Основним принципом розробників GUI повинно бути обережне використання кольорів на екранах.

Існує 14 правил ефективного використання кольору у GUI. Найбільш важливі такі:

1. Використовуйте обмежену кількість кольорів. Для вікон не треба використовувати більше за чотири або п`яти різних кольорів, в інтерфейсі системи не повинно бути більше за сім кольорів.

2. Використовуйте різні кольори для показу змін у стані систем. Якщо на екрані змінилися кольори, значить відбулась деяка подія. Виділення кольором особливо необхідно на складних екранах, на яких відображаються сотні різних об`єктів.

3. Для допомоги користувачу користуйтеся кольоровим кодуванням. Якщо користувачам необхідно виділяти аномальні елементи, виділяйте їх кольором. Якщо треба знайти подібні елементи, виділіть їх однаковим кольором.

4. Використовуйте кольорове кодування помірковано й послідовно. Якщо у деякій частині системи повідомлення про помилку відображається, наприклад, червоним кольором, то у всіх інших частинах подібні повідомлення повинні відображатися таким самим кольором. Тоді червоний колір не треба використовувати ні де більше. Слід також пам’ятати, що у користувача може бути власне уявлення про значення окремих кольорів.

5. Обережно використовуйте додаткові кольори. Фізіологічні особливості людського ока не дозволяють одночасно сфокусуватися на червоному й синьому кольорах. Тому послідовність червоних і синіх зображень викликає зорову напруженість. Деякі комбінації кольорів також можуть візуально порушувати або ускладнювати читання.

**2.5.3. Засоби підтримки користувача.** Досить важливою функцією проектування інтерфейсу являється передбачення підказок до використання певної функції, описання самої функції та оброблення повідомлень про помилки, пошук необхідної інформації.

Оскільки проектування корисної і місткої інформації для користувача – справа серйозна, то воно повинно оцінюватися на тому ж рівні, що й архітектура системи або програмний код. Проектування повідомлень потребує значного часу та чи малих зусиль.

Обробка повідомлень про помилку – це один з засобів навчання користувачем працювати з системою. Випадково виконавши певну дію з помилкою користувач повинен отримати повідомлення про це , в інтуїтивно зрозумілій формі, що саме користувач зробив неправильно та що необхідно зробити для виправлення цієї помилки.

Пошук інформації – засіб надання користувачеві отримання лише тої інформації, яку він хоче бачити, з великої кількості іншої інформації. Тому дана функція повинна бути передбачена в інтерфейсі системи в необхідному для неї місці.

Прикладом її використання в інформаційній системи з обліку діяльності центру обслуговування обладнання систем безпеки, є пошукові формі на сторінках перегляду заявок та перегляду списку користувачів системи, увівши необхідні дані, користувач відсортує усі існуючи данні та відшукає за лічені секунді необхідну йому інформацію.

**2.6 Програмне рішення**

**2.6.1. Вибір моделі процесу створення ПЗ.**У сучасній програмній інженерії використовуються різні моделі процесів створення ПЗ. Вони описані у частині 2. Там також вказано, що вибір моделі залежіть від конкретних умов (часові терміни, виділені кошти, кваліфікація команди програмістів, особливості спілкування із замовником тощо), в яких приходиться планувати виконання проекту.

**2.6.2 Вибір моделі процесу розробки системи.** Оскільки вибір моделі розробки ПЗ залежить від виділених на розробку коштів та особливостями спілкування із замовниками, я вибрав модель змішаного типу.

Тобто така модель включає у себе декілька необхідних можливостей при розробці ПЗ. Оскільки при спілкуванні з замовниками було встановлено , що вимоги надані для створення системи охоплюють лише частину процесу, передбачалось , що деякі вимоги будуть формуватися в процесі створення ПЗ, Тому модель розробки повинна була бути гнучкої та дозволяти вносити певні зміни на будь-якому етапі розробки. Тому при створенні я використовував еволюційну модель розробки на етапах, які не ввійшли спочатку до вимог, а до етапів, які було чітко визначено в наданих умовах я застосовував каскадну модель розробки.

**2.6.3. Ієрархія прав користувачів системи.** Оскільки розроблена мною система має декілька типів користувачів, є необхідність в їх специфікації. Згідно до вимог наданими для розроблення системи, такими користувачами можуть бути: керівник центру обслуговування, персонал центру, користувач. Більш точне уявлення про роль кожного з користувачів дає схема на Рисунок 2.12.

1

**2**

Користувач

Користувач

Керівник

Персонал

**3**

1

**2**

Рисунок 2.12 – Представлення різних видів користувачів системи.

Пояснення каналів передачі даних від користувачів до системи:

1.Користувач має змогу залишати замовлення на обслуговування своєї техніки і не має доступу до реєстрації, усіх користувачів в системі реєструє керівник.

2.Керівник має біль ширший функціонал. Керівник реєструє нових користувачів, розподіляє заявки між персоналом, редагує відомості користувачів, формує звіти про виконання та має доступ до архіву усіх виконаних замовлень.

3.Персонал центру обслуговування, має змогу перевіряти направлені саме йому заявки, їх опрацювання, зміна статусу їх виконання, заповнення звітності про виконану роботу та змінені деталі для їх обліку.

2**.6.4. Інтерфейс користувача головної системи.** Реалізація всього інтерфейсу проведена за допомогою css фреймворку Bootstrap, який було розроблено працівниками соціальної мережі Twitter. Цей фреймворк включає в себе налаштування стилів css для кожного елементу HTML. Для використання такого фреймворку, необхідно задавати певний клас різноманітним елементам гіперрозмітки для того, щоб вони приймали описаний в стилях вигляд. Ця система використовується в відображенні інтерфейсу соціальної мережі Twitter, яка користується широкою популярністю та використовується мільйонами користувачів. Вибір такого фреймворку для розробки інтерфейсу я зробила через його розповсюдженість, оскільки він має ідентичні з Twitter-ом елементи дизайну , він буде знайомим та зрозумілим користувачу моєї системи. Використовуючи Bootstrap, я оформлювала наступні елементи : Форми,таблиці, кнопки, повідомлення про помилку, блоки <div> для меню та основного блоку інформації, оформлення тексту.

**2.6.5. Проектування структур даних та алгоритмів.** При написанні системи на мові PHP, виникає проблема виведення даних, та організація способу за допомогою якого це буде відбуватися. Використання даної мови дозволяє використовувати змішаній тип написання, тобто кожен файл може вміщувати як код графічного інтерфейсу так и програмний код скриптів. Однак такий спосіб являється досить не зручним, тому-що при виникненні необхідності зміни якоїсь частини графічного оформлення, зміни необхідно проводити на кожній сторінці, що також змушує частково змінювати самі скрипти. Тому я відокремив графічну частину оформлення від функціональної частини. Мова PHP дозволяє об’єднувати декілька документів за допомогою функції include():

include({шлях до програми, яку необхідно завантажити});

Тому я створив головний файл index.php , який підключає в такий спосіб

графічне оформлення сайту, та файл з налаштуваннями.

В свою чергу файл який вміщує інтерфейс сайту підключається сторінка з функціоналом який повинен виводитися.

<?php include "$getwey\_fun/interface.php"; ?>

Для виводу певної сторінки в файлі route.php прописуємо наступне:

function renderPartial(){

if (isset($\_GET['page'])){

$page = $\_GET['page'];

include $page.".php"; }}

Оскільки назва сторінки задана в вигляді змінної $page, то вона буде змінюватися в результаті призначення цій змінній нового значення.

Призначення нового значення змінній, яка підключає певну сторінку відбувається за допомогою передачі значення цієї змінної

<div class="main-content">

<div class="content">

<?php

renderPartial();

?>

</div>

</div>

Тобто передавши значення яке відповідає назві сторінки, змінна page приймає це значення та в основну область підключається необхідна сторінка з її функціоналом.

Також функцію Inclide() я використовувала для роботи з базою даних, підключаючись до неї за допомогою рядку:

include"bd.php";

Сам файл вміщує наступний код:

$db = mysql\_connect ("localhost","root","\*\*\*\*\*\*");//дані для з’єднання

mysql\_select\_db ("diplom ",$db); //вибір бази даних

mysql\_query("SET NAMES 'utf8';");

mysql\_query("SET CHARACTER SET 'utf8';");

mysql\_query("SET SESSION collation\_connection = 'utf8\_general\_ci';");

//вибір кодування тексту за умовчанням

Така організація спрощує роботу з підключенням до бази даних, при якій зникає необхідність постійно дублювати одні й ті самі налаштування.

**2.7 Реалізація програмного продукту**

**2.7.1 Структура сайту.** Структура сайту, що розробляється, наведена на Рисунок 2.13.



Рисунок 2.13 – Структура сайту

**2.7.2 Дизайн web-сайту.** Відповідно до розробленої структури, була спроектована головна сторінка сайту (Рисунок 2.14). Вона містить всі основні структурні елементи, перехід за якими здійснюється за допомогою гіперпосилань.

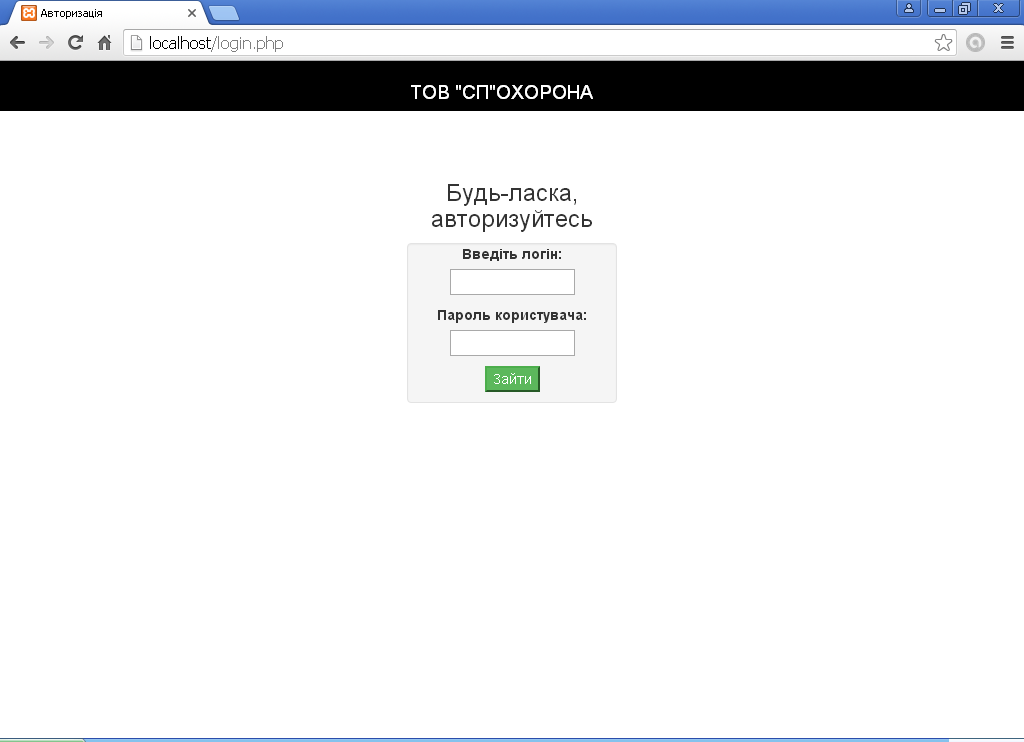


Рисунок 2.14 – Головна сторінка

При попадання на головну сторінку відбувається авторизація користувачів. На головній сторінці існує два поле для введення даних, в кожному з яких необхідно потрібно ввести login та пароль.

Після авторизації відповідний користувач потрапляє на відповідну сторінку з інтерфейсом користувача. При цьому як і планувалося у кожного користувача свій інтерфейс і свої відповідні функції (Рисунок 2.15, 2.16, 2.17)

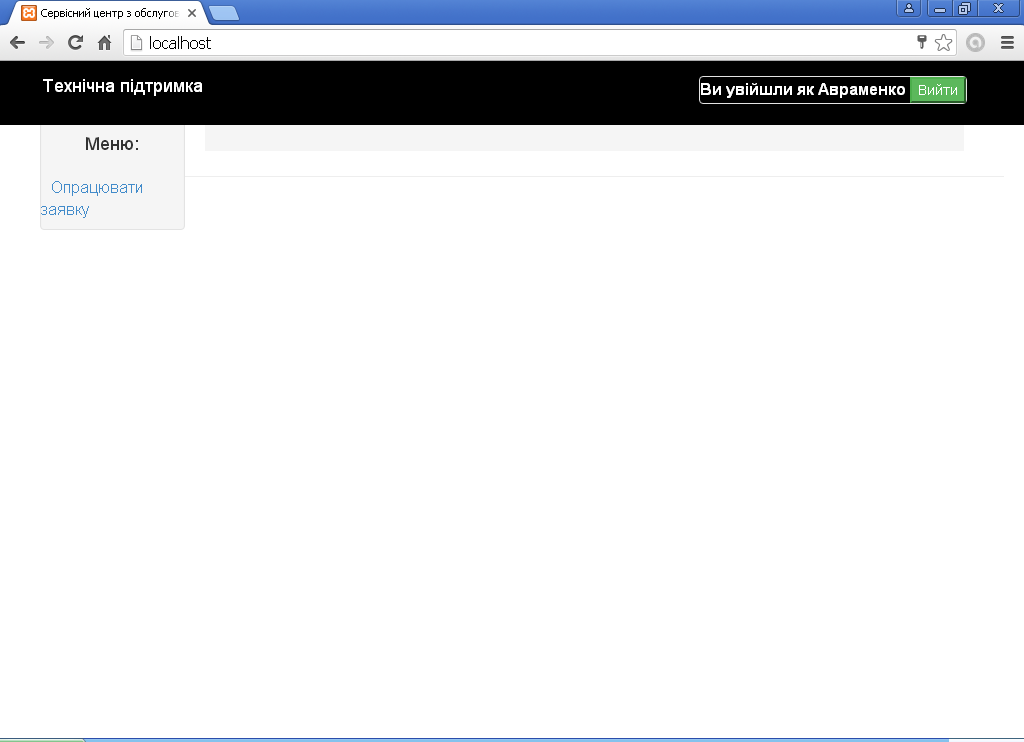


Рисунок 2.15 – Інтерфейс користувача (робітник)

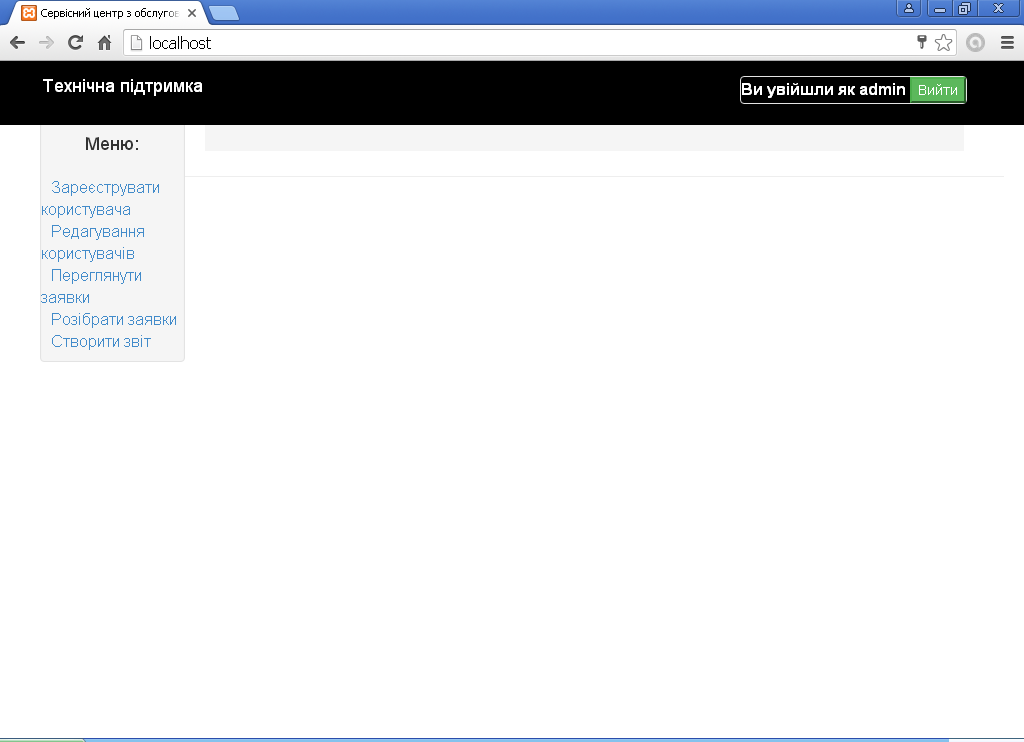


Рисунок 2.16 – Інтерфейс користувача (керівник)

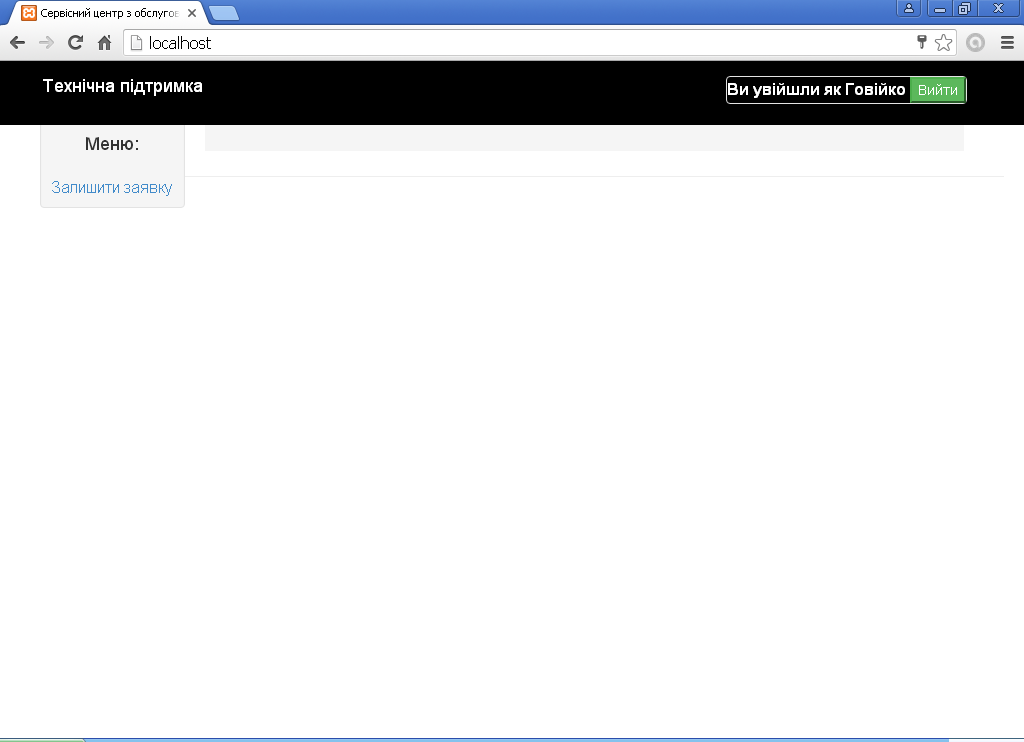


Рисунок 2.17 – Інтерфейс користувача (користувач)

Керівник має можливість реєструвати та редагувати користувачів на сайті (Рисунок 2.18, 2.19).

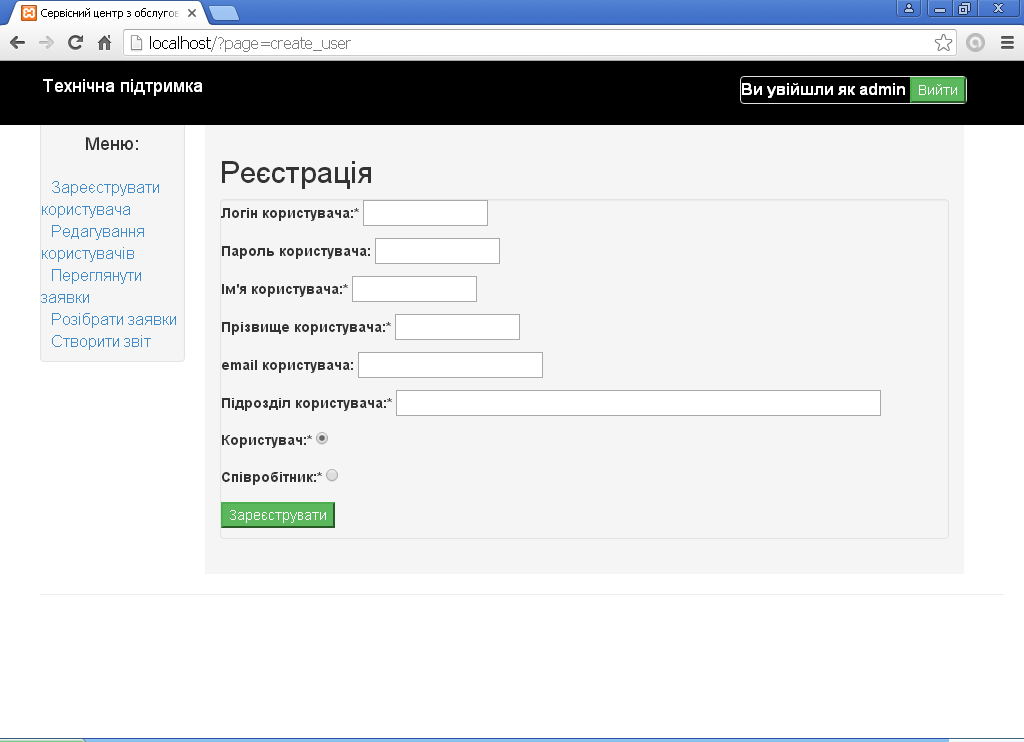


Рисунок 2.18 – Сторінка реєстрації користувача

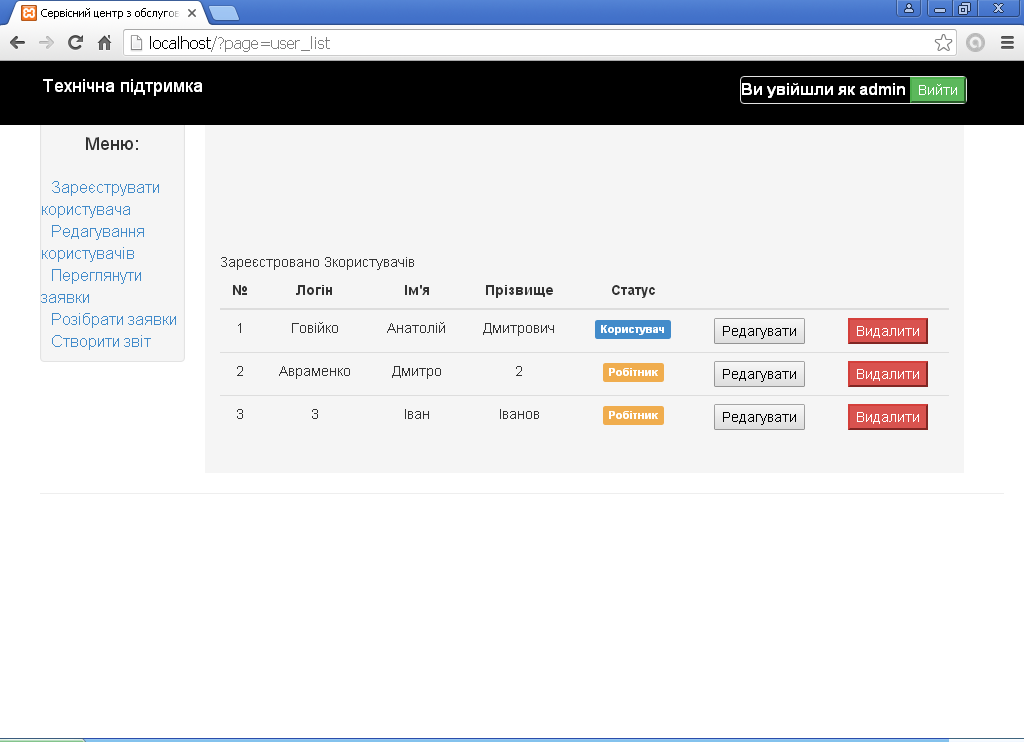


Рисунок 2.19 – Сторінка редагування користувача

При переході керівником на сторінку *Переглянути заявки* можливо продивитися кількість заявок, а також редагувати статус виконання, пріоритет виконання та назначити виконавця робіт (рис 2.20).

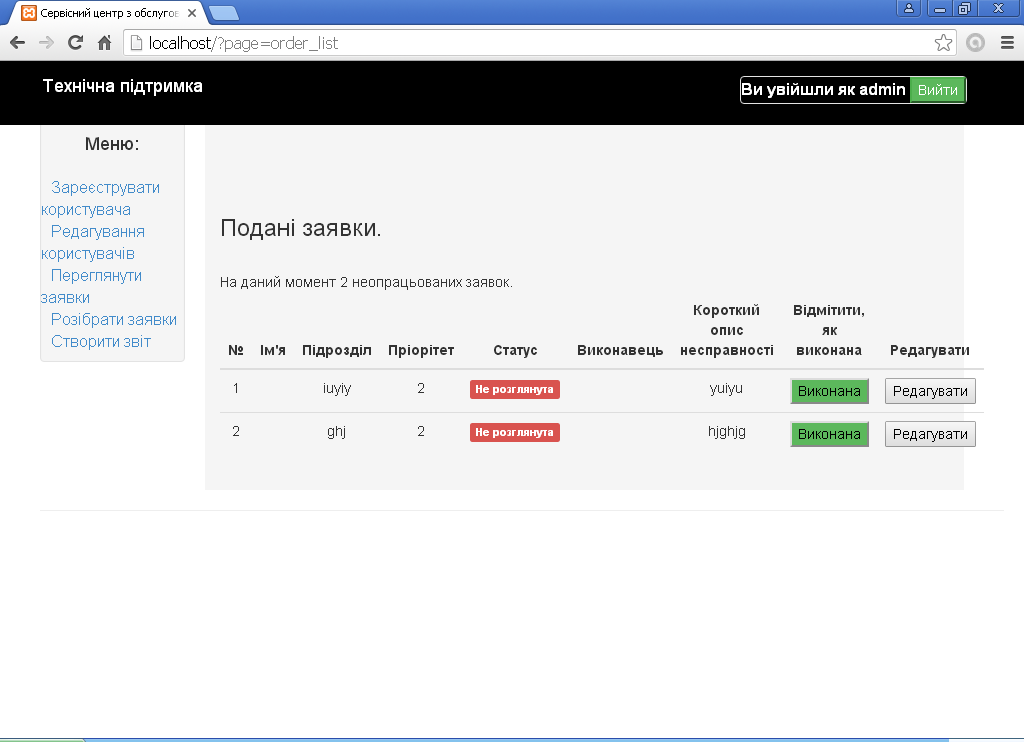


Рисунок 2.20 – Сторінка Переглянути заявки

Також керівником можливо створити звіт виконаних робіт. Для цього потрібно перейти на сторінку *Створити звіт*. На якій підраховується кількість виконаних та не виконаних робіт персоналом підприємства (Рисунок 2.21).

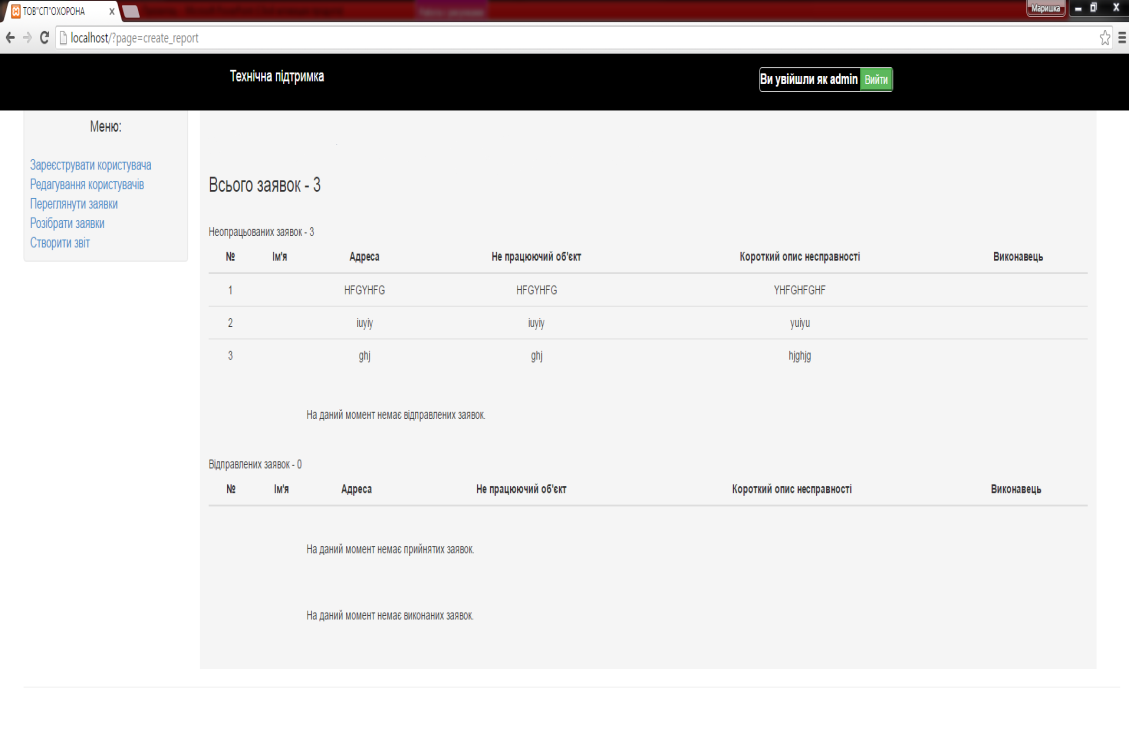


Рисунок 2.21 – Сторінка Створити звіт

**ВИСНОВКИ**

1. Сформульована необхідність створення інформаційної системи, яка допоможе більш ефективно та швидко проводити облік діяльності підприємства по обслуговуванні обладнання систем безпеки.

2.Виконана програмна реалізація такої системи.

3.Виконана система повністю функціонує, та готова до впровадження в роботу на базі локального сервера підприємства

Впровадження системи вирішуєтакі проблеми:

1. Зменшення часу на відновлення систем безпеки через несправність обладнання;

2. Облік діяльності персоналу підприємства;

3. Побудова звітів про роботу персоналу підприємства.

**СПИСОК ЛІТЕРАТУРИ**

1. Гагаріна Л.Г. Технологія розробки програмного забезпечення./ Є.В.Кокорева, Б.Д.Віснадул. - СПб. 2008.С. 400.

2. Липаев В.В. Проектування програмних засобів. - СПб. 2008.302 с.

3. Якоб Нільсен. Веб-дизайн. Аналіз зручності використання веб-сайтів по руху очей./ Кара Перніче. —Москва, Вільямс, 2010 р. – 496 с.

4. Томсон Л. РозробкаWeb-додатківна PHP і MySQL./Веллінг Л.-"ДиаСофт", 2001. – 672с.

5. Ратшіллер Т. PHP4: розробка Web-додатків./ Геркен Т. - СПб: Питер, 2001. – 384 с.

6. Яргер Р. MySQL и mSQL. Базы данных для небольших предприятий и Интерне-та./ Риз Дж., Кинг Т. - СПб. 2000 - 560 с.

7. Дебольт.HTML и CSS. Совместное использование./ Вирджиния.- "НТ

Пресс".2006. - 512 c.

8. Зандстра М., PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, Patterns and Practice, Third Edition — М.: [«Вильямс»](http://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2010. — С. 560. — [ISBN 978-5-8459-1689-1](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:BookSources/9785845916891).

9. Суэринг С., Конверс Т., Джойс П. PHP и MySQL. Библия программиста, 2-е издание = PHP 6 and MySQL 6 Bible — М.: [«Диалектика»](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B0%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2010. — 912 с. — [ISBN 978-5-8459-1640-2](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:BookSources/9785845916402).

**ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**

**1. Найменування сайту**

Найменування сайту: "Інформаційна система обліку обслуговування систем безпеки на об’єктах

**2. Призначення і область застосування**

Веб-сайт,основною метою якого є обробка заявок на ремонт обладнання на об’єктах підприємства.

База даних сайту містить наступні дані:

1. Дані про користувача(Ім’я, Прізвище, підрозділ, електронна пошта).

2. Список обладнаня,яку можливо ремонтувати.

3. Відомості про обробку заявки(дата виконання,виконавець,опис виконаної

роботи,пріоритет)

4. Облік замінених,доданих або знятих деталей.

**3. Вимоги до функціональних характеристик**

Система повинна забезпечувати можливість виконання перерахованих нижче функцій:

1.Поділ користувачів, які використовують сайт:

1). Керівник;

2). Обслуговуючий персонал;

3). Користувач.

2.Можливість залишити заявку.

3.Можливість обробляти подані заявки керівником,та назначати їм виконавця та пріорітет.

4.Можливість робітником отримувати надані йому заявки, та заповнювати данні про виконану роботу.

5.Можливість керівнику переглядати звіти по виконаній роботі.

6.Можливіст керівником реєструвати нових користувачів та робітників.

7.Можливість керівником переглядати список користувачів, та редагувати їх.

**4.Вимоги до вихідних кодів та мов програмування**

Додаткові вимоги не пред'являються.

**5. Вимоги до захисту інформації та програм**

Забезпечення обмеження доступу до функціоналу сайту різним користувачам системи.

**6. Спеціальні вимоги**

Система повинна мати інтуїтивно-зрозумілий, простий інтерфейс.

**ДОДАТОК Б ІНТЕРФЕЙС КОРИСТУВАЧА СИСТЕМИ**

Рисунок Б.1 – Інтерфейс користувача системи:

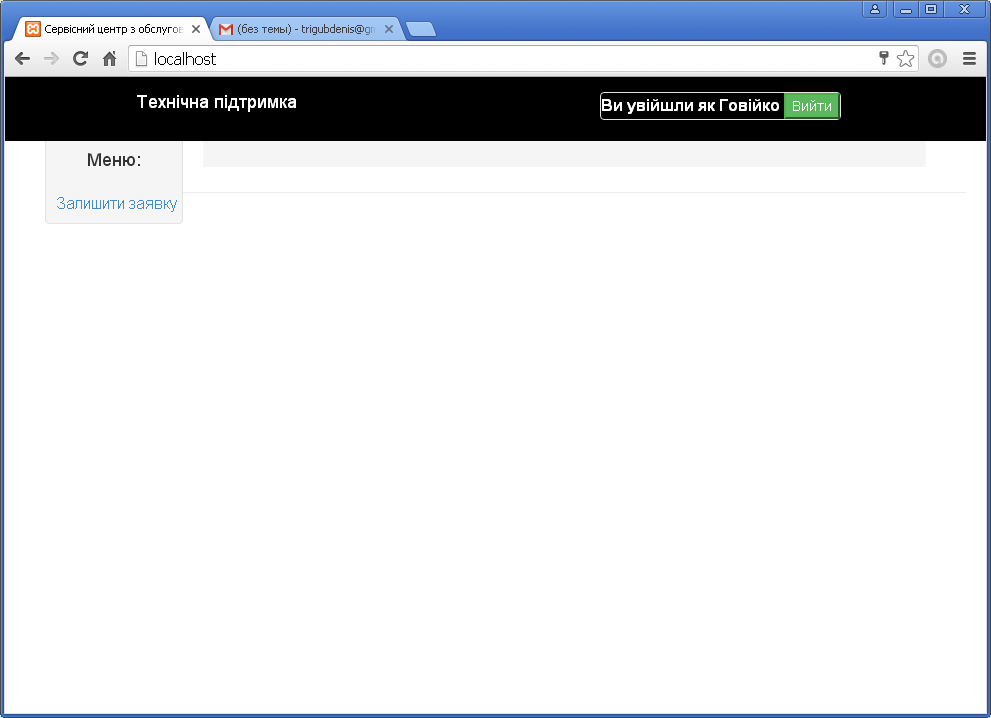


Рисунок Б.2 – Інтерфейс користувача(приклад подачі заявки):

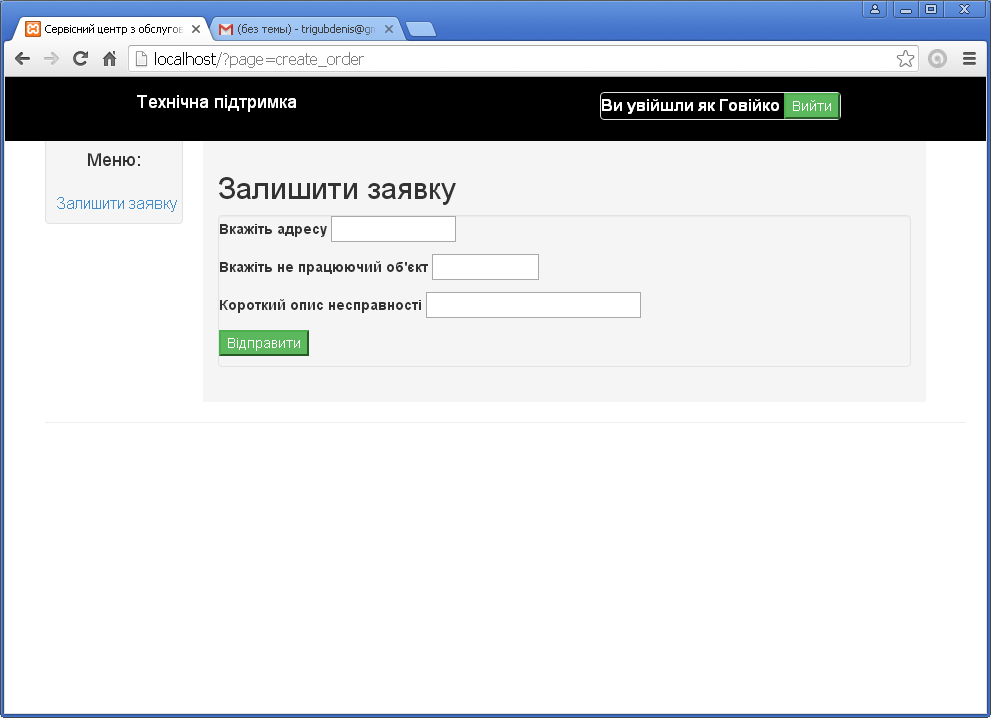


Рисунок Б.3 – Інтерфейс користувача(приклад подачі інформації списком):

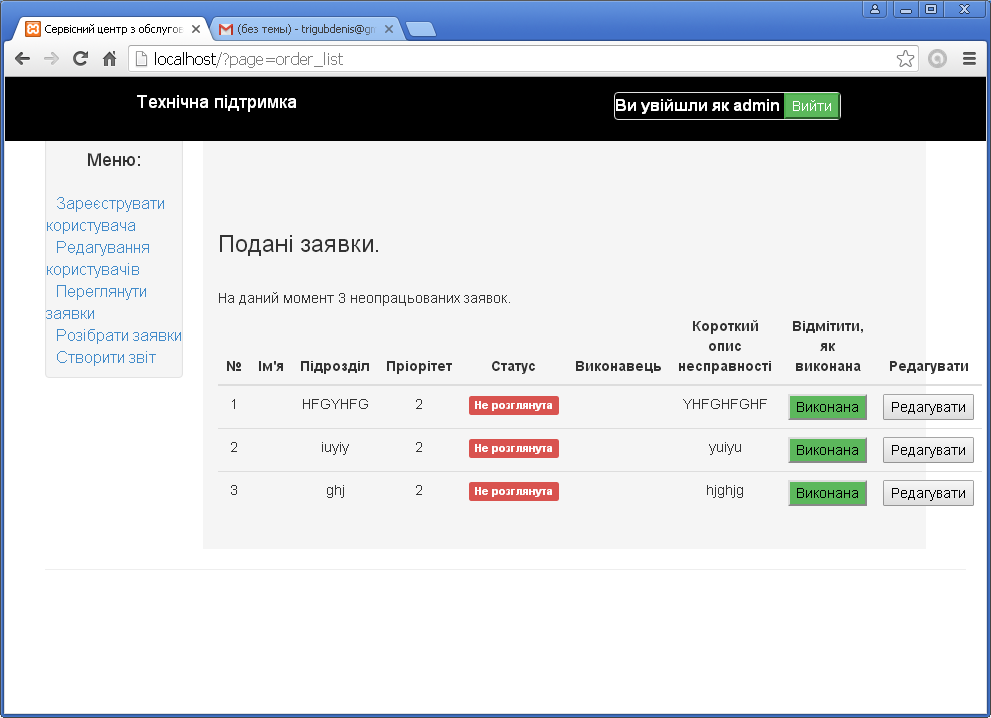
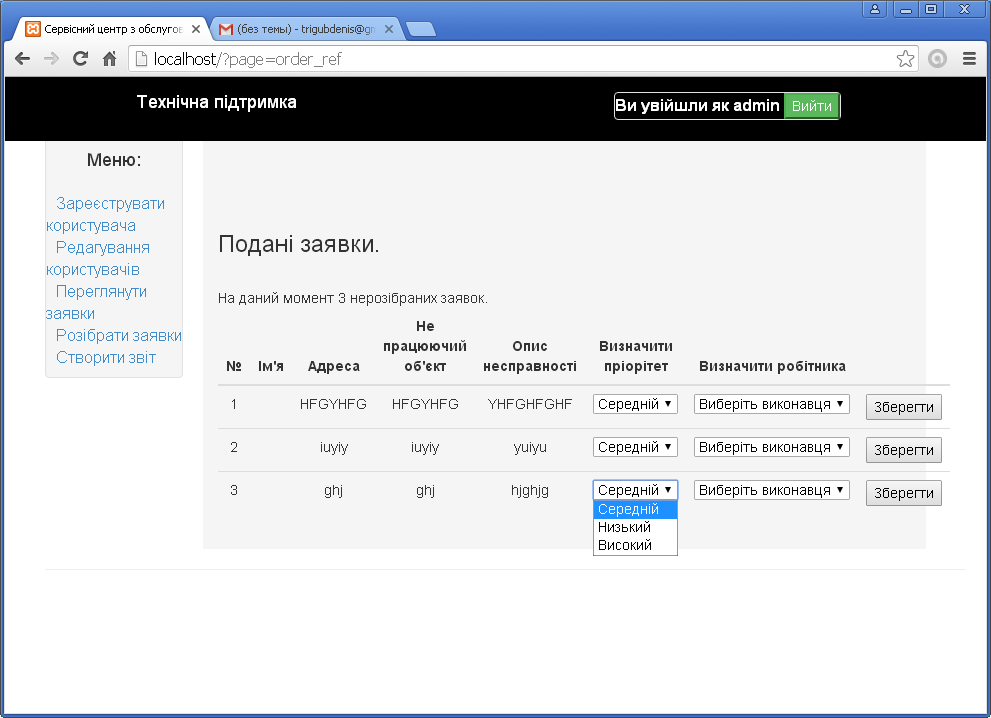


Рисунок Б.4 – Інтерфейс користувача(приклад призначення виконавця та пріорітета):



**ДОДАТОК В ВИХІДНІ КОДИ СИСТЕМИ**

1. **Функціонал (сторінка авторизації)**

<?php

session\_start();

require\_once "core/bootstrap.php";

?>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="template/css/bootstrap.css" rel="stylesheet">

<title>Авторизація</title>

</head>

<body>

<div class="navbar-line pos-top marg-auto">

<span class="top-span text-center marg-auto">ТОВ «СП»ОХОРОНА»</span>

</div>

<div class="login">

<h3>Будь-ласка, авторизуйтесь</h3>

<form method="post" action="login\_user.php" class="well form">

<p>

<label>Введіть логін:<br></label>

<input name="login" type="text" size="15" maxlength="15">

</p>

<p>

<label>Пароль користувача:<br></label>

<input name="password" type="password" size="15" maxlength="15">

</p>

<input class="btn-large btn-success" type="submit" value="Зайти">

</p>

</form>

</div>

</body>

</html>

1. **Інтерфейс користувача**

<?php

session\_start();

require\_once "core/bootstrap.php";

requiresAuthorized();

function getWay(){

if (isset($\_GET['page'])){

return $\_GET['page'];

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta charset="utf-8">

<title>ТОВ"СП"ОХОРОНА"</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content=" ТОВ"СП"ОХОРОНА"">

<link href="template/css/bootstrap.css" rel="stylesheet">

<style type="text/css">

body {

padding-top: 60px;

padding-bottom: 40px;

}

.sidebar-nav {

padding: 9px 0;

}

</style>

<link href="template/css/bootstrap-responsive.css" rel="stylesheet">

</head>

<body style="background-color: #fff">

<div class="navbar navbar-line navbar-fixed-top">

<div class="container">

<a class="navbar-brand" href="/">Технічна підтримка</a>

<div class="navbar-collapse">

<span class="navbar-text pull-right bgblack">

<?php

include "views/profile\_top.php";

?>

</span>

</div>

</div>

</div>

<div class="container-fluid">

<div class="row-fluid" >

<div class="main-menu fleft">

<div class="well sidebar-nav" >

<ul class="nav nav-list">

<li class="nav-header text-center">Меню:</li><br/>

<?php

include "menu.php";

?>

</ul>

</div>

</div>

<div class="main-content">

<div class="content">

<?php

renderPartial();

?>

</div>

</div>

</div>

<hr>

</div>

</body>

</html>

**3. Функціонал (сторінка реєстрації користувача)**

<?php

require\_once "core/bootstrap.php";

if(getAccessLevel()==3){

?>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<title>Реєстрація</title>

</head>

<body>

<h2>Реєстрація</h2>

<form action="?page=save\_user" method="post" class="well form">

<!--\*\*\*\* save\_user.php - это адрес обработчика. То есть, после нажатия на кнопку \"Зарегистрироваться\", данные из полей отправятся на страничку save\_user.php методом \"post\" \*\*\*\*\* -->

<p>

<label>Логін користувача:\*<br></label>

<input name="login" type="text" size="15" maxlength="15">

</p>

<!--\*\*\*\* В текстовое поле (name=\"login\" type=\"text\") пользователь вводит свой логин \*\*\*\*\* -->

<p>

<label>Пароль користувача:<br></label>

<input name="password" type="password" size="15" maxlength="15">

</p>

<!--\*\*\*\* В поле для паролей (name=\"password\" type=\"password\") пользователь вводит свой пароль \*\*\*\*\* -->

<p>

<label>Ім'я користувача:\*<br></label>

<input name="name" type="text" size="15" maxlength="15">

</p>

<!--\*\*\*\* В этом поле хранится имя пользователя \*\*\*\*\* -->

<p>

<label>Прізвище користувача:\*<br></label>

<input name="sname" type="text" size="15" maxlength="15">

</p>

<!--\*\*\*\* В этом поле хранится фамилия пользователя \*\*\*\*\* -->

<p>

<label>email користувача:<br></label>

<input name="email" type="text" size="25" maxlength="25">

</p>

<!--\*\*\*\* В этом поле хранится адрес электронной почти пользователя \*\*\*\*\* -->

<p>

<label>Підрозділ користувача:\*<br></label>

<input name="address" type="text" size="75" maxlength="100">

</p>

<!--\*\*\*\* В этом поле хранится домашний адрес пользователя \*\*\*\*\* -->

<p>

<label>Користувач:\*<br></label>

<input name="access" type="radio" value="1" checked>

</p>

<!--\*\*\*\* Пользователь \*\*\*\*\* -->

<p>

<label>Співробітник:\*<br></label>

<input name="access" type="radio" value="2" >

</p>

<!--\*\*\*\* Пользователь \*\*\*\*\* -->

<p>

<input class="btn-large btn-success" type="submit" value="Зареєструвати">

<!--\*\*\*\* Кнопочка (type=\"submit\") отправляет данные на страничку save\_user.php \*\*\*\*\* -->

</p>

</form>

</body>

</html>

<?php }

else { ?>

<div class="alert alert-block">

<a class="close" data-dismiss="alert"></a>

<h4 class="alert-heading">Увага!</h4>

Ви не маєте доступу до реєстрації користувачів, або не увійшли до сайту.

</div>

<?php }

?>

**4. Функціонал (додавання користувачів в базу)**

<?php

if(getAccessLevel()==3){

if (isset($\_POST['login'])) { $login = $\_POST['login']; if ($login == '') { unset($login);} }

//заносим введенный пользователем логин в переменную $login, если он пустой, то уничтожаем переменную

if (isset($\_POST['password'])) { $password=$\_POST['password']; if ($password =='') { unset($password);} }

//заносим введенный пользователем пароль в переменную $password, если он пустой, то уничтожаем переменную

if (isset($\_POST['email'])) { $email=$\_POST['email']; if ($email =='') { unset($email);} }

//заносим введенный пользователем email в переменную $email, если он пустой, то уничтожаем переменную

if (isset($\_POST['name'])) { $name=$\_POST['name']; if ($name =='') { unset($name);} }

//заносим введенный пользователем name в переменную $name, если он пустой, то уничтожаем переменную

if (isset($\_POST['sname'])) { $sname=$\_POST['sname']; if ($sname =='') { unset($sname);} }

//заносим введенный пользователем sname в переменную $sname, если он пустой, то уничтожаем переменную

if (isset($\_POST['address'])) { $address=$\_POST['address']; if ($address =='') { unset($address);} }

//заносим введенный пользователем adress в переменную $adress, если он пустой, то уничтожаем переменную

if (isset($\_POST['access'])) { $access=$\_POST['access']; if ($access =='') { unset($access);} }

//заносим введенный пользователем adress в переменную $adress, если он пустой, то уничтожаем переменную

if (empty($login) or empty($password)) //если пользователь не ввел логин или пароль, то выдаем ошибку и останавливаем скрипт

{

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\" href=\"?page=reg\">×</a>

<h4 class=\"alert-heading\">Увага!</h4>

Ви ввели не всю інформацію необхідну для реєстрації. Закрийте це повідомлення та повторіть введення даних.

</div>\n";

$stop=1;

}

//если логин и пароль введены,то обрабатываем их, чтобы теги и скрипты не работали, мало ли что люди могут ввести

$login = stripslashes($login);

$login = htmlspecialchars($login);

$password = stripslashes($password);

$password = htmlspecialchars($password);

//удаляем лишние пробелы

$login = trim($login);

$password = trim($password);

// подключаемся к базе

include ("db.php");// файл db.php должен быть в той же папке, что и все остальные, если это не так, то прос-то измените путь

//mysql\_query('SET NAMES utf8');

// проверка на существование пользователя с таким же логином

echo mysql\_error();

$result = mysql\_query("SELECT id FROM users WHERE login='$login'",$db);

$myrow = mysql\_fetch\_array($result);

if (!empty($myrow['id'])&& $stop!=1) {

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\" href=\"?page=create\_user\">×</a>

<h4 class=\"alert-heading\">Увага!</h4>

Користувач з таким логіном вже існує.Виберіть інший логін.

</div>\n";

$stop=1;

}

if($stop!=1) {

// если такого нет, то сохраняем данные

$result2 = mysql\_query ("INSERT INTO users (login,password,email,name,sname,address,accesslvl) VALUES('$login','$password','$email','$name','$sname','$address','$access')");

}

// Проверяем, есть ли ошибки

if ($result2=='TRUE' && $stop!=1)

{

echo "<a class=\"close\" data-dismiss=\"alert\" href=\"?page=create\_user\">×</a><br/>

Реєстрація пройшла успішно!";

}

else {

if($stop!=1){

echo "<div class=\"alert alert-error\">

<a class=\"close\" data-dismiss=\"alert\" href=\"?page=create\_user\">×</a>

<h4 class=\"alert-heading\">Увага!</h4>

Не вдалось зареєструвати користувача. Помилка обробки бази даних.

</div>\n";

}

}

}

else {

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

<h4 class=\"alert-heading\">Увага!</h4>

Ви не маєте доступу до реєстрації користувачів, або не увійшли до сайту.

</div>\n";

}

?>

**5. Функціонал (редагування користувачів)**

<?php

session\_start();

require\_once "core/bootstrap.php";

if (isset($\_POST['Id'])) { $Id=$\_POST['Id']; }

$res=mysql\_query("SELECT \* FROM users WHERE id='$Id'");

$edit\_id=mysql\_result($res,0,'id');

$edit\_login=mysql\_result($res,0,'login');

$edit\_name=mysql\_result($res,0,'name');

$edit\_sname=mysql\_result($res,0,'sname');

?>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="template/css/bootstrap.css" rel="stylesheet">

</head>

<body>

<h3 style="padding-left:20px">Редагування заявки</h3><a style="padding-left:20px" class="button button-primary" <?php echo" onclick=\"history.back();\"" ?>> Повернутися на попередню сторiнку </a> </br>

<table class="table">

<thead>

<tr>

<th>Логiн</th>

<th>Ім'я</th>

<th>Прiзвище</th>

<th></th>

</tr>

</thead>

<tbody>

<tr>

<?php echo"<form method=\"post\" action=\"user\_save\_changes.php\">

<input name=\"Id\" value=\"".$edit\_id."\" style=\"display:none\">

<td>

<input type=\"text\" value=\"$edit\_login\" name=\"login\"></td>

<td>

<input type=\"text\" value=\"$edit\_name\" name=\"name\"></td>

<td>

<input type=\"text\" value=\"$edit\_sname\" name=\"sname\"></td>

<td><input class=\"btn-large button-success\" type=\"submit\" value=\"Зберегти\"></form> </td>"; ?>

</tr>

</tbody>

</table>

</body>

</html>

**6. Функціонал (створення заявок)**

<?php

session\_start();

require\_once "core/bootstrap.php";

if(getAccessLevel()==3){

$query\_all = mysql\_query("select count(\*) from orders");

$count\_all = mysql\_fetch\_array($query\_all);

$count\_neroz = mysql\_query("select count(\*) from orders WHERE status='0'");

$count\_neroz = mysql\_fetch\_array($count\_neroz);

$query\_neroz = "SELECT \* FROM orders WHERE status='0'";

$res\_neroz = mysql\_query($query\_neroz);

$count\_sended = mysql\_query("select count(\*) from orders WHERE status='1'");

$count\_sended = mysql\_fetch\_array($count\_sended);

$query\_sended = "SELECT \* FROM orders WHERE status='1'";

$res\_sended = mysql\_query($query\_sended);

$count\_pol = mysql\_query("select count(\*) from orders WHERE status='2'");

$count\_pol = mysql\_fetch\_array($count\_pol);

$query\_pol = "SELECT \* FROM orders WHERE status='2'";

$res\_pol = mysql\_query($query\_pol);

$count\_done = mysql\_query("select count(\*) from orders WHERE status='4'");

$count\_done = mysql\_fetch\_array($count\_done);

$query\_done = "SELECT \* FROM orders WHERE status='4'";

$res\_done = mysql\_query($query\_done);

if($count\_all[0]=='0'){

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

Заявки вiдсутнi

</div>\n";

}

else if ($count\_all[0]>'0' ){

echo "<h3>Всього заявок - ".$count\_all[0]."</h3></br>\n";

if($count\_neroz[0]=='0'){

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

На даний момент немає нерозглянутих заявок.

</div>\n";

}

if($count\_neroz[0]>'0') {

echo"Неопрацьованих заявок - ".$count\_neroz[0]."</br>";

?>

<table class="table">

<thead>

<tr>

<th>№</th>

<th>Ім'я</th>

<th>Адреса</th>

<th>Не працюючий об'єкт</th>

<th>Короткий опис несправності</th>

<th>Виконавець</th>

</tr>

</thead>

<tbody> <?php

for ($i='0'; $i<$count\_neroz[0]; $i++) {

$numb=$i+1;

echo " <tr>

<td>".$numb."</td>

<td>".mysql\_result($res\_neroz,$i,'name')."</td>

<td>".mysql\_result($res\_neroz,$i,'podrazdel')."</td>

<td>".mysql\_result($res\_neroz,$i,'object')."</td>

<td>".mysql\_result($res\_neroz,$i,'neispravnost')."</td>

<td>".mysql\_result($res\_neroz,$i,'ispolnitel')."</td>

</tr>";

}

echo"

</tbody>

</table>\n";

}

if($count\_sended[0]=='0'){

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

На даний момент немає вiдправлених заявок.

</div>\n";

}

if($count\_neroz[0]>'0') {

echo"Вiдправлених заявок - ".$count\_sended[0]."</br>";

?>

<table class="table">

<thead>

<tr>

<th>№</th>

<th>Ім'я</th>

<th>Адреса</th>

<th>Не працюючий об'єкт</th>

<th>Короткий опис несправності</th>

<th>Виконавець</th>

</tr>

</thead>

<tbody> <?php

for ($i='0'; $i<$count\_sended[0]; $i++) {

$numb=$i+1;

echo " <tr>

<td>".$numb."</td>

<td>".mysql\_result($res\_sended,$i,'name')."</td>

<td>".mysql\_result($res\_sended,$i,'podrazdel')."</td>

<td>".mysql\_result($res\_sended,$i,'object')."</td>

<td>".mysql\_result($res\_sended,$i,'neispravnost')."</td>

<td>".mysql\_result($res\_sended,$i,'ispolnitel')."</td>

</tr>";

}

echo"

</tbody>

</table>\n";

}

if($count\_pol[0]=='0'){

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

На даний момент немає прийнятих заявок.

</div>\n";

}

if($count\_pol[0]>'0') {

echo"Прийнятих заявок - ".$count\_pol[0]."</br>";

?>

<table class="table">

<thead>

<tr>

<th>№</th>

<th>Ім'я</th>

<th>Адреса</th>

<th>Не працюючий об'єкт</th>

<th>Короткий опис несправності</th>

<th>Виконавець</th>

</tr>

</thead>

<tbody> <?php

for ($i='0'; $i<$count\_pol[0]; $i++) {

$numb=$i+1;

echo " <tr>

<td>".$numb."</td>

<td>".mysql\_result($res\_pol,$i,'name')."</td>

<td>".mysql\_result($res\_pol,$i,'podrazdel')."</td>

<td>".mysql\_result($res\_pol,$i,'object')."</td>

<td>".mysql\_result($res\_pol,$i,'neispravnost')."</td>

<td>".mysql\_result($res\_pol,$i,'ispolnitel')."</td>

</tr>";

}

echo"

</tbody>

</table>\n";

}

if($count\_done[0]=='0'){

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

На даний момент немає виконаних заявок.

</div>\n";

}

else if($count\_pol[0]>'0') {

echo"Виконаних заявок - ".$count\_done[0]." </br>";

?>

<table class="table">

<thead>

<tr>

<th>№</th>

<th>Ім'я</th>

<th>Адреса</th>

<th>Не працюючий об'єкт</th>

<th>Короткий опис несправності</th>

<th>Виконавець</th>

</tr>

</thead>

<tbody> <?php

for ($i='0'; $i<$count\_pol[0]; $i++) {

$numb=$i+1;

echo " <tr>

<td>".$numb."</td>

<td>".mysql\_result($res\_done,$i,'name')."</td>

<td>".mysql\_result($res\_done,$i,'podrazdel')."</td>

<td>".mysql\_result($res\_done,$i,'object')."</td>

<td>".mysql\_result($res\_done,$i,'neispravnost')."</td>

<td>".mysql\_result($res\_done,$i,'ispolnitel')."</td>

</tr>";

}

echo"

</tbody>

</table>\n";

}

}

else {

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

<h4 class=\"alert-heading\">Увага!</h4>

Ви не маєте доступу до розгляду заявок на ремонт, або не увійшли до сайту.

</div>\n";

}

}

?>

**7. Функціонал (список заявок)**

<?php

session\_start();

require\_once "core/bootstrap.php";

if(getAccessLevel()==3){

echo "<h3>Подані заявки.</h3> </br>\n";

include "db.php";

$count = mysql\_query("select count(\*) from orders WHERE status='0' OR status='1' OR status='2'");

$count = mysql\_fetch\_array($count);

if($count[0]=='0'){

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

<h4 class=\"alert-heading\">Увага!</h4>

На даний момент немає нерозглянутих заявок.

</div>\n";

}

else if ($count[0]>'0' ){ //начало обработки заявления

echo "На даний момент ".$count[0]." неопрацьованих заявок.</br>"; //Подсчет кол-ва необработаных заявлений.

$query = "SELECT \* FROM orders WHERE status='0' OR status='1' OR status='2'";

$res = mysql\_query($query);

echo "<table class=\"table\">

<thead>

<tr>

<th>№</th>

<th>Ім'я</th>

<th>Підрозділ</th>

<th>Прiорiтет</th>

<th>Статус</th>

<th>Виконавець</th>

<th>Короткий опис несправності</th>

<th>Відмітити, як виконана</th>

<th>Редагувати</th>

</tr>

</thead>

<tbody>\n";

for ($i='0'; $i<$count[0]; $i++) {

$numb=$i+1;

echo " <tr>

<td>".$numb."</td>

<td>".mysql\_result($res,$i,'name')."</td>

<td>".mysql\_result($res,$i,'podrazdel')."</td>

<td>".mysql\_result($res,$i,'prioritet')."</td>";

$sta=mysql\_result($res,$i,'status');

if($sta=='0') {

echo" <td><span class=\"label label-danger\">Не розглянута</span></td>" ;

}

else if($sta=='1') {

echo" <td><span class=\"label label-warning\">Відправлена</span></td>" ;

}

else if($sta=='2') {

echo" <td><span class=\"label label-primary \">Прийнята</span></td>" ; }

echo" <td>".mysql\_result($res,$i,'ispolnitel')."</td>";

echo"<td>";

$string=mysql\_result($res,$i,'neispravnost') ;

echo mb\_substr($string, 0, 45, 'UTF-8');

echo"</td><td><form method=\"post\" action=\"order\_list\_done.php\">

<input name=\"Id\" value=\"".mysql\_result($res,$i,'Id\_orders')."\" style=\"display:none\">

<input class=\"btn-large label-success\" type=\"submit\" value=\"Виконана\">

</form></td>";

echo"<td><form method=\"post\" action=\"?page=edit\_order\">

<input name=\"Id\" value=\"".mysql\_result($res,$i,'Id\_orders')."\" style=\"display:none\">

<input class=\"btn-large button-success\" type=\"submit\" value=\"Редагувати\">

</form></td>";

}

echo"

</tbody>

</table>\n";

}

}

else {

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

<h4 class=\"alert-heading\">Увага!</h4>

Ви не маєте доступу до розгляду заявок на ремонт, або не увійшли до сайту.

</div>\n";

}

?>

**8. Функціонал (розбирання заявок)**

<?php

session\_start();

require\_once "core/bootstrap.php";

if(getAccessLevel()==3){

$query = "SELECT \* FROM orders WHERE (status='0' OR status='1' OR status='2') AND (ispolnitel is null OR ispolnitel='')";

$res = mysql\_query($query);

$isp\_list=mysql\_query("SELECT \* FROM users WHERE accesslvl='2'");

$isp\_count=mysql\_query("SELECT count(\*) FROM users WHERE accesslvl='2'");

$isp\_count = mysql\_fetch\_array($isp\_count);

$count = mysql\_query("select count(\*) from orders WHERE (status='0' OR status='1' OR status='2') AND (ispolnitel is null OR ispolnitel='')");

$count = mysql\_fetch\_array($count);

if($count[0]=='0'){

echo "

<h3>На даний момент немає нерозiбраних заявок</h3>";

}

else if ($count[0]>'0' ){

echo "<h3>Подані заявки.</h3> </br>\n";

echo "На даний момент ".$count[0]." нерозiбраних заявок.</br>";

?>

<table class="table">

<thead>

<tr>

<th>№</th>

<th>Ім'я</th>

<th>Адреса</th>

<th>Не працюючий об'єкт</th>

<th>Опис несправності</th>

<th>Визначити прiорiтет</th>

<th>Визначити робiтника</th>

<th></th>

</tr>

</thead>

<tbody>

<?php

for ($i='0'; $i<$count[0]; $i++) {

$numb=$i+1;

$id=mysql\_result($res,$i,'Id\_orders');

echo " <tr>

<td>".$numb."</td>

<td>".mysql\_result($res,$i,'name')."</td>

<td>".mysql\_result($res,$i,'podrazdel')."</td>

<td>".mysql\_result($res,$i,'object')."</td>";

echo"<td>";

$string=mysql\_result($res,$i,'neispravnost') ;

echo mb\_substr($string, 0, 45, 'UTF-8');

echo"</td>

<form method=\"post\" action=\"order\_ref\_save.php\">

<td><select name=\"prioritet\">

<option selected value=\"2\">Середнiй</option>

<option value=\"1\">Низький</option>

<option value=\"3\">Високий</option>

</select></td>

<td><select name=\"ispolnitel\">

<option selected disabled>Виберiть виконавця</option>";

for($j='0'; $j<$isp\_count[0]; $j++){

echo"<option value=\"".mysql\_result($isp\_list,$j,'name')."\">".mysql\_result($isp\_list,$j,'name')."</option>";

}

echo"</select>

</td>

<td>

<input name=\"Id\" value=\"".$id."\" style=\"display:none\">

<input class=\"btn-large button-success\" type=\"submit\" value=\"Зберегти\">

</form></td></tr>";

}

echo"

</tbody>

</table>\n";

}

}

else {

echo "<div class=\"alert alert-block\">

<a class=\"close\" data-dismiss=\"alert\"></a>

<h4 class=\"alert-heading\">Увага!</h4>

Ви не маєте доступу до розгляду заявок на ремонт, або не увійшли до сайту.

</div>\n";

}

?>

**ДОДАТОК Г ДОВІДКА ПРО ВИКОРИСТАННЯ**

ЗАТВЕРДЖУЮ

Директор ТОВ «СП»Охорона»

Токар А.В.

„ ”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016р.

ДОВІДКА

про використання програмного забезпечення,

розробленого у кваліфікаційній роботі бакалавра Шмиголь М.Ю.

Цим підтверджуємо, що результати кваліфікаційнойї роботи бакалавра Шмиголь М.Ю. "Автоматизована інформаційна система обліку обслуговування систем безпеки об’єктів підприємства" у вигляді програмного забезпечення отримані й використані в роботі підприємства ТОВ «СП»Охорона». Для збирання замовлень на обслуговування та обліку діяльності працівників по обслуговуванню систем безпеки на об’єктах, що обслуговує підприємства.

Директор ТОВ «СП»Охорона» Токар А.В.