

Детектор Плагиата v. 2808 - Отчёт оригинальности: 18.01.2025 16:18:07

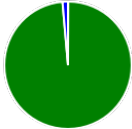
Проанализированный документ: Магістерська\_Шадрін\_П\_О (1)\_ітог.docx Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

Тип поиска: Дословный поиск Язык: Uk  
Тип проверки: Интернет  
ТЕЕ и кодировка: DocX n/a

Детальный анализ тела документа:

Диаграмма соотношения частей:

Плагиат 0.07%	Оригинал 98.57%
Кавычки 1.37%	ИИ 0%



Граф распределения зон:



Источники плагиата: 12

0.1%	22	1. <a href="https://simpentrec.com/mysql-introduction/">https://simpentrec.com/mysql-introduction/</a>
0.1%	15	2. <a href="https://ukrainiandigital.com/strong-movy-prohramuvannia-dlia-pochatktivsiv-ta-dosvidchenykh-ohliad-populiarnykh-strong/">https://ukrainiandigital.com/strong-movy-prohramuvannia-dlia-pochatktivsiv-ta-dosvidchenykh-ohliad-populiarnykh-strong/</a>
0.1%	16	3. <a href="https://foxminded.ua/chym-vidrizniaetsia-sait-vid-veb-dodatku/">https://foxminded.ua/chym-vidrizniaetsia-sait-vid-veb-dodatku/</a>

Детали обработанных ресурсов: 172 - ОК / 5 - Ошибок

Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
[не обнаружено]	[не обнаружено]	[не обнаружено]	

Античит-отчет UACE:

1. Статус: Анализатор <b>Включен</b> Нормализатор <b>Включен</b> сходство символов установлено на <b>100%</b>
2. Обнаруженный процент загрязнения UniCode: <b>10,1%</b> с лимитом: 4%
3. Процент нераспознанных символов после нормализации: <b>5,8%</b>
4. Все подозрительные символы будут отмечены фиолетовым цветом: <b>Abcd...</b>
5. Найдены невидимые символы: 0
Рекомендации по оценке: Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется передать это дело на более высокий уровень! В случае сомнений обращайтесь в службу поддержки Детектора плагиата!
Алфавитная статистика и анализ символов:

Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

Исключённые ресурсы:

URL не найдены



Включённые ресурсы:

URL не найдены

Детальный анализ документа:

Міністерство освіти і науки України Державний заклад	
<b>Цитування: 0,04%</b>	id: 1
«Луганський національний університет імені Тараса Шевченка»	
Навчально-науковий інститут математики та інформаційних технологій Кафедра інформаційних технологій та систем Шадрін Павло Олексійович РОЗРОБКА <b>CRUD</b> -ДОДАТКУ СИСТЕМНОГО АДМІНІСТРАТОРА НА ЗАСАДАХ PHP кваліфікаційна робота здобувача вищої освіти другого (магістерського) рівня освітньої програми	
<b>Цитування: 0,01%</b>	id: 2
«Мультимедійні системи»	
за спеціальністю 121	
<b>Цитування: 0,02%</b>	id: 3
„Інженерія програмного забезпечення”	
Особистий підпис Павло ШАДРІН Науковий керівник Світлана ПЕРЕЯСЛАВСЬКА, кандидат педагогічних наук, доцент кафедри інформаційних технологій та систем Завідувач кафедри Микола СЕМЕНОВ, кандидат педагогічних наук, доцент кафедри інформаційних технологій та систем Полтава 2025 ЗМІСТ ВСТУП6 РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ8 1.1 Загальна характеристика <b>CRUD</b> -додатків8 1.2 Вимоги для розробки <b>CRUD</b> -додатків10 1.2.1 Формат додатку10 1.2.2 Вимоги до безпеки12 1.2.3 Зручність використання13 1.3 Огляд сучасних інструментів для розробки <b>CRUD</b> -додатків13 1.4 Вибір формату додатку та інструментарію розробки31 Висновок до розділу 134 РОЗДІЛ 2. РЕАЛІЗАЦІЯ <b>CRUD</b> -ДОДАТКУ36 2.1 Підготовка середовища розробки36 2.2. Розробка бази даних додатку42 2.2.1. Створення бази даних додатку42 2.2.2. Загальний опис структури таблиць бази даних до нормалізації та їх призначення43 2.2.3. Детальний опис таблиць бази даних до нормалізації45 2.2.4 Нормалізація бази даних50 2.2.5. Підключення бази даних до проекту54 2.2.6. Організація взаємодії бази даних з додатком55 2.3 Розробка веб-інтерфейсу користувача60 2.3.1 Загальні принципи розробки веб-інтерфейсу додатку60 2.3.2. Детальний огляд розробки елементів веб-інтерфейсу та веб-сторінок додатку62 2.3.3. Реалізація функціональних кнопок на сторінках додатку72 2.3.4. Реалізація розмежування доступу користувачам за їх ролями80 2.3.5. Захист сторінок додатку від несанкціонованого доступу82 Висновок до розділу 284 ВИСНОВКИ86 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ88 ДОДАТОК А92 АНОТАЦІЯ Шадрін П. О. Тема: розробка <b>CRUD</b> -додатку системного адміністратора на засадах PHP. Спеціальність: 121	
<b>Цитування: 0,02%</b>	id: 4
"Інженерія програмного забезпечення".	
Установа: ДЗ ЛНУ імені Тараса Шевченка, 2025р. Кваліфікаційна робота містить: 104 стор., 40 рис., 1 табл, 36 джерел, 8 додатків. Об'єкт дослідження: <b>CRUD</b> -додаток системного адміністратора, з урахуванням необхідних функціональних можливостей, зручності використання та безпеки даних. Предмет дослідження: розробка <b>CRUD</b> -додатку системного адміністратора з урахуванням необхідних функціональних можливостей, зручності використання та безпеки даних. Мета роботи: розробити програмний додаток, що буде мати увесь необхідний функціонал для автоматизації ведення документації та обліку при роботі системного адміністратора. Методи дослідження: теоретичні: аналіз предметної області дослідження; емпіричні: аналіз власного досвіду роботи, порівняльний аналіз функціональних можливостей додатків схожого призначення, практичний: розробка програмного додатку. Результати роботи: досліджено предметну область та сформовано вимоги до додатку системного адміністратора, на основі чого розроблено програмний продукт, що відповідає даним вимогам та дає змогу автоматизувати процес документації роботи системного адміністратора. Ключові слова: системне адміністрування, програмування, <b>CRUD</b> -додаток, база даних, операції з базою даних. <b>ABSTRACT Shadrin P. Theme: development of CRUD system administrator application based on PHP. Specialty: 121</b>	
<b>Цитування: 0,01%</b>	id: 5
"Software Engineering".	
<b>Institution: Taras Shevchenko National University of Luhansk, 2025. Qualification work contains: 104 Pages., 40 fig., 1 table, 36 sources, 8 appendixes. The object of the study: CRUD application of the system administrator, taking into account the necessary functionality, ease of use and data security. The subject of the study: the development of the CRUD application of the system administrator, taking into account the necessary functionality, ease of use and data security. The purpose of the work: to develop a software application that will have all the necessary functionality for automating documentation and accounting during the work of a system administrator. Research methods: theoretical: analysis of the subject area of research; empirical: analysis of one's own work experience, comparative analysis of the functionality of applications of similar purpose, practical: development of a software application. Work results: the subject area was researched and the requirements for the system administrator's application were formed, on the basis of which a software product was developed that meets these requirements and makes it possible to automate the process of documenting the system administrator's work. Keywords: system administration, programming, CRUD application, database, database operations.</b> ВСТУП У сучасному світі інформаційних технологій роль системного адміністратора є ключовою для забезпечення стабільної та безперервної роботи інформаційної інфраструктури компаній, організацій та установ. Робота системних адміністраторів напряму пов'язана з необхідністю швидко, зручно та безпечно зберігати велику кількість різноманітних і у більшості випадків конфіденційних даних або інформації про мережу, що обслуговується адміністратором, та організацію вцілому. Це можуть бути паролі, облікові записи користувачів, особисті дані працівників та інша конфіденційна інформація, що необхідна системному адміністратору при роботі. Такий обсяг інформації фізично не можливо запам'ятати, тому постає необхідність ведення документації та її збереження. У цьому випадку у нагоді можуть стати саме <b>CRUD</b> -додатки. Ефективне управління серверами, користувачами, мережами та іншими <b>IT</b> -ресурсами потребує зручних та функціональних інструментів, які дозволяють автоматизувати рутинні завдання ведення документації та обліку та збити їх більш зручними. Отже, <b>CRUD</b> -додатки можуть стати потужним інструментом, котрий може суттєво підвищити ефективність роботи адміністратора або цілої команди людей, що обслуговують певну IT-інфраструктуру. Можливість одночасної роботи з однією і тією самою інформацією є ще одною перевагою <b>CRUD</b> -систем у порівнянні з більш	
<b>Цитування: 0,01%</b>	id: 6
«традиційними»	
методами ведення документації – на папері або на власному комп'ютері. Також власний <b>CRUD</b> -додаток організації має перевагу і в тому, що при розташуванні його на серверах всередині власної локальної мережі, інформація, з якою буде працювати додаток, не зможе потрапити назовні (за умови правильності захисту серверу та мережі), оскільки	

весь трафік буде проходити лише у межах локальної мережі без виходу до мережі Інтернет. Підхід, коли сервери організації

<p> Цитування: <b>0,01%</b></p>	id: 7
«ховають»	
<p>від зовнішньої мережі – суттєво підвищує безпеку та конфіденційність даних, що у свою чергу запобігає потенційним атакам та витікам інформації назовні. Об'єкт дослідження: <b>CRUD</b>-додаток системного адміністратора, з урахуванням необхідних функціональних можливостей, зручності використання та безпеки даних. Предмет дослідження: розробка <b>CRUD</b>-додатку системного адміністратора з урахуванням необхідних функціональних можливостей, зручності використання та безпеки даних. Мета роботи: розробити програмний додаток, що буде мати необхідний функціонал для автоматизації ведення документації та обліку при роботі системного адміністратора. Для досягнення мети необхідно: 1. Провести аналіз предметної області; 2. Визначити вимоги до додатку. 3. На основі аналізу предметної області обрати необхідний для розробки інструментарій. 4. Виконати розробку програмного продукту з урахуванням вимог. 5. Реалізація захисту розробленого додатку від несанкціонованого стороннього доступу. Методи дослідження: теоретичні: аналіз предметної області дослідження; емпіричні: аналіз власного досвіду роботи, порівняльний аналіз функціональних можливостей додатків схожого призначення, практичний: розробка програмного додатку. Результати роботи: досліджено предметну область та сформовано вимоги до додатку системного адміністратора, на основі чого розроблено програмний продукт, що відповідає даним вимогам та дає змогу автоматизувати процес документації роботи системного адміністратора. РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ Загальна характеристика <b>CRUD</b>-додатків <b>CRUD</b>-додатки (від англ. <b>Create, Read, Update, Delete</b>) — це інформаційні системи, що забезпечують базові операції управління даними в базах даних. Ці операції є фундаментальними для роботи з базами даних або іншими структурами зберігання інформації, що забезпечують можливість взаємодії з даними у зручній та структурованій формі[12]. Основні функціональні можливості <b>CRUD</b>-додатку Оскільки обсяги необхідної для роботи інформації постійно зростають, постає питання її безпечного та зручного зберігання. Одним з таких варіантів є бази даних. У сучасному світі практично кожна сфера діяльності використовує бази даних у своїй роботі для тих чи інших потреб, не є винятком і системні адміністратори. <b>CRUD</b>-додатки дозволяють зручно та безпечно працювати з великими обсягами інформації. Вони реалізують чотири основні операції роботи з базою даних, а саме: <b>Create</b> (створення): операція, яка дозволяє додавати нові записи до бази даних. Наприклад, у системах адміністрування це може бути створення нового користувача, додавання пристрою до обліку чи створення нового завдання. <b>Read</b> (читання): операція перегляду або пошуку інформації в базі даних. Вона може бути реалізована у вигляді списків, таблиць, фільтрів, пошуку за критеріями чи деталізації конкретного запису з подальшим виводом цієї інформації у додаток користувачу. <b>Update</b> (оновлення): операція редагування наявних записів у базі даних. Прикладом операції <b>update</b> може бути: зміна паролю користувача у БД або привласнення завданню статусу виконаного. <b>Delete</b> (видалення): операція видалення записів із бази даних, які більше не актуальні або застарілі. Прикладом може бути видалення з таблиці користувачів співробітника, що звільнився з організації і дані про нього більше не потрібні для зберігання.[13] Роль <b>CRUD</b>-додатку в роботі системного адміністратора Робота системного адміністратора передбачає роботу з великим обсягом різноманітної інформації, що необхідна у робочому процесі. У більшості випадків, ця інформація складна або неможлива до людського запам'ятовування через її кількість або зміст, наприклад – вкрай складно запам'ятати пароль, що складається з 20-24 випадково згенерованих символів, а якщо таких паролів у роботі використовується більше ніж 30-50 одиниць – задача стає неможливою [15]. Також у більшості випадків інформація має конфіденційний характер та не повинна бути розповсюджена для сторонніх осіб, оскільки це може призвести до вкрай серйозних наслідків, аж до повного програмного руйнування IT-інфраструктури організації. У такому випадку постає питання швидкого, зручного та конфіденційного зберігання інформації. Бази даних є вирішенням цього питання, оскільки дозволяють безпечно та надійно зберігати інформацію різного характеру. Приклади інформації, що необхідна системному адміністратору у роботі: Ведення переліку облікових записів користувачів, їх паролів та інш.; Ведення обліку IT-інфраструктури організації, її характеристик, налаштувань, розміщення та інш.; Ведення документації про мережу, сервери та іншу IT-інфраструктуру;[16] З описаного вище можна зробити висновок, що використання баз даних є невід'ємною частиною роботи системного адміністратора. У такому випадку <b>CRUD</b>-додатки відіграють ключову роль у діяльності системного адміністратора, забезпечуючи автоматизацію рутинних завдань, управління інформацією та спрощення адміністрування системи. В умовах сучасного IT-середовища, де обсяги даних постійно зростають, а складність систем збільшується, <b>CRUD</b>-додатки стають необхідним інструментом для ефективної роботи. Вимоги для розробки <b>CRUD</b>-додатків 1.2.1 Формат додатку На теперішній час, з урахуванням наявних технологій розробки, вимог до програмного забезпечення та інших факторів, можна виокремити два можливі формати <b>CRUD</b>-додатку: <b>web</b>-додаток та <b>desktop</b>-додаток, кожен з яких має свої переваги та недоліки, основні з яких будуть розглянуті у цьому розділі. По-перше необхідно визначити поняття, що таке <b>web</b>-додаток та <b>desktop</b>-додаток, щоб у подальшому мати змогу чітко та вірно порівнювати їх переваги та недоліки у контексті розробки <b>CRUD</b>-додатку системного адміністратора. <b>Desktop</b>-додаток (настільний додаток) — це програмне забезпечення, яке встановлюється та запускається безпосередньо на комп'ютері користувача, використовуючи ресурси його операційної системи (процесор, пам'ять, дисковий простір). Ці програми створюються для роботи на певній платформі, наприклад <b>Windows</b>, <b>macOS</b> або <b>Linux</b>[18]. Основні переваги <b>Desktop</b>-додатку: Оптимізований для локального використання, працює швидко завдяки доступу до апаратних ресурсів комп'ютера. Менший обсяг обчислень на сервері, оскільки обробка даних виконується на стороні клієнта. Підтримка роботи з локальними службами (наприклад, <b>PowerShell</b>, <b>SSH</b>-клієнти). Основні недоліки <b>Desktop</b>-додатку: Може бути прив'язаний до конкретної операційної системи, що ускладнює масштабування та використання на різних платформах. Встановлення оновлень вимагає дій від користувача (ручне оновлення або оновлення через інстальатор). Версійний контроль складніший порівняно з веб-додатками. Для кожного клієнтського ПК необхідно налаштувати програму та забезпечити відповідність системним вимогам. Складність реалізації мобільних версій додатку для смартфонів або планшетів, через те, що потребує створення окремого додатку для систем <b>iOS</b> та <b>Android</b>. <b>Web</b>-</p>	
<p> Обнаружен Плагіат: <b>0,07%</b> <a href="https://foxminded.ua/chym-vidrizniaietsia-sait-v-...">https://foxminded.ua/chym-vidrizniaietsia-sait-v-...</a> + 3</p>	id: 8
<p><b>desktop</b> (веб-додаток) — це програмне забезпечення, яке працює у веб-браузері та доступне через інтернет або локальну мережу. Для його використання не потрібно встановлювати жодне додаткове програмне забезпечення на пристрій користувача, окрім веб-браузера. При використанні веб-додатку усі обчислення у ході виконання програми, відбуваються на сервері, а на комп'ютер користувачів передається лише результат[19]. Основні переваги <b>Web</b>-додатку: Може використовуватися з будь-якого пристрою через веб-</p>	

браузер (ПК, смартфон, планшет). Легкий доступ з будь-якого місця, де є підключення до мережі. Не вимагає встановлення додаткового програмного забезпечення на клієнтській стороні. Оновлення автоматично доступні всім користувачам. Працює на будь-якій операційній системі, яка підтримує сучасний веб-браузер. Легко адаптується до великої кількості користувачів та нових функцій. Серверна архітектура дозволяє ефективніше обробляти запити. Основні недоліки **Web**-додатку: Необхідне стабільне інтернет-з'єднання для роботи з базою даних на сервері. Може мати затримки через мережеві обмеження або низьку швидкість. Дані передаються через мережу, що вимагає захищених протоколів передачі даних (**HTTPS**, **VPN**). Ризики атак на сервер додатку або базу даних. 1.2.2 Вимоги до безпеки **CRUD**-додаток системного адміністратора працює з чутливими даними та має високі вимоги до безпеки, адже він працює з даними напряму пов'язаними з безпекою мережі та даних користувачів. Забезпечення належного рівня захисту є критично важливим. Основні вимоги захисту: Наявність аутентифікації для входу в систему; Створення ролей користувачів та обмеження доступу відповідно до них; Забезпечення резервного копіювання як бази даних так і файлів додатку; Обмеження доступу до додатку через **IP**-адресацію та фаєрвол; Забезпечення захисту від **SQL**-ін'єкцій; Зберігання конфіденційних даних у зашифрованому вигляді. 1.2.3 Зручність використання Для забезпечення зручної та ефективної роботи користувачів **CRUD**-додатку необхідно дотримуватися існуючих стандартів та правил розробки інтерфейсу програми. Також слід враховувати досвід використання інших популярних додатків, що мають схожий або ідентичний функціонал, з урахуванням їх переваг та недоліків. Інтерфейс повинен бути інтуїтивно зрозумілим та простим у використанні. Елементи інтерфейсу мають бути послідовно та логічно розташовані для зручності їх використання під час роботи. Інтерфейс програми має давати змогу користувачам повноцінно та зручно виконувати усі основні дії, що передбачені розробником. Огляд сучасних інструментів для розробки **CRUD**-додатків У даному пункті буде проведено огляд основних сучасних інструментів, що можуть використовуватися для розробки **CRUD**-додатку системного адміністратора такі як: Інструменти для програмування веб-додатків – локальні веб-сервери, існуючі СКБД під веб-розробку, мови програмування веб-додатків, інструменти верстки та стилізації веб-сторінок. Інструменти для розробки **desktop**-додатків – технології розробки, СКБД та інш. У подальшому виходячи з аналізу інформації з даного розділу та попередніх, буде сформовано кінцевий формат додатку та інструменти, що будуть використані для його розробки. Огляд існуючих технологій для створення **desktop**-додатків Почнемо огляд технологій саме з **desktop**-додатків, оскільки вони є більш класичним, але вже певною мірою застарілим варіантом формату додатка. Огляд інструментів для розробки **desktop**-додатків у загальному випадку зводиться до вибору конкретної мови програмування, що буде використовуватись під час розробки додатку. Оскільки у більшості випадків, мови програмування, що підтримують розробку **desktop**-додатків, вже мають весь необхідний інструментарій, котрий мало відрізняється у різних мовах програмування, і у більшості випадків не потребують встановлення та застосування додаткових бібліотек або фреймворків. Основні мови програмування та їх технології, що використовуються для розробки **desktop**-додатків[20]: **C#** - використовується разом із платформою **.NET** для розробки **Windows**-додатків. Інструменти: **WPF**, **WinForms**. **C++** - Підходить для високопродуктивних, системних або кросплатформних додатків. Інструменти: **Qt**, **MFC**. **Java** - Кросплатформна мова з використанням **JavaFX** або **Swing** для створення графічного інтерфейсу. **Python** - Використовується для швидкої розробки додатків із графічним інтерфейсом. Інструменти: **PyQt**, **Tkinter**. Кожна з даних мов програмування та її інструменти можуть використовуватися для розробки **desktop**-додатків. Вони мають дуже схожий між собою функціонал. У такому випадку вибір залежить лише від знань конкретної мови програмування конкретним розробником або командою розробників. Огляд існуючих інструментів для розгортання локальних веб-серверів для розробки веб-додатків У даному розділі буде розглянуто основні існуючі локальні веб-сервери, що використовуються при розробці веб-додатків. Оскільки без встановленого локального веб-серверу розробка веб-додатку фізично не можлива – вибір веб-серверу є досить важливим кроком у проектуванні та розробці додатку. На даний час існує велика кількість локальних веб-серверів, кожен з яких має свої переваги та недоліки. Для визначення оптимального варіанту, необхідно провести короткий огляд популярних локальних веб-серверів під ОС **Windows**. 1. **Open Server Open Server Panel** — це програмний комплекс для розгортання локального веб-сервера на платформі **Windows**. Він включає в себе набір інструментів для веб-розробки, таких як веб-сервери **Apache** і **Nginx**, бази даних **MySQL/MariaDB**, і підтримує **PHP**, **Perl**, **Python**[21]. Основні переваги **Open Server**: Простота використання: Інтуїтивно зрозумілий інтерфейс, що підходить як для новачків, так і для досвідчених розробників. Гнучкість налаштувань: Можливість вибору версій **PHP**, **MySQL**, **MariaDB**, і веб-сервера. Портативність: Може працювати без інсталяції, наприклад, з **USB**-накопичувача. Розширений функціонал: Підтримка різних інструментів, таких як **Node.js**, **Memcached**, **Redis** тощо. Автоматизація: Просте керування проектами та автозапуск веб-серверів. Швидке розгортання: Мінімальні налаштування для запуску локального веб-сервера. Основні недоліки **Open Server**: Обмеження платформи: Працює лише на **Windows**, не підтримує **Linux** або **macOS**. Ресурсомісткість: Може використовувати значний обсяг оперативної пам'яті при запуску великих проєктів. Залежність від конфігурації ОС: Робота може ускладнитися через конфлікти з іншими локальними серверами або налаштуваннями мережі. 2. **XAMPP XAMPP** — це безкоштовний кросплатформний програмний пакет для розгортання локального веб-сервера[22]. Назва складається з аббревіатур: **X** — кросплатформність (**Windows**, **Linux**, **macOS**), **A** — **Apache** (веб-сервер), **M** — **MySQL/MariaDB** (база даних), **P** — **PHP**, **P** — **Perl**. **XAMPP** використовується для розробки, тестування і локального розгортання веб-додатків. Основні переваги **XAMPP**: Кросплатформність: Підтримує **Windows**, **macOS** і **Linux**. Комплексний набір інструментів: Включає все необхідне для веб-розробки — **Apache**, **MySQL/MariaDB**, **PHP**, **Perl**. Підтримка спільноти: Велика кількість документації, форумів і прикладів для вирішення проблем. Основні недоліки **XAMPP**: Ресурсомісткість: Може значно сповільнювати систему при роботі з великими проєктами. Мала безпека: Локальний сервер не налаштований для використання у виробничих умовах, що створює ризики для даних. Може викликати конфлікти: Конфлікти портів або сумісності з іншими локальними серверами. Не підтримує переносимість: Потребує інсталяції на конкретному пристрої, не можна запускати з **USB**. 3. **MAMP MAMP** — це програмний пакет для розгортання локального веб-сервера, який призначений для **macOS** і **Windows**. **MAMP** використовується для локальної розробки та тестування веб-додатків[23]. Переваги **MAMP**: Простота у використанні: Легко встановлюється та налаштовується, навіть для початківців. Можливість роботи на **macOS** і **Windows**: Особливо зручний для користувачів **macOS**. Гнучкість налаштувань: Підтримка різних версій **PHP**, можливість підключення додаткових модулів. Недоліки **MAMP**: Мала кросплатформність: Хоча доступна версія для **Windows**, основна орієнтація — **macOS**. Обмежений набір функцій у безкоштовній версії: Деякі важливі інструменти доступні лише у **MAMP Pro**. Менша популярність: У порівнянні з **XAMPP** чи **OpenServer**, **MAMP** має менш широкую спільноту. 4. **Стек технологій LAMP LAMP** — це набір програмного забезпечення для розгортання веб-серверу, який включає[24]: **L** — **Linux** (операційна система), **A** — **Apache** (веб-сервер), **M** — **MySQL/MariaDB** (система управління базами даних), **P** — **PHP** (мови програмування для



серверної логіки). [LAMP](#) використовується для створення, тестування та розгортання веб-додатків. Переваги [LAMP](#): Відкритий код: Усі компоненти є безкоштовними та з відкритим вихідним кодом. Гнучкість: Підтримка різних конфігурацій і можливість заміновати окремі компоненти (наприклад, [MariaDB](#) замість [MySQL](#)). Стабільність: [LAMP](#) є одним із найпопулярніших і перевірених стеків для веб-розробки. Широка спільнота: Велика кількість документації, навчальних матеріалів і активна підтримка. Придатність для розгортання на виробництві: Може використовуватися як для локальної розробки, так і для розгортання готових продуктів. Недоліки [LAMP](#): Складність налаштувань: У порівнянні з готовими пакетами, такими як [XAMPP](#) чи [OpenServer](#), [LAMP](#) вимагає ручного встановлення та конфігурації. Обмеження платформи: Працює виключно на [Linux](#). Для використання під ОС [Windows](#) необхідно встановлювати віртуальну машину з ОС [Лінукс](#) та на ній розгортати веб-сервер, що може бути не зручно. Потреба в адміністративних навичках: Для ефективного використання [LAMP](#) потрібні базові знання [Linux](#) та командного рядка. Менша інтеграція: Компоненти працюють автономно, що може ускладнити початкову настройку. Огляд існуючих мов програмування для розробки веб [CRUD](#)-додатків На відміну від [desktop](#)-версії [CRUD](#)-додатку, при розробці [web](#)-продукту вибір мови програмування грає велику роль. Це пов'язано з великим різноманіттям сучасних мов програмування, що можуть використовуватися для цього. Відповідно, кожна мова програмування має свою специфіку синтаксису, а також велику кількість бібліотек та фреймворків, що можуть функціонувати кардинально інакше у порівнянні один з одним. У даному розділі буде розглянуто основні мови програмування, що можуть бути використані для розробки веб [CRUD](#)-додатку. Буде розглянуто їх основні переваги та недоліки, а також конкретну специфіку кожної з них, щоб у подальшому спираючись на дану інформацію, обрати мову програмування, що задовольнить усім вимогам розробки. [PHP PHP](#) — це серверна мова програмування, розроблена для створення динамічних веб-сторінок. Вона інтегрується з [HTML](#) і використовується для роботи з базами даних, обробки форм, сесій, аутентифікації користувачів та інших серверних завдань.[25] Специфіка [PHP](#) Серверна мова: Код виконується на сервері, а користувач отримує вже готовий [HTML](#). Інтеграція з [HTML](#): [PHP](#) можна писати безпосередньо в [HTML](#)-коді. Робота з базами даних: Вбудована підтримка [MySQL](#), [PostgreSQL](#), [SQLite](#) тощо. Кросплатформність: Працює на різних операційних системах ([Windows](#), [Linux](#), [macOS](#)). Легка інтеграція: Проста у використанні разом із веб-серверами ([Apache](#), [Nginx](#)). Переваги [PHP](#) Простота у використанні: Підходить для початківців завдяки простому синтаксису. Широка підтримка: Більшість хостингів підтримують [PHP](#)

Цитування: 0,01%id: 9

"з коробки".

Велика спільнота: Доступно багато документації, форумів і готових рішень. Багата екосистема: Широкий вибір фреймворків ([Laravel](#), [Symfony](#), [CodeIgniter](#)). Гнучкість: Можливість швидкого створення прототипів і невеликих проєктів. Ефективна робота з базами даних: Підтримує різні типи БД із простим синтаксисом. Документація: [PHP](#) має одну з кращих, або якщо не найкращу офіційну документації Недоліки [PHP](#) Застарілі підходи: Старі версії мови не мали сучасних стандартів, через що часто зустрічаються проблеми з безпекою та застарілим кодом. Продуктивність: Менша швидкість виконання порівняно з новішими мовами (наприклад, [Go](#) або [Node.js](#)). Неуніверсальність: Основна сфера застосування — веб-додатки, що обмежує використання в інших галузях. Перевагненість фреймворками: Велика кількість рішень іноді ускладнює вибір оптимального. Проблеми зі складними проєктами: Не завжди підходить для масштабних або високонавантажених систем. [C# C#](#) — об'єктно-орієнтована мова програмування, розроблена компанією [Microsoft](#) у рамках платформи [.NET](#). Вона призначена для створення різноманітних додатків, включаючи десктопні, веб-додатки, ігри та мобільні рішення. [25] Специфіка [C#](#) Мова високого рівня: Має простий синтаксис, схожий на [C++](#) та [Java](#). Платформа [.NET](#): Тісно інтегрована з екосистемою [.NET](#), що забезпечує легку розробку для [Windows](#) та інших платформ. Об'єктно-орієнтованість: Дозволяє створювати модульний, багаторазовий і легко підтримуваний код. Кросплатформність: Завдяки [.NET Core](#) і [.NET 5+](#) підтримує розробку на [Windows](#), [Linux](#) та [macOS](#). Широкі можливості: Використовується для розробки веб-додатків ([ASP.NET](#)), ігор ([Unity](#)), серверного програмування, мобільних додатків ([Xamarin](#)) тощо. Переваги [C#](#) Простота та читабельність: Інтуїтивний синтаксис полегшує вивчення та роботу. Потужна екосистема: Інтеграція з [.NET](#) дозволяє використовувати безліч бібліотек і інструментів. Кросплатформність: Можливість створювати додатки для різних операційних систем. Безпека коду: Вбудовані механізми типізації та управління пам'яттю зменшують кількість помилок. Підтримка багатозадачності: Легке створення асинхронних і багатопоточних програм. Широка сфера застосування: Підходить для розробки веб-додатків, корпоративних систем, ігор, мобільних додатків. Недоліки [C#](#) Залежність від [Microsoft](#): Багато інструментів і фреймворків розроблено під [Windows](#). Високі вимоги до ресурсів: Не завжди оптимально підходить для невеликих або низькорівневих програм. Менша популярність за межами [Windows](#): Хоча кросплатформність покращується, конкуренти, як [JavaScript](#), мають більшу підтримку у веб-розробці. [Python Python](#) — це інтерпретована мова програмування високого рівня, яка відзначається простим синтаксисом і широкою сферою застосування. Вона ідеально підходить як для новачків, так і для досвідчених розробників. [25] Специфіка [Python](#) Інтерпретована мова: Код виконується без компіляції, що дозволяє швидко тестувати та змінювати програми. Простий синтаксис: Наближений до природної мови, що спрощує навчання та розробку. Універсальність: Використовується в різних сферах — від веб-розробки до аналізу даних, машинного навчання та автоматизації. Кросплатформність: Працює на [Windows](#), [macOS](#), [Linux](#) та інших платформах. Багата екосистема: Величезна кількість бібліотек та фреймворків (наприклад, [Django](#), [Flask](#), [Pandas](#), [NumPy](#)). Переваги [Python](#) Простота та зручність: Ідеально підходить для швидкого прототипування. Кросплатформність: Один і той самий код може працювати на різних операційних системах. Широка спільнота: Величезна кількість навчальних матеріалів, бібліотек і форумів. Універсальність: Використовується для веб-додатків, аналізу даних, розробки ШІ, скриптів тощо. Динамічна типізація: Зменшує необхідність вручну задавати типи змінних. Недоліки [Python](#) Низька швидкість виконання: Інтерпретована природа [Python](#) робить його повільнішим порівняно з компільованими мовами (наприклад, [C++](#) або [Java](#)). Високе споживання пам'яті: Не підходить для додатків, чутливих до ресурсів. Динамічна типізація: Хоча вона зручна, може призводити до помилок у великих проєктах. [Java Java](#) — це мова програмування загального призначення, відома своєю стабільністю, кросплатформністю та широким використанням у корпоративних рішеннях, мобільних додатках і веб-розробці. [26] Специфіка [Java](#) Кросплатформність: Принцип

Цитування: 0,03%id: 10

"Write Once, Run Anywhere"

([WORA](#)) забезпечується завдяки використанню [Java Virtual Machine \(JVM\)](#), що дозволяє виконувати код на будь-якій платформі з [JVM](#). Об'єктно-орієнтована мова: Побудована на принципах [ООП](#) (об'єктно-орієнтоване програмування), що сприяє модульності та повторному використанню коду. Суворі типізація: Всі змінні повинні мати чітко визначений

тип. Поширене використання: Використовується в [Android](#)-додатках, серверних застосунках, великих корпоративних системах, веб-додатках і наукових проектах. Потужна екосистема: Включає багаті бібліотеки та фреймворки ([Spring](#), [Hibernate](#), [JavaFX](#)). Переваги [Java](#) Кросплатформність: Програми, написані на [Java](#), можуть запускатися на будь-якій операційній системі з [JVM](#). Стабільність: Підходить для великих і довготривалих проектів. Висока продуктивність: Завдяки [JIT](#)-компілятору ([Just-In-Time](#)) і оптимізації виконання коду. Масштабованість: Добре підходить для розробки високонавантажених і масштабованих систем. Широка спільнота: Велика база знань, навчальних матеріалів і готових рішень. Недоліки [Java](#) Відносно високе споживання ресурсів: [Java](#)-програми потребують більше пам'яті та обчислювальних потужностей порівняно з деякими іншими мовами. Довший час запуску: [JVM](#) потребує часу для ініціалізації. Складність для початківців: Синтаксис [Java](#) може здатися складним для новачків. Менш гнучка для швидких прототипів: Порівняно з мовами типу [Python](#), [Java](#) вимагає більше коду для досягнення результату. Огляд існуючих СКБД для розробки бази даних [CRUD](#)-додатку [CRUD](#)-додаток за своєю сутністю передбачає постійну та тісну взаємодію з базою даних, а отже огляд існуючих сучасних СКБД з подальшим вибором однієї з них – є невід'ємною частиною розробки. У даному розділі буде оглянуто основні типи систем керування базами даних, їх особливості, переваги та недоліки. В подальшому спираючись на інформацію з даного розділу, враховуючи усі особливості проекту – буде зроблено вибір СКБД, що буде максимально задовольняти усім вимогам. Спочатку для чіткого подальшого розуміння необхідно визначити термін СКБД. СКБД (система керування базою даних) - це програмне забезпечення, яке використовується для створення, управління, організації та маніпулювання базами даних. Вона забезпечує доступ до даних для користувачів і додатків, а також управляє їх збереженням, захистом і обробкою.[27] Сучасні СКБД можна поділити на дві великі групи – реляційні та нереляційні. Реляційні СКБД — це системи керування базами даних, які організовують дані у вигляді таблиць (реляцій), що пов'язані між собою за допомогою ключів. Дані зберігаються у структурованому форматі, що полегшує їх обробку та запити за допомогою мови [SQL](#). [27] Нереляційні СКБД — це системи керування базами даних, які зберігають дані у форматах, відмінних від таблиць (наприклад, документи, ключ-значення, графи). Вони добре підходять для роботи з великими обсягами даних, що мають гнучку структуру, та забезпечують високу масштабованість. [27] Таблиця 1.3.1 Порівняння реляційних та нереляційних СКБД Характеристика Реляційні СКБД Нереляційні СКБД Структура даних Таблиці Гнучкі формати ([JSON](#), графи) Масштабованість Вертикальна Горизонтальна Швидкість Стабільна для транзакцій Вища для читання/запису Гнучкість Обмежена схемою таблиць Висока У даній роботі буде розглянуто лише реляційні СКБД, оскільки вони є класичним варіантом для розробки додатків такого формату, а також забезпечують високу стабільність роботи, що є необхідним під час роботи з важливими даними. Розглянемо основні існуючі реляційні СКБД та їх переваги та недоліки. [MySQL](#) [MySQL](#) — це реляційна система керування базами даних (СКБД), що базується на мові [SQL](#) ([Structured Query Language](#)). Вона є однією з найпопулярніших у світі завдяки своїй продуктивності, надійності та простоті використання. [MySQL](#) використовується для зберігання, організації та управління даними в додатках різного масштабу, від невеликих сайтів до великих корпоративних систем. [27] Переваги [MySQL](#) Простота використання: Зрозуміла структура та документація, що дозволяє швидко освоїти роботу з СКБД. Висока продуктивність: Оптимізована для швидкої роботи з великими обсягами даних. Масштабованість: Підходить як для невеликих проектів, так і для великих систем. Кросплатформність: Підтримує роботу на багатьох операційних системах, включаючи [Windows](#), [Linux](#) і [macOS](#). Відкритий код: Безкоштовна для більшості застосувань, із можливістю внесення змін у вихідний код. Широка підтримка: Велика спільнота, численні ресурси для навчання та вирішення проблем. Інтеграція: Легко інтегрується з популярними мовами програмування, як-от [PHP](#), [Python](#), [Java](#). Недоліки [MySQL](#) Обмежена функціональність порівняно з іншими СКБД: Менш гнучка в порівнянні з [PostgreSQL](#) у складних сценаріях. Недостатня підтримка складних запитів: Підтримує не всі типи запитів і аналітичних функцій, які доступні в інших СКБД. Обмеження ліцензування: Хоча основна версія безкоштовна, комерційні рішення можуть вимагати платної ліцензії. Переваги [PostgreSQL](#): Відкритий вихідний код : [PostgreSQL](#) є безкоштовною, з відкритим кодом і активно підтримується спільнотою розробників, що дозволяє гнучко налаштовувати систему під потреби конкретного проекту. Відповідність стандартам [SQL](#): [PostgreSQL](#) дотримується стандартів [SQL](#) ([ANSI SQL](#):2011) краще, ніж більшість інших СКБД, що забезпечує сумісність із багатьма додатками. Масштабованість і продуктивність: підтримує складні запити, великі обсяги даних і може ефективно працювати на кластерах. Добре підходить для аналітичних задач та [OLAP](#) ([Online Analytical Processing](#)). Підтримка складних типів даних : [PostgreSQL](#) працює з [JSON](#), [XML](#), масивами, геопросторовими даними (через розширення [PostGIS](#)) та іншими нестандартними типами даних, що робить її ідеальною для сучасних застосунків. Надійність і безпека : [PostgreSQL](#) забезпечує потужні механізми контролю доступу, реплікацію даних, підтримку транзакцій [ACID](#) та механізм для резервного копіювання і відновлення. Недоліки [PostgreSQL](#): Складність налаштування: при відсутності досвіду використання [PostgreSQL](#) може здатися складнішою у встановленні, налаштуванні та адмініструванні порівняно з іншими СКБД, наприклад, [MySQL](#). Вимогливість до ресурсів: [PostgreSQL](#) використовує більше пам'яті та ресурсів сервера, що може бути критичним для невеликих або обмежених за ресурсами систем. Швидкість роботи в деяких випадках: хоча [PostgreSQL](#) демонструє високу продуктивність, у простих операціях і запитах [MySQL](#) або [SQLite](#) можуть працювати швидше. Менша популярність : [PostgreSQL](#) менш популярна серед невеликих проектів і малого бізнесу, через що менше підручників або прикладів може бути доступно для певних специфічних завдань. Розширення можуть викликати складності: деякі розширення можуть конфліктувати або вимагати додаткових налаштувань, що ускладнює роботу для команд із невеликим досвідом. Огляд існуючих інструментів для верстки та розробки дизайну сторінок веб-додатків Розробка дизайну сторінок та їх верстка є невід'ємною частиною розробки веб-додатків. Від зрозумілості інтерфейсу додатку та якості його верстки залежить не менше, ніж від стабільності роботи серверної частини додатку, оскільки користувачі 100% свого часу, проведеного за роботою у додатку, взаємодіють саме з веб-інтерфейсом програми. Наразі в області розробки веб-сторінок та їх верстки монополістами є: [HTML](#), [CSS](#) та [JavaScript](#). Розглянемо кожного окремо. [HTML](#) [HTML](#) ([HyperText Markup Language](#)) — це стандартна мова розмітки, яка використовується для створення веб-сторінок і веб-додатків. [HTML](#) визначає структуру та зміст веб-документа за допомогою тегів, що дозволяє браузерам інтерпретувати ці дані й відображати їх для користувача.[28] Призначення [HTML](#): Створення структури веб-сторінки Здійснює навігацію між сторінками Дозволяє додавати зображення, відео, аудіо, інтерактивні елементи (теги [img](#) , [video](#) , [audio](#) ). Забезпечення семантики (наприклад, [header](#) , [footer](#) , [article](#) , [section](#) ) роблять структуру документа зрозумілішою для браузерів, пошукових систем та інших інструментів. Підтримки інтерактивності [CSS](#) [CSS](#) ([Cascading Style Sheets](#)) — це мова стилів, яка використовується для опису зовнішнього вигляду та форматування елементів на веб-сторінках, створених за допомогою [HTML](#). Вона дозволяє налаштовувати дизайн сторінки, включаючи кольори, шрифти, відступи, розміри, розташування елементів, а також створювати анімації та інші візуальні ефекти. [28] Головною особливістю [CSS](#) є каскадність,

що означає застосування стилів у порядку їхньої пріоритетності, залежно від походження та специфічності. Стили можна задавати кількома способами: через зовнішні файли, у самому [HTML](#)-коді або безпосередньо всередині тегів, що дозволяє розробникам гнучко керувати оформленням. [CSS](#) також підтримує наслідування, завдяки чому деякі стилі автоматично передаються дочірнім елементам. Ця технологія забезпечує розділення структури сторінки та її дизайну, полегшуючи розробку, оновлення та підтримку веб-проектів. [JavaScript](#) [JavaScript](#) — це високорівнева, динамічна мова програмування, яка використовується для створення інтерактивності на веб-сторінках. [28] Вона працює на стороні клієнта (у браузері) і дозволяє додавати динамічні елементи, такі як анімації, інтерактивні форми, спливаючі вікна, оновлення даних без перезавантаження сторінки (завдяки [AJAX](#)) та багато іншого. Завдяки [JavaScript](#) веб-сторінки стають не просто статичними, а функціональними та інтерактивними. Особливістю [JavaScript](#) є його інтеграція з [HTML](#) і [CSS](#), що дозволяє взаємодіяти з елементами сторінки, змінювати їхній вигляд чи поведінку в реальному часі. [JavaScript](#) також підтримує об'єктно-орієнтоване, функціональне та імперативне програмування, що робить його універсальним для різних завдань. Крім роботи в браузері, [JavaScript](#) використовується і на стороні сервера завдяки платформі [Node.js](#), що розширює його можливості на створення серверних застосунків, [API](#) та інших рішень. [JavaScript](#) підтримується всіма сучасними браузерами, що робить його одним із найважливіших інструментів веб-розробки.

#### 1.4 Вибір формату додатку та інструментарію розробки

Вибір формату додатку Спираючись на інформацію з попередніх пунктів даного розділу, можна зробити висновок, що розробляти [CRUD](#)-додаток системного адміністратора краще у форматі [web](#)-додатку, оскільки: [Web](#)-додаток запускається безпосередньо у браузері, а усі обчислення відбуваються на стороні сервера зменшуючи навантаження на ПК користувача, що може бути вигідно на слабких машинах. [Web](#)-додаток є кросплатформенним, оскільки всі обчислення відбуваються на стороні серверу, а клієнту виводиться лише результат. У такому випадку значно спрощується розробка додатку, через відсутність необхідності розробляти окрему версію додатку під кожну операційну систему. [Web](#)-додаток не потребує ручного оновлення, клієнт завжди працює з останньою наявною версією додатку, навідміну від [desktop](#)-версії, де користувачу необхідно власноруч встановити оновлення, що для деяких користувачів може бути складною задачею. [Web](#)-додаток не потрібно окремо інтегрувати під смартфони або планшети, використання яких може бути зручним та необхідним під час роботи системного адміністратора. Вибір локального веб-серверу для розробки Оскільки у пункті 1.4.1 цього документу, було обрано, що [CRUD](#)-додаток буде розроблятися у [web](#)-форматі, для подальшої роботи необхідно обрати, який локальний [web](#)-сервер буде використовуватися для розробки додатку. Виходячи з інформації з пункту 1.3.2 цього документу, було обрано, що у якості локального [web](#)-серверу для розробки програми буде використовуватися [Open Server](#) 6, через його простоту та швидкість конфігурації під ОС [Windows](#), а також можливості швидкого тестування роботи програмного продукту на різних версіях основних мов програмування та СКБД. Також перевагою [Open Server](#) 6 є можливість швидкого та просто перемикання у ході розробки між [web](#)-серверами [Apache2](#) та [Nginx](#), що має свої переваги та може бути досить зручно. Вибір мови програмування У ході аналізу існуючих сучасних мов програмування у пункті 1.3.3 цього документу, що можуть використовуватися для розробки веб-додатків, було сформовано, що одним з кращих варіантів вибору є мова програмування [PHP](#), оскільки: Є безкоштовною для використання. Має простий, лаконічний та зрозумілий синтаксис. Заточена саме під розробку веб-додатків. Для розробки може використовуватися у чистому вигляді без використання складних бібліотек або фреймворків, забезпечуючи при цьому весь необхідний функціонал. Має потужні вбудовані інструменти роботи з базами даних та веб-сторінками. Має можливість інтеграції [php](#)-коду безпосередньо до [html](#)-документу, що спрощує процес розробки. Має одну з найкращих офіційних документаций серед інших мов програмування, що можуть використовуватися для веб-розробки. Має широку популярність, велике ком'юніті розробників, а також велику кількість інформації в мережі інтернет. Також важливим є вибір версії [PHP](#), що буде використовуватися під час розробки. Виходячи з офіційної документатії, а також інформації з мережі Інтернет, було зроблено вибір на користь версії 7.4. Версія 7.4 не є досить новою, а тому демонструє значну стабільність роботи і може бути інтегрована з великою кількістю інших інструментів розробки. Вибір СКБД Враховуючи інформацію з пункту 1.3.4 цього документу, було прийнято рішення, що для розробки додатку будуть використовуватися, саме реляційні бази даних, оскільки для роботи з даними системному адміністратору необхідна саме стабільність роботи бази даних, а також на неї не буде великого навантаження у вигляді великої кількості різноманітних запитів від різних користувачів. Також враховуючи вибір мови програмування для розробки додатку у пункті 1.4.3 цього документу, а саме мови [PHP](#), у якості системи керування базами даних можна обрати будь-яку існуючу та популярну на даний час реляційну СКБД ([MySQL](#), [MariaDB](#), [PostgreSQL](#) та інш.), оскільки мова [PHP](#) на даний момент часу має інструменти для роботи з кожною з них. Але логічним вибором у даній ситуації є вибір СКБД [MySQL](#), оскільки вона входить до класичного набору стеку технологій [LAMP](#) ([Linux](#), [Apache2](#), [MySQL](#), [PHP](#)), що широко використовується для розробки веб-додатків та показує високу стабільність роботи, простоту підтримки та непогану швидкість роботи. Загалом [MySQL](#) є гарним вибором не тільки для роботи з [PHP](#), оскільки дана система керування базами даних є дуже популярною та перевіреною часом. Висновок до розділу 1 Аналіз предметної області є невід'ємною частиною будь-якої розробки, а тим паче, коли мова йде про IT-сферу, оскільки кількість наявних технологій, їх особливості та напрями застосування – грають важливу роль та напрями впливають на кінцевий результат роботи та його якість. У даному розділі було визначено поняття [CRUD](#)-додатку, його роль у роботі системного адміністратора та основні функціональні можливості. В подальшому розробка додатку буде проводитись, саме спираючись на дану інформацію. Також було сформовано основні вимоги до розробки додатку: основні формати у яких може бути реалізований додаток, вимоги до безпеки додатку та його застосування. У розділі було розглянуто основні інструменти, що можуть бути використані при розробці додатку, а саме: мови програмування, інструменти розробки [desktop](#)-додатків, локальні веб-сервери для розробки, системи керування базами даних, інструменти верстки та стилізації веб-сторінок. У результаті аналізу предметної області, проведення порівняльних аналізів та вибору оптимальних варіантів інструментарію, що буде використовуватися для розробки [CRUD](#)-додатку системного адміністратора було прийнято рішення використовувати: [Web](#)-формат додатку, що дає змогу спростити процес розробки, виключаючи необхідність розробляти окрему версію програмного забезпечення під кожну операційну систему окремо. [OpenServer](#) 6 у якості локального веб-серверу, що буде використовуватися для розробки додатку та його тестування перед подальшим перенесенням на реальну комерційну платформу, що буде забезпечувати достатній рівень стабільності та швидкодії. Мову програмування [PHP](#), у якості основної мови розробки. Оскільки вона є однією з найпопулярніших на даний момент, що використовуються для розробки веб-додатків, а також має увесь необхідний для цього функціонал. Систему керування базами даних [MySQL](#) у якості інструменту роботи з базами даних. [MySQL](#) є типовим вибором під час розробки додатків засобами мови програмування [PHP](#). У цілому, такий вибір інструментарію дозволяє забезпечити увесь необхідний функціонал для розробки стабільного та продуктивного додатку для системних адміністраторів.

Підсумовуючи – РОЗДІЛ 1 дав змогу, отримати необхідну кількість вичерпної інформації, що є необхідною при розробці [CRUD](#)-додатку системного адміністратора. В подальшому розробка буде проводитися, саме спираючись на інформацію, отриману у РОЗДІЛІ 1 цього документу. РОЗДІЛ 2. РЕАЛІЗАЦІЯ [CRUD](#)-ДОДАТКУ 2.1 Підготовка середовища розробки Налаштування [OpenServer Panel](#) 6 Першим кроком розробки будь-якого веб-додатку є налаштування локального веб-серверу. У перелік дій, що необхідно виконати на даному етапі розробки входять: Створення папки проекту та локального домену додатку; Внесення змін до конфігураційних файлів проекту; Вибір версій ПЗ, що буде використано; Підключення додаткових інструментів розробки. Розглянемо кожен крок детальніше. Створення папки проекту та локального домену додатку. У [OpenServer Panel](#) 6 створення додатку починається зі створення папки з проектом. Під кожен проект/додаток створюється окрема папка, назва якої повинна співпадати за майбутнім доменним ім'ям кінцевого продукту. Папки створюються за шляхом `\\Локальний_диск\\OSPanel\\home\\папка_проекту` та мають наступну структуру: Рис. 2.1. Структура каталогів проекту [OpenServer](#) 6 Каталог `.osp` містить лише один файл – [project.ini](#) [29]. Даний файл є основним конфігураційним файлом проекту, усі налаштування пов'язані з доступом, доменним ім'ям та інші налаштування проводяться у ньому. Його структуру та конфігурацію буде розглянуто нижче за текстом даної роботи. [29] Каталог `public` за замовчуванням є пустим. Основним його призначенням є збереження файлів додатку, що були написані розробником. Саме у цьому каталозі будуть міститися основні файли проекту – `html`, `css`, `js` та інші файли додатку. [29] Щоб створити проект, створимо за шляхом `\\Локальний_диск\\OSPanel\\home\\` папку з назвою `crud.local`, що буде відповідати, доменному імені веб-ресурсу, що буде розроблено у ході цього розділу. Також створимо необхідну структуру файлів та каталогів у відповідності до тексту вище. Також для спрощення роботи з додатком через веб-браузер, що корисно для розробки та тестування продукту, необхідно відредагувати файл `host` операційної системи `Windows`, що дасть змогу відкривати додаток за локальним доменним ім'ям, котре ми визначили вище, при створенні проекту. Даний файл знаходиться за шляхом `C:\\Windows\\system32\\drivers\\etc\\`. [29] У даний файл необхідно додати наступну строку: Рис. 2.2. Зміни у файлі `hosts` Внесення змін до конфігураційних файлів проекту. Після створення теки з проектом, необхідно створити конфігураційний файл [project.ini](#) за шляхом `\\Локальний_диск\\OSPanel\\home\\папка_проекту\\``.osp` та привести його до наступного вигляду: Рис. 2.3. Структура файлу [project.ini](#) У даному документі ключовими є поля[29]: `[crud.local]` – доменне ім'я веб-додатку, за яким буде здійснюватися доступ до нього через веб-браузер. `aliases` – те саме, що і попередній пункт, але з використанням `www` для помітки, що це саме веб-ресурс. `enabled` – чи є веб-ресурс включений на даний момент на веб-сервері. `php_engine` – версія `php`, що використовується для роботи даного веб-додатку. `project_dir` – локальний шлях на сервері до теки з проектом. `public_dir` – локальний шлях до теки, що містить файли веб-додатку, написані саме розробником. Вибір версій ПЗ, що буде використано Після дій, описаних вище за текстом даного документу, для продовження розробки додатку необхідно у налаштуваннях [OpenServer Panel](#) 6 підключити необхідні модулі, що будуть використовуватися під час розробки, а саме `PHP` та `MySQL`. У [OpenServer Panel](#) 6 дану дію можна виконати як використовуючи консоль так і графічний інтерфейс користувача. Використаємо останній варіант для підключення модулів необхідних нам версій – `MySQL` 8.2 та `PHP` 7.4. Рис. 2.4. Підключення необхідних модулів веб-серверу через `GUI` Підключення додаткових інструментів розробки Для продовження розробки необхідно підключити та налаштувати ще декілька важливих та потужних інструментів, що спрощують та пришвидшують розробку веб-додатку. Даними інструментами є – система контролю версій `Git` та онлайн-платформа `GitHub` та веб-інтерфейс керування базами даних `MySQL` `PhpMyAdmin`. Підключення `PhpMyAdmin` та його налаштування. `PhpMyAdmin` — це веб-застосунок, написаний на мові програмування `PHP`, який використовується для адміністрування баз даних `MySQL` і `MariaDB` через веб-інтерфейс. Це один із найпопулярніших інструментів для роботи з базами даних завдяки своїй зручності, широким можливостям і підтримці різних платформ. [30] У [OpenServer Panel](#) 6, навідміну від попередніх версій, `PhpMyAdmin` не є вбудованим інструментом, що йде з веб-сервером за замовчуванням. У даній версії його необхідно підключати окремо, налаштовуючи як окремий проект у теці `\\Локальний_диск\\OSPanel\\home\\`. [30] Після створення теки та необхідної структури каталогів, необхідно завантажити `PhpMyAdmin` з офіційного веб-сайту та перемістити завантажені файли до каталогу `public`. Після вищезазначених дій, необхідно провести подальші налаштування. У каталозі `public` необхідно знайти та перейменувати файл `config.inc.sample.php` на `config.inc.php`. Після чого у даному файлі необхідно знайти та відредагувати певні строки наступним чином: Рис. 2.5. Фрагмент конфігураційного файлу `config.inc.php` Де параметри позначають: `auth_type` =

Цитування: 0,01%

id: 11

`'config'`

– метод авторизації налаштовується вручну, а не використовується за замовчуванням. `host` =

Цитування: 0,01%

id: 12

`'MySQL-8.2'`

– версія `MySQL`, що використовується у додатку. `user` =

Цитування: 0,01%

id: 13

`'root'`

– ім'я користувача, під котрим буде здійснюватися підключення до БД додатку. `password` =

Цитування: 0,01%

id: 14

`''`

– пароль користувача, за допомогою якого буде здійснюватися підключення до БД (залишаємо пустим на час локальної розробки, з метою спрощення процесу). `AllowNoPassword` =

Цитування: 0,01%

id: 15

`'true'`

– дозволяє підключення до БД без парольного захисту (дозволяємо на час локальної розробки, з метою спрощення процесу) [30] Після виконання даних дій, необхідно у веб-браузері перейти за адресою `localhost/phpmyadmin` або `127.0.0.1/phpmyadmin`, якщо налаштування проведено коректно, відкриється веб-сторінка додатку: Рис. 2.6. Веб-сторінка додатку `phpmyadmin` Підключення системи контролю версій `Git` та онлайн платформи `GitHub` `Git` — це розподілена система керування версіями, яка використовується для відстеження змін у коді, співпраці між розробниками та управління розробкою програмного забезпечення. Вона дозволяє працювати над проектом одночасно кільком людям, зберігаючи історію змін і забезпечуючи можливість повернутися до будь-якої попередньої версії коду. `GitHub` — це хмарний сервіс для зберігання репозиторіїв `Git`, який також надає інструменти для спільної роботи, управління проектами та автоматизації



робочих процесів. Основна перевага цих сервісів – можливість спільної та одночасної роботи з кодом кількома різними розробниками. Але навіть при розробці додатку однією людиною, ці технології мають свої переваги при використанні, а саме: можливість працювати над кодом з декількох пристроїв (ПК, ноутбук тощо) без необхідності постійного копіювання та перенесення файлів, використовуючи зовнішні накопичувачі; використання [GitHub](#) у якості платформи для резервного копіювання, що допоможе зберегти файли проекту при виникненні фізичних несправностей ПК або ноутбука; можливість працювати з версіями додатку – наприклад у випадку, коли внесені зміни негативно вплинули на загальну працездатність додатку, завдяки системі [Git](#) можна легко та швидко повернути додаток до працездатної версії та продовжити розробку з урахуванням попереднього досвіду.

## 2.2. Розробка бази даних додатку

У процесі розробки інформаційних систем проектування бази даних є ключовим етапом, який забезпечує структуроване зберігання, обробку та доступ до даних. У межах даного дослідження особливу увагу приділено розробці бази даних для [CRUD](#)-додатку системного адміністратора, оскільки саме ефективна організація даних є запорукою стабільності роботи додатку. Метою є створення оптимальної моделі даних, яка забезпечить швидке виконання операцій додавання, редагування, видалення та перегляду інформації, а також відповідатиме вимогам безпеки та цілісності даних. У результаті побудована база даних слугуватиме основою для реалізації функціоналу додатку та забезпечення ефективного виконання його завдань.

### 2.2.1. Створення бази даних додатку

Робота з базою даних починається з її створення. Для операцій з базою даних буде використовуватися встановлений раніше [PhpMyAdmin](#), він має увесь необхідний функціонал для роботи з базами даних. Створимо базу даних додатку на локальному сервері, використовуючи графічний інтерфейс додатку та назвемо її [crud](#). Також одразу задамо кодування символів [utf8mb4\\_general\\_ci](#), для їх подальшого коректного зберігання та відображення.

### 2.2.2. Загальний опис структури таблиць бази даних до нормалізації та їх призначення

На основі власного досвіду роботи, а також з урахуванням досвіду використання інших систем зберігання інформації, було визначено, що кістяк даного додатку можна сформувати з бази даних, що у свої структурі містить 7 таблиць. Розглянемо призначення кожної з них:

1. Таблиця [users](#) – містить у собі інформацію про користувачів, що можуть працювати з додатком (логін, пароль, ім'я, рівень доступу та інш.).
2. Таблиця [ip\\_table](#) – є головною таблицею даного проекту, вона містить вичерпну інформацію про локальну мережу, що обслуговується системним адміністратором. Вона містить у собі інформацію про: айпі адреси пристроїв, доменні ім'я комп'ютерів та серверів, логіни та паролі користувачів та адміністраторів, MAC-адреси пристроїв, технічні характеристики пристроїв та їх інвентарну інформацію.
3. Таблиця [employees](#) – містить у собі інформацію про співробітників організації, таку як: ПІБ, номер телефону, електронну пошту, фізичне місцезнаходження робочого місця працівника у підрозділі/організації та інше.
4. Таблиця [corp\\_email](#) – містить у собі інформацію про корпоративні поштові скриньки співробітників організації, їх паролі та резервні коди доступу.
5. Таблиця [ip\\_phones](#) – містить у собі інформацію про IP-телефонію організації (абоненти, номери телефонів та інше.).
6. Таблиця [servers](#) – містить у собі розширену, у порівнянні з таблицею [ip\\_table](#), інформацію про сервери організації, а саме: їх призначення, особливості налаштувань, розміщення, технічні характеристики та інше.
7. Таблиця [manuals](#) – містить у собі різноманітні інструкції, що використовуються при налаштуванні мережі, обладнання тощо під час роботи системного адміністратора. Кожна з таблиць спроектована таким чином, щоб містити у собі вичерпну кількість інформації, необхідної системному адміністратору у роботі. Такий підхід збільшує ефективність роботи системного адміністратора, зменшуючи кількість запитів до бази даних зі сторони додатку, а також полегшує взаємодію користувача з інтерфейсом програми. На даному етапі розробки у проект додано основні і найнеобхідніші таблиці. По мірі продовження розвитку додатку, кількість таблиць може бути збільшена, враховуючи специфіку роботи та їх необхідність. Це демонструє гнучкість та масштабованість додатку, що в подальшому спростить його розвиток та підтримку.

### 2.2.3. Детальний опис таблиць бази даних до нормалізації

Після загального опису таблиць бази даних додатку, необхідно більш детально розглянути структуру кожної таблиці окремо. Це необхідно для чіткого розуміння функціонування окремих частин та усього додатку в цілому.

Таблиця [users](#) Як було сказано вище за текстом у розділі 2.2.2 таблиця [users](#) містить у собі інформацію про зареєстрованих користувачів, що можуть працювати у програмі. Для опису кожного користувача використовуються поля:

- [id](#) – тип даних [INT](#)(255) – дане поле є первинним ключем. Зберігає ідентифікатор запису про користувача у таблиці.
- [Login](#) – тип даних [VARCHAR](#)(100) – зберігає логін користувача у текстовому вигляді. За замовчуванням [NULL](#).
- [Password](#) – тип даних [VARCHAR](#)(255) – зберігає хешований пароль користувача у текстовому вигляді. За замовчуванням [NULL](#).
- [Full\\_name](#) – тип даних [VARCHAR](#)(400) – зберігає ПІБ користувача у текстовому вигляді. За замовчуванням [NULL](#).
- [Role](#) – тип даних [VARCHAR](#)(100) – зберігає роль користувача у текстовому вигляді. За замовчуванням [NULL](#).
- [Is\\_checked](#) – тип даних [TINYINT](#) – службове поле, для подальшого використання при створенні інтерфейсу користувача. Дозволяє відміти запис про користувача під час виводу у інтерфейс програми. За замовчуванням [NULL](#).

Таблиця [ip\\_table](#) Таблиця [ip\\_table](#), є основною таблицею з якою системний адміністратор буде найбільше взаємодіяти під час використання додатку. Дана таблиця зберігає у собі можливий максимум інформації про локальну мережу однієї або декількох організацій, що обслуговується системним адміністратором. Для зберігання необхідної інформації використовуються наступні поля:

- [id](#) – тип даних [INT](#) (255) – дане поле є первинним ключем. Зберігає ідентифікатор запису у таблиці.
- [IP](#) – тип даних [INT](#)(255) – зберігає у собі IP-адресу пристрою, запис про котрий зберігається у базі даних. За замовчуванням [NULL](#).
- [Full\\_name](#) – тип даних [VARCHAR](#) (255) – ім'я користувача або назва відділу, що використовують даний пристрій. За замовчуванням [NULL](#).
- [Username\\_pass](#) – тип даних [VARCHAR](#)(255)– містить назву пристрою, а також логін користувача та пароль облікового запису без прав адміністратора. За замовчуванням [NULL](#).
- [MAC](#) – тип даних [VARCHAR](#) (255) – зберігає MAC-адресу пристрою. За замовчуванням [NULL](#).
- [Root\\_pass](#) – тип даних [VARCHAR](#) (255) – зберігає пароль від облікового запису з правами адміністратора. За замовчуванням [NULL](#).
- [OS](#) – тип даних [VARCHAR](#) (100) – містить інформацію про операційну систему або прошивку пристрою та її версію. За замовчуванням [NULL](#).
- [Tech\\_spec](#) – тип даних [VARCHAR](#) (400) – містить технічні характеристики пристрою. За замовчуванням [NULL](#).
- [Part numb](#) – тип даних [VARCHAR](#) (400) – містить інвентарну та облікову інформацію про пристрій, наприклад – інвентарний та серійні номери, виробник та модель пристрою, іншу інформацію за необхідністю. За замовчуванням [NULL](#).
- [Notes](#) – тип даних [VARCHAR](#) (400) – за необхідністю містить додаткові примітки про обладнання, користувача, специфічні налаштування або інші відомості. За замовчуванням [NULL](#).
- [Is\\_checked](#) – тип даних [TINYINT](#) – службове поле, для подальшого використання при створенні інтерфейсу користувача. Дозволяє відміти запис під час виводу у інтерфейс програми. За замовчуванням [NULL](#).

Таблиця [employees](#) Як зазначено вище за текстом у пункті 2.2.2 цього документу таблиця [employees](#) містить інформацію про співробітників організації. Для збереження цієї інформації таблиця містить наступні поля:

- [id](#) – тип даних [INT](#) (255) – дане поле є первинним ключем. Зберігає ідентифікатор запису у таблиці.
- [Full\\_name](#) – тип даних [VARCHAR](#) (255) – ПІБ співробітника. За замовчуванням [NULL](#).
- [Phone numb](#) – тип даних [VARCHAR](#) (100) – містить

номер телефону співробітника. За замовчуванням [NULL](#). [Email](#) – тип даних [VARCHAR](#) (255) – містить адресу особистої електронної пошти співробітника. За замовчуванням [NULL](#). [Notes](#) – тип даних [VARCHAR](#) (255) – містить додаткові відомості про співробітника, наприклад, місцезнаходження робочого місця співробітника у організації або відділ у котрому працює. За замовчуванням [NULL](#). [Is\\_checked](#) – тип даних [TINYINT](#) – службове поле, для подальшого використання при створенні інтерфейсу користувача. Дозволяє відміти запис під час виводу у інтерфейс програми. За замовчуванням [NULL](#). Таблиця [corp\\_email](#) Дана таблиця містить інформацію про корпоративні електронні скриньки співробітників організації. У більшості випадків адміністрування корпоративної електронної пошти полягає на системного адміністратора, тому ведення обліку електронних скриньок є важливою частиною роботи системного адміністратора. Таблиця містить наступні поля: [Id](#) – тип даних [INT](#) (255) – дане поле є первинним ключем. Зберігає ідентифікатор запису у таблиці. [Full\\_name](#) – тип даних [VARCHAR](#) (255) – ПІБ співробітника. За замовчуванням [NULL](#). [Email](#) – тип даних [VARCHAR](#) (255) – містить адресу корпоративної електронної пошти співробітника. За замовчуванням [NULL](#). [Password](#) – тип даних [VARCHAR](#) (255) – містить пароль від копоративної електронної скриньки користувача. За замовчуванням [NULL](#). [Res\\_codes](#) – тип даних [VARCHAR](#) (255) – містить резервні коди доступу, необхідні для відновлення доступу до електронної пошти, у разі якщо користувач забув або втратив пароль. За замовчуванням [NULL](#). [Notes](#) – тип даних [VARCHAR](#) (255) – містить додаткові відомості про корпоративну електронну пошту співробітника або інші відомості. За замовчуванням [NULL](#). [Is\\_checked](#) – тип даних [TINYINT](#) – службове поле, для подальшого використання при створенні інтерфейсу користувача. Дозволяє відміти запис під час виводу у інтерфейс програми. За замовчуванням [NULL](#). Таблиця [servers](#) Таблиця [servers](#) необхідна для зберігання більш розгорнутої інформації, у порівнянні з таблицею [ip\\_table](#), щодо серверів організації, котрі обслуговуються системним адміністратором. Дана таблиця містить наступні поля: [Id](#) – типу даних [INT](#) (255) – дане поле є первинним ключем. Зберігає ідентифікатор запису у таблиці. [Name](#) – типу даних [VARCHAR](#) (255) – дане поле зберігає коротке ім'я серверу. За замовчуванням [NULL](#). [Domain\\_name](#) – типу даних [VARCHAR](#) (255) – зберігає повне доменне ім'я серверу. За замовчуванням [NULL](#). [Ip](#) – типу даних [VARCHAR](#) (255) – зберігає IP-адресу або декілька адреса серверу. За замовчуванням [NULL](#). [Mac](#) – типу даних [VARCHAR](#) (100) – зберігає MAC-адресу серверу. За замовчуванням [NULL](#). [OS](#) – типу даних [VARCHAR](#) (100) – зберігає інформацію про встановлену на сервер операційну систему та її версію. За замовчуванням [NULL](#). [Is\\_virtual](#) – типу даних [VARCHAR](#) (20) – позначає чи є сервер віртуальним. За замовчуванням [NULL](#). [Usage](#) – типу даних [VARCHAR](#) (400) – зберігає інформацію про призначення та використання серверу. За замовчуванням [NULL](#). [Location](#) – типу даних [VARCHAR](#) (400) – зберігає інформацію про віртуальне або фізичне розміщення серверу. За замовчуванням [NULL](#). [Tech\\_spec](#) – типу даних [VARCHAR](#) (400) – зберігає інформацію про фізичні або віртуальні технічні характеристики серверу. За замовчуванням [NULL](#). [Notes](#) – типу даних [VARCHAR](#) (400) – слугує для збереження інших приміток, за необхідності. За замовчуванням [NULL](#). Таблиця [ip\\_phones](#) Дана таблиця зберігає у собі інформацію про усі IP-телефони, що використовуються у організації, а також інформацію про їх налаштування та інші необхідні для роботи системного адміністратора відомості. [Id](#) – типу даних [INT](#) (255) – дане поле є первинним ключем. Зберігає ідентифікатор запису у таблиці. [Phone\\_num](#) – тип даних [VARCHAR](#) (100) – зберігає номер SIP-телефону, даного пристрою. За замовчуванням [NULL](#). [Full\\_name](#) – тип даних [VARCHAR](#) (255) – зберігає ПІБ співробітника. За замовчуванням [NULL](#). [Username\\_pass](#) – тип даних [VARCHAR](#) (255) – містить логін та пароль облікового запису з правами адміністратора від даного пристрою. За замовчуванням [NULL](#). [Sip\\_pass](#) – тип даних [VARCHAR](#) (255) – зберігає SIP-пароль від білового запису, котрий необхідний для авторизації акаунта на сервері IP-телефонії. За замовчуванням [NULL](#). [Notes](#) – типу даних [VARCHAR](#) (400) – зберігає інші примітки. За замовчуванням [NULL](#). [Is\\_checked](#) – тип даних [TINYINT](#) – службове поле, для подальшого використання при створенні інтерфейсу користувача. Дозволяє відміти запис під час виводу у інтерфейс програми. За замовчуванням [NULL](#). Таблиця [manuals](#) Таблиця [manuals](#), як вже було зазначено вище за текстом, зберігає у собі мануали, інструкції налаштувань та інші важливі записи, що використовуються системним адміністратором під час налаштування обладнання або обслуговування мережі організації. [Id](#) – типу даних [INT](#) (255) – дане поле є первинним ключем. Зберігає ідентифікатор запису у таблиці. [Title](#) – тип даних [VARCHAR](#) (255) – короткий заголовок запису, слугує для полегшення навігації. [Category](#) – тип даних [VARCHAR](#) (100) – містить у собі декілька фіксованих категорій записів, що при виводі до додатку, дозволяє розділити їх на групи, що спрощує навігацію при великій кількості наявних в таблиці записів. За замовчуванням [NULL](#). [Manual](#) – тип даних [VARCHAR](#) (5000) – містить текст мануалу або іншого запису. За замовчуванням [NULL](#). 2.2.4 Нормалізація бази даних Після етапу первинного моделювання будь-яка база даних повинна пройти процес приведення її до нормальних форм (НФ) – процес нормалізації. Процес нормалізації — це метод упорядкування структури бази даних для забезпечення її ефективності, мінімізації надмірності даних та уникнення аномалій при їх оновленні, видаленні або додаванні. Метою нормалізації є розділення даних на логічно пов'язані таблиці та визначення чітких зв'язків між ними. Для додатків невеликого формату з базою даних, що містять у собі невелику кількість таблиць, є достатнім приведення бази даних до третьої нормальної форми (3НФ), це забезпечить її нормальне ефективне та функціонування. Розглянемо детальніше кожну з нормальних форм баз даних: 1. Перша нормальна форма (1НФ): Усі поля містять атомарні (неподільні) значення. У кожному полі таблиці лише одне значення. Немає повторюваних груп даних. 2. Друга нормальна форма (2НФ): Відповідність 1НФ. Усунення часткових залежностей: кожен неключовий атрибут повинен залежати від первинного ключа, а не від його частини. 3. Третя нормальна форма (3НФ): Відповідність 2НФ. Усунення транзитивних залежностей: неключові атрибути не повинні залежати від інших неключових атрибутів. Аналізуючи наведену вище базу даних, для її приведення до нормальних форм необхідно виконати наступні дії: 1. Поле [full\\_name](#), що зберігає ім'я співробітника, використовується одразу у декількох таблицях бази даних, а саме у таблицях: [employees](#), [ip\\_table](#), [corp\\_email](#), [users](#), [ip\\_phones](#). У такому випадку доцільним рішенням, буде винести зберігання даної інформації до таблиці [employees](#), що зберігає інформацію про співробітників організації. У інших таблицях, дане поле необхідно замінити на поле [employee\\_id](#), що буде зберігати [id](#) запису у таблиці [employees](#), завдяки чому буде отримуватися доступ до необхідної інформації при запитах зі сторони додатку. Такий підхід забезпечить виключення повторів збереження однакової інформації у базі даних та оптимізує її роботу. 2. У таблицях [ip\\_table](#) та [servers](#) міститься поле [os](#), що зберігає інформацію про операційну систему, під керівництвом якої працює даний пристрій. Оскільки перелік операційних систем не є інформацією унікальною, а їх кількість є скінченною та типовою, необхідно винести дане поле в окрему таблицю бази даних, що буде зберігати записи про операційні системи. Відповідно у таблицях [ip\\_table](#) та [servers](#), поле [os](#), необхідно замінити на поле [os\\_id](#), що буде містити ідентифікатор запису з таблиці [os](#) бази даних, що дасть змогу отримувати необхідну інформацію про операційну систему при запитах до бази даних зі сторони додатку. Виконавши дані дії, ми отримаємо базу даних, що відповідає наступним нормальним формам: 1. Першій нормальній формі (1НФ) оскільки: Усі поля таблиць містять тільки одне атомарне значення; Відсутні повторювані

або багатозначні групи даних; 2. Другій нормальній формі (2NF) оскільки: Забезпечена відповідність першій нормальній формі (1NF); Відсутні часткові залежності таблицях є первинні ключі та неключові атрибути і вони залежать лише від первинних ключів. 3. Третій нормальній формі (3NF) оскільки: Забезпечена відповідність другій нормальній формі (2NF). Транзитивні залежності усунені за рахунок розділу даних на окремі таблиці, наприклад: таблиці `ip_table`, `users`, `ip_phones`, `corp_email` посилаються на таблицю `employees` через поле `employee_id`. У результаті проведеного аналізу бази даних та процесу її приведення до нормальних форм, а саме третьої нормальної форми, було отримано базу даних, що має наступну структуру та відповідає наступній інфологічній структурі: 1. Загальна структура бази даних після нормалізації: Рис. 2.8. Загальна структура бази даних після нормалізації 2. Інфологічна модель бази даних після нормалізації: Рис. 2.9 Інфологічна модель бази даних після нормалізації 2.2.5. Підключення бази даних до проекту Після створення бази даних додатку, наступним кроком йде її підключення до проекту. Для забезпечення цієї мети буде використовуватися

Цитування: **0,02%** id: **16**  
«принцип єдиного підключення»

з англ.

Цитування: **0,02%** id: **17**  
«Single Connection Pattern».

Основою цього паттерну, є принцип, коли підключення до бази не розноситься за кодом додатку, а виконується в одному місці. Для цього необхідно створити файл у якому буде відбуватися підключення до бази даних, назовемо його `db_connect.php` та напишемо в ньому наступний код: Рис. 2.10. Підключення БД до проекту Після підключення БД до проекту, для організації взаємодії додатку та бази даних, необхідно підключити даний файл до іншого файлу з кодом, де необхідна взаємодія з БД. PHP дозволяє це зробити використовуючи функцію `require_once`. Такий підхід має практичні переваги та забезпечує: Зручність підтримки: Зміни легко реалізувати в одному файлі без впливу на інші частини коду. Зменшення дублікатів: Немає необхідності дублювати код підключення в кожному скрипті. Підвищення безпеки: Файл з підключенням можна розташувати поза кореневою директорією веб-сервера, захищаючи його від несанкціонованого доступу. 2.2.6. Організація взаємодії бази даних з додатком Після підключення бази даних до додатку, наступним кроком є організація взаємодії його елементів з останньою. У даному додатку для кожної таблиці бази даних буде створена окрема сторінка, на котру буде виводитися інформація з таблиць для подальшої взаємодії з користувачем. Для взаємодії даному додатку буде використовуватися принцип операційно-орієнтованого підходу (*Operation-Oriented Approach*)[31,32]. Основною особливістю даного принципу є те, що для кожної операції *CRUD* (*create, read, update, delete*) реалізується окремий скрипт. Даний підхід можна використовувати лише у невеликих або середніх за розмірами додатках, оскільки по мірі зростання продукту та збільшення його функціоналу пропорційно збільшується й кількість скриптів, що буде містити у собі кінцевий продукт, що у свою чергу ускладнить його підтримку та розвиток. Проте, навідміну від великих додатків, у малих або середніх даний підхід гарно себе рекомендує, оскільки: Він є простим у реалізації, що пришвидшує та здешевшує розробку додатку, не впливаючи на його функціонал. Підвищується надійність програми, оскільки виникнення помилок у роботі одного скрипта, не вплине на роботу інших скриптів. Спрощується підтримка програми, оскільки кожен скрипт має просту та логічну структуру, що може бути корисним, при роботі початківців або людей не знайомих з програмою. Виходячи з цього принципу, у додатку буде реалізовано окремий скрипт під кожну операцію(створення, читання, оновлення, видалення) для кожної окремої сторінки додатку. Реалізація операції `read` Операція читання у даному додатку реалізовує вивід інформації з бази даних на екран користувача. Можливості мови PHP дозволяють гнучко комбінувати виведені дані з баз даних разом з `HTML` та `CSS`, що у результаті дає змогу створити зручний, лаконічний та зрозумілий інтерфейс кінцевого продукту. Оскільки операція читання у даному додатку не обмежується лише простим виводом неформатованої інформації на екран, а потребує форматування під кожну сторінку окремо, з урахуванням потреб до зручності та особливостей даних, що зберігаються у таблицях, дана операція буде реалізована нижче за текстом у розділах, присвячених розробці інтерфейсу користувача. Реалізація операції `create` Нижче наведемо приклад типового скрипту для операції `create`, сторінки `users.php`: Рис. 2.11. Приклад коду операції `create` для сторінки `users.php` З рис. 2.11 видно, що до скрипту підключається файл підключення бази даних, що є реалізацією принципу єдиного підключення. Також видно метод `session_start()`, що відкриває сесію, для можливості роботи з суперглобальними масивами `$_POST` та `$_GET`, котрі дозволяють організувати передачу даних між сторінками додатку. Після отримання даних, вони записуються до змінних, після чого виконується запит до бази даних на додавання нового запису про користувача. Також на даному скриншоті видно, що змінна `$password`, перед записом до бази даних хешується з використанням методу `md5()`, що є вимогою до безпеки додатку та напряду впливає на її підвищення. Реалізація операції `update` Наведемо приклад типового скрипта для операції `update` сторінки `users.php` Рис. 2.12. Приклад коду реалізація операції `update` на сторінці `users.php` Як бачимо з рис. 2.12 код є доволі схожим з операцією `create`, що є підтвердженням принципу

Цитування: **0,01%** id: **18**  
«операційно-орієнтованого підходу».

Проте у даному скрипті навідміну від запиту `INSERT INTO` виконується запит `UPDATE WHERE ID`. Як і було зазначено вище у пункті 2.2.2, саме поле `id` є первинним ключем, тому пошук записів у базі даних здійснюється саме по ньому. Реалізація операції `delete` Наведемо приклад реалізації скрипта, що виконуватиме операцію `delete` на сторінці `users.php`: Рис. 2.13. Приклад коду реалізації операції видалення на сторінці `users.php` Як бачимо з рис. 2.13 операція видалення також працює за первинним ключем, котрим виступає поле `id`, виконуючи запит `DELETE FROM WHERE ID`. Скрипти аналогічних операцій на інших сторінках будуть розроблені за аналогією до даного прикладу. Код даних операцій буде наведено у додатках до даного документа нижче за текстом. Варто зазначити, що отримання даних, з котрими будуть взаємодіяти скрипти даного додатку, буде відбуватися за допомогою `JavaScript`, котрий буде зчитувати вже виведені на екран користувача дані й передавати їх для обробки скриптам. Такий підхід є реалізацією технології

Цитування: **0,03%** id: **19**  
"Обробка даних на стороні клієнта"

(*Client-Side Handling*). Такий підхід дозволяє значно зменшити навантаження на базу даних, оскільки кратно зменшує кількість запитів до неї, що буде позитивно впливати на стабільність роботи додатку при великих навантаженнях. Оскільки використання `JavaScript` напряду залежить від `HTML`-змісту сторінки, реалізація даного підходу буде описана нижче за текстом у розділах присвячених розробці веб-інтерфейсу додатку. 2.3. Розробка веб-інтерфейсу користувача Після розробки бази даних та її підключення до додатку, постає

час перейти до розробки фронт-енд частини або ж простіше кажучи – веб-інтерфейсу користувача. Даний етап є не менш важливим з точки зору розробки, оскільки саме з веб-інтерфейсом користувач взаємодіє найбільше. Саме від правильно спроектованого, надійного та зрозумілого інтерфейсу залежить те, наскільки користувачу буде комфортно працювати за програмою.

### 2.3.1. Загальні принципи розробки веб-інтерфейсу додатку

Враховуючи досвід роботи з веб-додатками у роботі та повсякденному житті, а також враховуючи досвід роботи з додатками схожого функціоналу – було прийнято рішення використовувати при розробці веб-інтерфейсу так званий принцип

Цитування: **0,03%**

id: **20**

«блокової структури» ([Block layout](#))[33]

, основою даного принципу є розділ сторінки на основні блоки, з котрих в подальшому складається уся сторінка. На блоки можна ділити за змістом, розташуванням тощо. У даному додатку, буде використано класичний поділ сторінки на: [Header](#) – верхня панель сторінки. Зазвичай містить функціональні кнопки, що найбільш часто використовуються користувачами. [Main](#) – основна частина сторінки на яку виводиться основний контент. [Sidebar](#) – бокова панель. Може містити кнопки навігації або інші функціональні елементи. [Footer](#) – так званий підвал, або нижня частина сторінки. Зазвичай містить менш важливий контент, а також інформацію про розробників тощо. Нижче наведено загальний макет веб-інтерфейсу програми: Рис. 2.14. Загальний макет веб-інтерфейсу додатку Також мова PHP дозволяє використовувати принцип

Цитування: **0,01%**

id: **21**

«модульної розробки»

([Modular Development](#)). Під час використання даного підходу розробки, кожен з таких елементів ([header](#), [footer](#), [sidebar](#)), що залишаються незмінними при переході між сторінками додатку, можна винести у окремий файл. Цей файл за необхідності можна підключати до будь-якої сторінки, у результаті отримавши його на сторінці. Даний підхід має наступні переваги: Зменшує кількість повторів коду - великі елементи можна підключати до сторінки однією строчкою коду. Спрощує редагування коду – якщо необхідно змінити той чи інший елемент, варто відредагувати лише один файл, замість необхідності редагувати кожен сторінку окремо. Загальний принцип побудови сторінок додатку На початку розробки необхідно чітко сформулювати, який зовнішній вигляд матимуть сторінки у кінцевому їх варіанті. Оскільки більшість інформації, що зберігається у базі даних можна представити у вигляді таблиці, саме таблиця і буде основним елементом більшості сторінок, за виключенням тих, де формат даних у базі даних потребує іншого варіанту відображення у інтерфейсі користувача. Як було сказано вище за тестом, у розділі присвяченому базі даних додатку, кожній таблиці БД буде відповідати власна сторінка, на яку будуть виводитися дані для користувача. Кожна сторінка буде містити власний набір функціональних кнопок, що будуть давати змогу користувачу працювати з даними, отриманими з БД. Також кожна сторінка повинна містити блок навігації для можливості переходити між сторінками додатку. Для максимального збільшення робочого простору, а також кількості одночасно видимої інформації на екрані, більшість функціональних елементів будуть зроблені модальними (будуть з'являтися на екрані лише при натисканні на відповідні кнопки). Це забезпечить максимально ефективне використання робочого простору на екрані.

### 2.3.2. Детальний огляд розробки елементів веб-інтерфейсу та веб-сторінок додатку

У даному підпункті документу буде наведено приклади розробки основних елементів веб-інтерфейсу додатку а також наведено приклади кінцевого зовнішнього вигляду готових елементів. Варто зауважити, що у даному розділі не буде наведено приклади розробки всіх сторінок додатку, оскільки це не є доцільно, через велику кількість інформації, що буде повторюватися. У розробці багатьох сторінок використовуються схожі або ідентичні принципи, тому вони не потребують окремого пояснення у даному розділі. Сторінка авторизації користувачів Першою сторінкою, що побачить перед собою користувач, коли вперше відкриє додаток – є сторінка авторизації, вона є свого роду стартовою точкою додатку, тому логічно буде розпочати розробку саме з нього. Дана сторінка буде містити форму авторизації, що міститиме у собі два поля для вводу: логін та пароль, а також кнопку підтвердження. Також необхідно реалізувати вивід користувачу повідомлення при неправильності вводу логіна чи пароля, при помилковому вводі. Логіка роботи даної сторінки наступна: скрипт зчитує логін та пароль, введений користувачем у поля форми, після чого шукає у базі даних у таблиці [users](#) запис з відповідними даними, якщо запис знайдено – йде переадресація на головну сторінку додатку, якщо ні – виводиться повідомлення про неввірно введений логін або пароль. Приклад форми авторизації разом з вікном помилки входу наведено нижче: Рис. 2.15. Форма авторизації та повідомлення про помилку вводу даних Розробка головної сторінки додатку, а також елементів з котрих вона складається ([header](#), [footer](#), [sidebar](#), [main](#), [go-top button](#)) [Header](#) Після успішної авторизації користувач потрапляє до головної сторінки додатку, дана сторінка буде містити у собі дані з бази даних, що зберігаються у таблиці [ip\\_table](#), так як саме вона зберігає у собі основну необхідну інформацію для роботи системного адміністратора, а саме – інформацію про локальні мережі/мережу, що обслуговуються адміністратором. Дана сторінка є типовою для даного додатку, а отже її розробку можна привести у приклад, оскільки більшість інших сторінок будуть розроблені за схожими або ідентичними принципами. Почнемо розробку з верхнього меню сторінки ([header](#)). Як було сказано вище за текстом, для забезпечення максимального робочого простору, [header](#) буде містити елементи керування та інше необхідне функціональне навантаження. В залежності від даних да формату сторінки, набір елементів верхнього меню може відрізнятися від сторінки до сторінки, нижче наведемо приклад верхнього меню для сторінки [main.php](#): Рис. 2.16. Приклад верхнього меню для сторінки [main.php](#) На скриншоті можна побачити наступні елементи(у порядку зліва направо): Кнопка відкриття бокового меню навігації між сторінками. Два випадаючих списки, що містять інформацію про усі

Цитування: **0,01%**

id: **22**

«білі»

айпі адреси організації та перелік адрес локальних мереж організації. Дана інформація є важливою та часто необхідною для адміністратора, тому вона на головній сторінці додатку винесена на верхню панель. Кнопка додавання запису до таблиці, інформація з котрої виведена на даній сторінці. ПІБ користувача, від імені котрого на даний момент здійснено вхід у додаток. Кнопка виходу з додатку ([LogOut](#)). Усі елементи, описані вище, окрім двох випадаючих списків, є типовими для кожної сторінки додатку, це означає, що при переході між сторінками додатку, вони завжди будуть доступні на верхній панелі додатку. [Sidebar](#) Після написання верхнього меню додатку, перейдемо до написання бокового меню навігації. Дане меню є досить простим за своєю структурою, але від цього не втрачає свою важливість. Воно необхідне для створення можливості користувачу, переходити між сторінками додатку, що є однією з ключових функцій, що необхідні користувачу у роботі. Як було сказано у попередніх розділах, для збільшення корисного робочого простору, багато елементів будуть створені модальними(з'являтимуться лише при натисканні на



<p>відповідні кнопки у інтерфейсі), бокове меню навігації, є одним з таких елементів. Забезпечити такий функціонал можна завдяки <a href="#">JavaScript</a>. Напишемо наступний код, що дозволить показувати бокове меню, лише при натисканні на кнопку на верхній панелі: Рис. 2.17. Код відкриття модального бокового меню</p> <p>Тепер необхідно створити саме бокове меню, котре буде містити посилання на усі сторінки, що в подальшому будуть розроблені. Наведемо приклад готового варіанту бокового меню програми: Рис. 2.18. Готовий варіант бокового меню навігації</p> <p><b>Footer</b> Ще одним елементом необхідним на кожній сторінці є так званий підвал (<b>Footer</b>), зазвичай даний елемент містить контактну інформацію на сайт компанії, або інші корисні посилання, а також інформацію про розробника. У даному варіанті, оскільки проект не є комерційним, розмістимо лише інформацію про розробника. Наведемо приклад кінцевого варіанту футеру: Рис. 2.19. Приклад футеру додатка</p> <p>Кнопка переходу до гори сторінки</p> <p>Одразу перед переходом до створення та наповнення контентом основної частини сторінки, необхідно створити функціональну кнопку, що дозволить переходити до гори сторінки. Створення та інтеграція такої кнопки, зробить значно зручнішим використання даного додатку при великих обсягах інформації, адже кількість скролів екрану може бути зовсім великою для комфортної роботи. Розробимо <a href="#">HTML</a>-код даної кнопки та наведемо нижче її приклад: Рис. 2.20. Зовнішній вигляд кнопки переходу у гору сторінки</p> <p>Після створення самої кнопки, необхідно написати наступний <a href="#">JavaScript</a> скрипт, що реалізує функціонал даної кнопки: Рис. 2.21. Скрипт, що реалізує функціонал кнопки переходу до гори сторінки</p> <p>Після написання даного скрипта, кнопка переходу до гори сторінки буде з'являтися лише при прокрутці екрану не менше ніж на 150 пікселів у низ, а після натискання на неї, екран користувача буде переміщено на самий верх сторінки. <a href="#">Main</a> (основний контент сторінки)</p> <p>Після створення допоміжних елементів сторінки, необхідно переходити до наповнення сторінки основною інформацією з котрою буде працювати та взаємодіяти користувач. Як вже було описано вище за текстом, інформація на сторінки буде виводитися з бази даних, реалізуючи при цьому одну з основних функцій даного додатку, а саме <a href="#">READ</a> (читання інформації з бази даних). Інформація на більшості сторінок буде знаходитися у вигляді таблиць(окрім сторінок де формат інформації з БД не дозволяє представити його у вигляді таблиці), з метою спрощення сприйняття інформації користувачем та полегшенню роботи з нею. Таблиця за структурою у більшості випадків буде співпадати з таблицею бази даних, а у останній стовпчик будуть додані функціональні кнопки, що реалізовуватимуть ключові операції цього додатку по роботі з базою даних, а саме редагування та видалення. Завдяки можливості мови програмування PHP вбудовувати свій код безпосередньо у <a href="#">HTML</a> код, створимо таблицю наступного вигляду, та заповнимо її певною інформацією з бази даних: Рис. 2.22. Приклад типової таблиці зі сторінки <a href="#">main.php</a></p> <p>Як ми можемо побачити на рис. 2.22, утворена таблиця у більшості полів відповідає за структурою таблиці з бази даних, а у останньому стовпчику розміщені функціональні кнопки, що дозволяють взаємодіяти з даними таблиці. Після створення усіх основних елементів сторінки необхідно поєднати їх в одне ціле, щоб отримати готовий варіант сторінки. Використовуючи описаний вище за текстом принцип</p>	
<div><div>»</div><div>Цитування: 0,01%</div></div>	id: 23
«модульної розробки»	
<p>(<a href="#">Modular Development</a>), а також використовуючи можливості мови PHP підключимо усі допоміжні елементи до головної сторінки: Рис.2.23. Підключення елементів до сторінки <a href="#">main.php</a></p> <p>У результаті після підключення усіх елементів до сторінки <a href="#">main.php</a>, отримаємо її наступний вигляд: Рис. 2.24. Сторінка <a href="#">main.php</a> з усіма підключеними до неї елементами</p> <p>Оскільки сторінка <a href="#">main.php</a> є типовим прикладом, за аналогією до неї будуть розроблені й інші сторінки даного додатку, що міститимуть у своїй основній структурі таблицю, наведемо приклад декількох(<a href="#">users.php</a>, <a href="#">employees.php</a>, <a href="#">ip-phones.php</a>) створених за аналогією сторінок додатку: Рис. 2.25. Сторінка <a href="#">users.php</a> з усіма підключеними до неї елементами</p> <p>Рис. 2.26. Сторінка <a href="#">employees.php</a> з усіма підключеними до неї елементами</p> <p>Рис. 2.27. Сторінка <a href="#">ip-phones.php</a> з усіма підключеними до неї елементами</p> <p>Створення</p>	
<div><div>»</div><div>Цитування: 0,01%</div></div>	id: 24
«нетипової»	
<p>сторінки додатку</p> <p>Як було сказано вище за текстом, не всі сторінки додатку можна призвести до</p>	
<div><div>»</div><div>Цитування: 0,01%</div></div>	id: 25
«типового вигляду»,	
<p>через формат та структуру інформації, що зберігається у базі даних. Однією з таких сторінок є <a href="#">manuals.php</a>. Як видно з назви сторінки вона буде виводити на екран змісти таблиці у базі даних з назвою <a href="#">manuals</a>, котра зберігає мануали та інструкції, що використовуються системним адміністратором у роботі. Дана сторінка так само як і інші буде містити верхнє меню, бокове меню навігації та футер як й інші сторінки. Проте основний контент буде відрізнятися за своїм форматом та виглядом. Інструкції та мануали можуть бути значними за розміром, а отже зберігати їх у вигляді таблиці не є коректним рішенням. У такому разі правильніше буде виводити їх на сторінку певними блоками, кожен з яких буде містити короткий заголовок та сам текст мануалу. Також необхідно у такий блок додати функціональні кнопки, що дозволять користувачу взаємодіяти х інформацією. Нижче наведемо приклад такого блоку з виведенням з бази даних мануалом: Рис. 2.28. Приклад блоку з мануалом на сторінці <a href="#">manuals.php</a></p> <p>Як бачимо з рис. 2.28, створений блок містить усі описані вище елементи, що необхідні для зручності взаємодії користувача з інформацією. Також для зручності навігації при великій кількості записів на дані сторінці необхідно створити блок змісту сторінки, що у свою чергу буде поділений на категорії. Зміст буде автоматично формуватися при виводі інформації з бази даних та містити у собі посилання на конкретні записи на сторінці, посилання будуть відповідати назвам коротких заголовків записів із бази даних. Такий підхід значно спростить навігацію сторінкою, а також пришвидшить пошук необхідної інформації. Наведемо приклад змісту сформованого автоматично при виводі записів на сторінку <a href="#">manuals.php</a> із бази даних: Рис. 2.29. Приклад блоку змісту на сторінці <a href="#">manuals.php</a></p> <p>Тепер до сторінки можна усі необхідні елементи, а саме: <a href="#">header</a>, <a href="#">footer</a>, <a href="#">Sidebar</a>, <a href="#">go-top_button</a>. У результаті отримаємо сторінку наступного виду: Рис. 2.30. Фінальний вигляд сторінки <a href="#">manuals.php</a></p> <p>Інші</p>	
<div><div>»</div><div>Цитування: 0,01%</div></div>	id: 26
«нетипові сторінки»	
<p>будуть розроблятися за схожими принципами, вони формуватимуться з урахуванням їх змісту, а отже структура кожної з них може певним чином відрізнятися. Основні елементи та функціональні кнопки можуть мати інше розташування на сторінці, але вони матимуть незмінний односторонній функціонал. 2.3.3. Реалізація функціональних кнопок на сторінках додатку</p> <p>Після створення структури сторінок та наповнення їх інформацією з бази даних, наступним етапом йде організація можливості взаємодії користувача з інформацією на сторінках. Як бачимо з пункту 2.3.3 цього документу, сторінки додатку у своїй структурі вже містять функціональні кнопки, але вони ще не наділені необхідним функціоналом.</p>	

Виходячи с формату додатку (**CRUD**-додаток), одразу можна зробити висновок який функціонал мають отримати кнопки, що розміщені на сторінках додатку – вони повинні реалізувати основі операції роботи з базою даних, а саме **Create** – створення записів у базі даних, **Update** – оновлення/редагування записів у базі даних, **Delete** – видалення записів із бази даних. Перейдемо до реалізації даного функціоналу. Реалізація додавання записів до БД Для додавання записів на сторінках додатку необхідно розробити модальну форму(прихована доти, доки користувач не натисне на відповідну кнопку) для кожної сторінки. В дану форму користувач буде вносити дані котрі необхідно внести до бази даних. Після введення ці дані будуть передаватися до **php**-скрипта, що буде вносити їх до бази даних. Розглянемо даний принцип на прикладі модальної форми додавання запису до таблиці **ip\_table** на сторінці **main.php**. Для початку розробимо форму внесення даних: Рис. 2.31. Модальна форма додавання запису на сторінці **main.php** Тепер розробимо код **JavaScript**, що буде показувати або приховувати дану форму на сторінці: Рис. 2.32. Код показу або приховування модальної форми Тепер після підключення даного скрипта до сторінки, при натисканні на функціональну кнопку, що відповідає за додавання – модальна форма з'явиться на сторінці і в неї можна буде внести дані. Після натискання підтвердження дані будуть передані скрипту PHP додавання записів до таблиці, що був описаний у пункті 2.2.5 цього документа. Даний скрипт додасть дані до бази даних й вони з'являться на сторінці. Наведемо приклад, як виглядає модальна форма додавання на сторінці **main.php**: Рис. 2.33. Модальна форма додавання на сторінці **main.php** Реалізація редагування записів у БД Наступною операцією, що буде реалізована – буде операція оновлення/редагування записів у базі даних. Як і попередню операцію, дану розглянемо на прикладі сторінки **main.php**. Так само як для операції додавання створимо модальну форму та додамо її до сторінки. Дана модальна форма буде практично ідентична до форми додавання, оскільки вона потребує тих самих даних, що і попередня операція. Але є певна відмінність, котра буде розглянута нижче за текстом після опису основного принципу роботи даної операції. Створивши модальну форму, необхідно реалізувати функціонал, котрий буде підставляти до її полів, значення того запису з таблиці, котру планує редагувати користувач. Такий функціонал суттєво спрощує взаємодію з даними, оскільки виключає потребу знову друкувати увесь зміст таблиці, що може бути суттєво при великому обсязі інформації, а також є елементом захисту даних, оскільки знижує вірогідність помилки при повторному внесенні даних у повному обсязі. З метою реалізації даного функціоналу напишемо **JavaScript** скрипт, котрий буде не лише показувати/приховувати форму на екрані користувача, а ще буде заповнювати форму даними з таблиці. Як вже було сказано вище за текстом у пункті 2.2.5 цього документа, організація подібної взаємодії, коли форма заповнюється даними з екрану без виконання запити до бази даних, є реалізацією технології, що має назву

Цитування: **0,03%**

id: **27**

"Обробка даних на стороні клієнта"

(**Client-Side Handling**). Реалізуємо даний скрипт: Рис. 2.34. Скрипт для модальної форми редагування запису у БД Як бачимо зі скриншоту скрипта, він відстежує натискання функціональної кнопки, та отримує текстовий контент його сусідніх елементів, звертаючись до батьківського за ієрархією елемента. Після отримання ці дані вносяться до відповідних полів модальної форми, що у кінцевому підсумку дає наступний результат: Рис. 2.35. Модальна форма додавання запису на сторінці **main.php** з підставленими даними завдяки роботі скрипта Далі за аналогією до форми додавання запису до бази даних, інформація з форми редагування буде передаватися до PHP-скрипта, що був описаний у пункті 2.2.5 цього документа, котрий у свою чергу вже виконає операцію оновлення запису у базі даних. Аналізуючи зміст рис. 2.34 та 2.35 можна побачити й ту відмінність між формою додавання записів та формою редагування. Цією особливістю є чекбокс

Цитування: **0,01%**

id: **28**

«відмітити».

Він реалізує функціонал, що був закладений ще на етапі проектування бази даних додатку. Повертаючись до пункту 2.2.3 цього документа, можна побачити, що майже кожна таблиця бази даних цього додатка, має службове поле **is\_checked**, дане поле слугує для можливості

Цитування: **0,01%**

id: **29**

«віділити»

той чи інший запис кольором при його виводі на сторінку, для полегшення сприйняття інформації або для інших потреб. Реалізується це завдяки додаванню до **html**-властивостей, класу ім'я якого задається в залежності від вміста поля **is\_checked**. Якщо елемент у базі даних містить у даному полі значення 1 – класу призначається ім'я **checked**, якщо ж у даному полі зберігається 0 – клас отримує ім'я **not\_checked**. Далі при накладенні на **html css**-стилів, елементи що мають клас **checked**, фарбуються відповідним кольором, у даному випадку – жовтим. Наведемо скриншот приклад: Рис. 2.36. Приклад кольорового позначення записів у таблицях Як бачимо на рис. 2.36 три записи у таблиці мають жовтий колір, це означає, що при редагуванні запису вони були відмічені чекбоксом. Тепер, щоб зняти виділення необхідно відкрити модальну форму редагування даних записів та зняти чекбокс й кольорове виділення цих елементів буде зняте. Реалізація видалення записів з БД Останньою операцією, котру необхідно реалізувати - є операція видалення записів з бази даних. Ключовим моментом у цьому є реалізація попередження користувача про видалення запису. Такий підхід реалізує підвищений захист даних, оскільки виключає можливість випадкового видалення інформації з бази даних без підтвердження користувачем. Для реалізації даного підходу створимо модальну форму, що буде з'являтися на екрані при натисканні на кнопку видалення запису та буде попереджати користувача про намір видалити певний запис з бази даних. У якості інформативної складової, вона буде містити поле, що є унікальним у рамках даної таблиці. Наведемо приклад даної форми та **JavaScript** скрипта, що буде реалізовувати даний функціонал на сторінці **main.php**: Рис. 2.37. Модальна форма попередження про видалення запису з бази даних Рис. 2.38. Скрипт, що реалізує попередження про видалення запису з бази даних Далі за аналогією до форм додавання та редагування інформація з даної форми передається до PHP-скрипта видалення запису з бази даних, що був описаний у пункті 2.2.5 цього документа, котрий у свою чергу вже виконає операцію видалення запису у базі даних. Вище було наведено приклади реалізації функціоналу функціональних кнопок на сторінці **main.php**. Дані кнопки дають змогу користувачу повноцінно взаємодіяти з інформацією, що зберігається у БД, а саме – додавати записи, редагувати та видалити їх. Реалізація даних кнопок на інших сторінках додатку виконується за аналогічним принципом, лише в урахуванням формату даних на сторінці й не потребують додаткового висвітлення у даному документі. 2.3.4. Реалізація розмежування доступу користувачам за їх ролями Реалізація розмежування доступу до функціоналу додатку за ролями користувачів є важливим інструментом, що підвищує безпеку додатку в цілому. У даному додатку, аналізуючи структуру таблиці **users** бази даних, можна поділити користувачів на дві групи – користувачі з повними правами доступу – адміністратори, та користувачі з

обмеженими правами доступу – користувачі. Відповідно до цього розподілення необхідно обмежити функціонал додатку, для користувачів з обмеженим доступом. На даному етапі розробки, користувачів можна обмежити лише у доступі до певних сторінок додатку, а також обмежити можливості записувати, редагувати чи видаляти записи у базі даних. Реалізуємо даний функціонал. По-перше необхідно вирішити, які елементи будуть доступні користувачам з обмеженим доступом, а які ні. У даному випадку користувачам з обмеженим доступом буде обмежено можливість взаємодіяти з функціональними кнопками, а також буде обмежено можливість переходити на сторінки: [servers.php](#), [users.php](#), [manuals.php](#). Даний функціонал буде реалізовано наступним чином – до усіх елементів, що необхідно сховати від користувачів з обмеженим доступом, буде додано [html](#)-клас

Цитування: **0,01%**

id: **30**

«[PHP - адмін](#)»,

після чого до усіх сторінок буде додано [JavaScript](#) скрипт, що буде перевіряти належність користувача до групи користувачів з обмеженими правами і у випадку такого, шукати елементи з таким класом й ховати їх від даного користувача. Наведемо приклад такого скрипта: Рис. 2.39. Скрипт, що приховує елементи від користувачів з обмеженим доступом

Тепер перевіримо роботу скрипта, зайшовши у програму під користувачем з обмеженими правами доступу. Рис. 2.40. Приклад роботи скрипта, для приховання елементів від користувачів з обмеженим доступом

Як можемо побачити з рис. 2.40, зайшовши під користувачем з обмеженими правами, на екрані відсутні функціональні кнопки, що дозволяються вносити зміни у дані з бази даних, а отже таким користувачам залишається у рамках даної сторінки, доступ лише на читання, що може бути корисно у деяких випадках, при цьому – забезпечуючи безпеку даних. Використовуючи даний підхід, коли у вільному порядку можна відмітити будь-який елемент з метою сховати/показати його, можна організувати досить гнучку та варіаційну систему безпеки даних, що може бути досить зручно так корисно у багатьох випадках. 2.3.5. Захист сторінок додатку від несанкціонованого доступу

Якщо вже говорити про захист додатку – не можна обійти стороною й захист сторінок від несанкціонованого доступу. Це такий доступ, коли зломисники можуть перейти на ту чи іншу сторінку не використовуючи авторизацію, а просто набравши шлях до цієї сторінки у пошуковому полі й перейти на неї в обхід авторизації, тим самим отримавши несанкціонований доступ до додатку. Для забезпечення захисту від такого роду атак зробимо наступне: При успішній авторизації користувачів, необхідно засобами мови програмування PHP відкрити сесію й створити у нову змінну з назвою [user](#), котра буде містити інформацію про даного користувача, що знаходиться у базі даних. Наведемо приклад коду: Рис. 2.41. Відкриття сесії та створення змінної [user](#)

Тепер на кожну сторінку нашого додатку, необхідно підключити скрипт, котрий кожний раз при вході на сторінку, буде перевіряти наявність цієї змінної у сесії. Якщо у сесії не буде цієї змінної, це означатиме, що користувач намагається здійснити несанкціонований доступ до сторінки і його автоматично буде повернуто на сторінку авторизації. Наведемо приклад такого скрипта: Рис. 2.42 Скрипт захисту від несанкціонованого доступу до сторінок додатку

Даний підхід суттєво підвищує безпеку додатку, захищаючи дані системного адміністратора від потенційних зломисників та можливих наслідків їх заволодіння даною інформацією. Висновок до розділу 2

У результаті проведеної роботи, описаної в розділі 2 даного документу, а саме – програмної реалізації [CRUD](#)-додатку, було отримано наступний результат: Було підготовлено та налаштовано робоче середовище розробки додатку, що дало змогу комфортно та ефективно працювати над додатком, використовуючи усі можливості середовища розробки. Було розроблено основний кістяк бази даних додатку, що може бути розширений необхідним чином після продовження розробки на наступних етапах та при додаванні додаткового функціоналу за необхідністю. Також було проведено процес нормалізації бази даних додатку, у результаті якого – базу даних було приведено до відповідності третій нормальній формі (3NF). Було організовано взаємодію між додатком та базою даних, що у свою чергу дозволило реалізувати основні операції взаємодії з базою даних, а саме – читання, створення, редагування, видалення. Було розроблено графічний інтерфейс користувача, що забезпечив можливість повноцінної взаємодії користувача з програмою та базою даних, а також полегшив сприйняття наявності у базі даних інформації та спростив навігацію додатком. Було проведено роботу над посиленням захисту додатку, а саме – забезпечено механізм авторизації користувачів, забезпечено розмежування прав доступу користувачів до певних елементів додатку, було забезпечено захист сторінок від несанкціонованого доступу, а також було підвищено захист даних, реалізацією застережливих повідомлень перед видаленням інформації з бази даних. Підсумовуючи, можна сказати, що РОЗДІЛ 2 є практичною реалізацією інформації отриманої з РОЗДІЛУ 1. РОЗДІЛ 2 є ключовим розділом даної роботи, оскільки він є фінальним етапом реалізації [CRUD](#)-додатку системного адміністратора. ВИСНОВКИ У рамках кваліфікаційної роботи за другим (магістерським) рівнем освіти на тему

Цитування: **0,05%**

id: **31**

«Розробка [CRUD](#)-додатку системного адміністратора на засадах [PHP](#)»

було проведено детальне дослідження предметної області з подальшою розробкою кінцевого додатку. У ході виконання роботи, на основі даних отриманих з дослідження предметної області, було обрано та налаштовано необхідний інструментарій розробки, а саме: [OpenServer Panel](#) 6 у якості локального веб-серверу розробки веб-додатку; Систему контролю версій [Git](#) та веб-сервіс [Github](#), у якості інструментарію, що забезпечує безпеку даних та їх резервне копіювання; [PHP](#)-7.4 у якості основної мови програмування, що відповідає за реалізацію серверної частини додатку; СКБД [MySQL](#)-8.2 у якості системи керування базою даних додатку; [HTML](#), [CSS](#), [JavaScript](#) у якості інструментів реалізації графічного інтерфейсу та клієнтської частини додатку. Після підготовки інструментарію, було проведено програмну реалізацію [CRUD](#)-додатку системного адміністратора, а саме: Було створено локальний домен додатку; Було створено базу даних додатку; Було підключено базу даних до проекту та реалізовано їх взаємодію між собою; Було реалізовано основні операції взаємодії з базою даних, а саме – читання, створення, редагування, видалення. Було реалізовано веб-інтерфейс додатку, що містить усі необхідні користувачу функціональні елементи, для повноцінної та комфортної взаємодії з даними з бази даних. Було реалізовано механізм авторизації користувачів. Було реалізовано механізми обмеження доступу користувачів за ролям; Було реалізовано механізми захисту сторінок від несанкціонованого доступу зломисників. У підсумку, результатом даної кваліфікаційної роботи є реалізований [CRUD](#)-додаток системного адміністратора, що має базовий набір основних функцій, що допоможуть системному адміністратору у робочому процесі, а саме – при документуванні інформації про мережі та пристрої, що ним обслуговуються. Даний додаток зможе суттєво полегшити роботу системного адміністратора, шляхом зменшення необхідної кількості дій, необхідних для повноцінного документування інформації необхідної для роботи. Також наведено детальний опис процесу розробки функціоналу даного [CRUD](#)-додатку. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 1. [Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming, No Starch Press](#), 2019. 544 с. 2. [Ascher D., Lutz M. Learning Python, Second Edition. O'Reilly Media](#),

```

" Цитирования: 0,01% id: 32
"table c
table tr th class=
" Цитирования: 0,01% id: 33
"i a field"
Id /th th IP /th th ПИБ користувача /th th PC User-name: pass /th th MAC /th th Root-pass /th th OS
/th th TIX /th th Інв. інфо /th th Нотатки /th th class=
" Цитирования: 0,01% id: 34
"for admin"
Дії /th /t ?php if(mysql_num_rows($result) 0){ // Вивод даних x из каждой строки while ($row
= $result- fetch_assoc()) { $is_checked =
" Цитирования: 0% id: 35
""
; if ($row
" Цитирования: 0,01% id: 36
"i _checked"
] == 1) { $is_checked =
" Цитирования: 0,01% id: 37
"c checked"

```



<pre>; } else { \$is_checked =</pre>	Цитирования: 0,01%	id: 38
<pre>"not_checked"</pre>		
<pre>; } echo 'tr class=</pre>	Цитирования: 0,02%	id: 39
<pre>"'.\$is_checked'"</pre>		
<pre>td class=</pre>	Цитирования: 0,03%	id: 40
<pre>"ifield"'. \$row["i"]</pre>		
<pre>.' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 41
<pre>"i"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 42
<pre>"full_name"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 43
<pre>"username_pass"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 44
<pre>"mac"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 45
<pre>"root_pass"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 46
<pre>"o"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 47
<pre>"tech_spec"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 48
<pre>"port_numb"</pre>		
<pre>] .' /td td'. \$row[</pre>	Цитирования: 0,01%	id: 49
<pre>"notes"</pre>		
<pre>] .' /td'; echo 'td class=</pre>	Цитирования: 0,01%	id: 50
<pre>"for_admin"</pre>		
<pre>button onclick=</pre>	Цитирования: 0,02%	id: 51
<pre>"viewEditModal(event)"</pre>		
<pre>id=</pre>	Цитирования: 0,01%	id: 52
<pre>"editModalBtn"</pre>		
<pre>class=</pre>	Цитирования: 0,01%	id: 53
<pre>"action_btn"</pre>		
<pre>title=</pre>	Цитирования: 0,01%	id: 54
<pre>"Редагувати"</pre>		
<pre>img src=</pre>	Цитирования: 0,01%	id: 55
<pre>"img/edit-icon.png"</pre>		
<pre>alt=</pre>	Цитирования: 0,01%	id: 56
<pre>"Edit"</pre>		
<pre>class=</pre>	Цитирования: 0,01%	id: 57
<pre>"icon"</pre>		
<pre>/button button onclick=</pre>	Цитирования: 0,02%	id: 58
<pre>"viewDeleteConfirmModal(event)"</pre>		
<pre>id=</pre>	Цитирования: 0,01%	id: 59
<pre>"deleteModalBtn"</pre>		
<pre>class=</pre>	Цитирования: 0,01%	id: 60
<pre>"action_btn"</pre>		
<pre>title=</pre>	Цитирования: 0,01%	id: 61
<pre>"Видалити"</pre>		
<pre>img src=</pre>	Цитирования: 0,01%	id: 62

"/input/delete.php"	
alt=	
Цитирования: 0,01%	id: 63
"Delete"	
class=	
Цитирования: 0,01%	id: 64
"icon"	
/button/td'; } } else{ echo	
Цитирования: 0,04%	id: 65
"Жодного запису в таблиці не знайдено"	
; }; ? /tr/table/div Код скрипта, що реалізує операцію Create (додавання до бази даних) на сторінці main.php?php//создаем сессию для передачи сообщений между страницами приложения session_start(); //подключаем файл подключения к БД require_once	
Цитирования: 0,01%	id: 66
"../connection.php"	
; //записываем переменные полученные из метода POST \$ip = \$_POST['ip']; \$full_name = \$_POST['full_name']; \$username_pass = \$_POST['username_pass']; \$mac = \$_POST['mac']; \$root_pass = \$_POST['root_pass']; \$os = \$_POST['os']; \$tech_spec = \$_POST['tech_spec']; \$part_num = \$_POST['part_num']; \$notes = \$_POST['notes']; //делаем запрос к БД, чтобы добавить запись \$query =	
Цитирования: 0,14%	id: 67
"INSERT INTO ip_table (ip, full_name, username_pass, mac, root_pass, os, tech_spec, part_num, notes) VALUES ('\$ip', '\$full_name', '\$username_pass', '\$mac', '\$root_pass', '\$os', '\$tech_spec', '\$part_num', '\$notes')"	
; mysqli_query(\$connection, \$query); header('Location: ../pages/main.php'); ? Код скрипта, що реалізує операцію Delete (видалення з бази даних) на сторінці main.php?php//создаем сессию для передачи сообщений между страницами приложения session_start(); //подключаем файл подключения к БД require_once	
Цитирования: 0,01%	id: 68
"../connection.php"	
; //записываем переменные полученные из метода POST \$id = \$_POST['id']; //делаем запрос к БД, чтобы добавить запись \$query =	
Цитирования: 0,05%	id: 69
"DELETE FROM ip_table WHERE id = '\$id'"	
; mysqli_query(\$connection, \$query); header('Location: ../pages/main.php'); ? Код скрипта, що реалізує операцію Update (оновлення/редагування записів у БД) на сторінці main.php?php//создаем сессию для передачи сообщений между страницами приложения session_start(); //подключаем файл подключения к БД require_once	
Цитирования: 0,01%	id: 70
"../connection.php"	
; //записываем переменные полученные из метода POST \$id = \$_POST['id']; \$ip = \$_POST['ip']; \$full_name = \$_POST['full_name']; \$username_pass = \$_POST['username_pass']; \$mac = \$_POST['mac']; \$root_pass = \$_POST['root_pass']; \$os = \$_POST['os']; \$tech_spec = \$_POST['tech_spec']; \$part_num = \$_POST['part_num']; \$notes = \$_POST['notes']; \$is_checked = 0; //проверяем выбран ли чекбокс выделения цветом if (isset(\$_POST['is_checked'])) { \$is_checked = \$_POST['is_checked']; } //делаем запрос к БД, чтобы добавить запись \$query =	
Цитирования: 0,24%	id: 71
"UPDATE ip_table SET id = '\$id', full_name = '\$full_name', username_pass = '\$username_pass', mac = '\$mac', root_pass = '\$root_pass', os = '\$os', tech_spec = '\$tech_spec', part_num = '\$part_num', notes = '\$notes', is_checked = '\$is_checked' WHERE id = '\$id'"	
; mysqli_query(\$connection, \$query); header('Location: ../pages/main.php'); ? Фрагмент коду, що реалізує підключення бази даних до проекту ?php//переменная подключения к БД \$connection = mysqli_connect('MySQL-8.2', 'root', '', 'crud'); //проверяем успешность подключения к БД if(!\$connection){ die('Error connect to database!'); } ? Код скрипта, що реалізує захист сторінок від несанкціонованого доступу ?php session_start(); if(!\$_SESSION['user']){ header('Location: /'); } ?	

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу. Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)