Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно – інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка SPA обліку ТЗ підприємства з використанням PHP та РБД»

Виконав(ла): студент(ка) IV курсу, групи СПс-42
спеціальності 121− Інженерія програмного забезпечення
(шифр і назва спеціальності)

		Ільїн В.В.
	(підпис)	(прізвище та ініціали)
Керівник		Цебрій О.Р.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Стоянов Ю.М.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Петрик М.Р.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2024

АНОТАЦІЯ

Бакалаврська робота на тему «Розробка SPA обліку ТЗ підприємства з використанням PHP та РБД» містить 63 сторінок, 10 таблиць, 34 рисунки, 1 додаток. Перелік посилань нараховує 11 найменувань.

Актуальність теми, метою дослідження є необхідність удосконалення та покращення методів обліку транспортних засобів підприємства.

Метою роботи є розробка web сервісу що пришвидшить та підвищить ефективність отримання інформації про транспортні засоби на підприємстві шляхом запису або оновлення актуальної інформації. За основу взято мову гіпертекстової розмітки HTML та скриптову мову програмування PHP.

У результаті роботи було створено web сервіс який надає доступ користувачам до актуальної інформації про транспортні засоби на підприємстві з можливістю редагування та записів, Було проведено дослідження методів обліку ТЗ, обчислення амортизації та бухгалтерських записів.

Набір інструментів для розробки SPA включає: дистрибутив серверної платформи Open Server, що надає можливість створення локального сервера, та включає більшість потрібних інтегрованих інструментів для розробки web сервісів. Для розробки бази даних було доречно використати систему управління базами даних phpMyAdmin, що підтримує мову запитів SQL, тобто, є вебзастосунком для роботи з базами даних, а саме SQL-сервером.

ABSTRACT

The bachelor's thesis on "Development of SPA accounting of enterprise vehicles using PHP and RDBMS" contains 63 pages, 10 table, 34 figures, 1 appendices. The list of references includes 11 titles.

The relevance of the topic, the purpose of the study is the need to improve and improve the methods of accounting for enterprise vehicles.

The aim of the work is to develop a web service that will speed up and increase the efficiency of obtaining information about vehicles at the enterprise by recording or updating relevant information. The hypertext markup language HTML and the scripting language PHP are used as a basis.

As a result of the work, a web service was created that provides users with access to up-to-date information about vehicles at the enterprise with the ability to edit and record, a study of methods of vehicle accounting, depreciation calculation and accounting records was conducted.

The set of tools for the development of SPA includes: a distribution of the Open Server server platform, which provides the ability to create a local server, and includes most of the necessary integrated tools for developing web services. To develop the database, it was appropriate to use the phpMyAdmin database management system, which supports the SQL query language, i.e., it is a web application for working with databases, namely the SQL server.

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ІС - Інформаційна система

ПЗ - Програмне забезпечення

БД - База даних

ТЗ - Транспортний засіб

ЗМІСТ

ВСТУП

Сучасний світ розвивається щосекунди, все більше і більше впроваджуються новітні технології, попит на інформаційні системи зростає, сучасне суспільство не уявляє свого життя без використання новітніх інформаційних технологій, наприклад, розробка вебсайтів, вебзастосунків, програмних застосунків дуже спрощує роботу людини, як на підприємстві, так і у повсякденні.

Мета роботи полягає у розробці системи обліку наявних програмних засобів на підприємстві (легкові та вантажні авто, трактори, сільськогосподарські дрони, комбайни). Щоб виконати поставлену мету треба виконати такі завдання:

1) провести аналіз предметної області;
2) скласти вимоги до розроблюваної ІС;
3) створити зручний односторінковий вебсайт, з доступним інтерфейсом для користувача;
4) розробити базу даних.

Предметом проведення дослідження є бухгалтерські бібліотечні бази даних для обліку транспортних засобів на підприємстві.

Для розробки бази даних було доречно використати систему управління базами даних phpMyAdmin, що підтримує мову запитів SQL, тобто, є вебзастосунком для роботи з базами даних, а саме SQL-сервером, а для інтерфейсу було використано мову гіпертекстової розмітки HTML, скриптові мови програмування JavaScript та PHP, таблицю каскадних стилів CSS.

Було проведено дослідження на перевірку нормативних документів для реєстрації транспортного засобу на підприємстві, проаналізовано процеси у бухгалтерській звітності, процеси амортизації та оподаткування транспортних засобів.

На основі проведеного аналізу вище описаних документів було зпроєктовано базу даних обліку транспортних засобів на підприємстві.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області

Сучасний світ виробництва продукції рослинного походження широко використовує машинні технології. Машинні технології - результат довготривалих наукових досліджень і польових експериментів. Завдяки цим технологіям постійно вдосконалюються і доповнюються нові методи та створюються нові технології, але нові методи можуть бути впровадженні лише в результаті повної перевірки на практиці та отримання позитивного економічного ефекту. Для різних умов господарювання пропонуються різні варіанти технології, проте не всі вони будуть однаково ефективними [1].

Для вибору оптимального варіанту агрономи проводять техніко-економічний аналіз всіх доступних методів з урахуванням нормативів та необхідних ресурсів. Одним з методів амортизації є метод прискореного зменшення залишкової вартості, де річна сума амортизації визначається як добуток залишкової вартості об'єкта на початок звітного року або первісної вартості на дату початку нарахування амортизації та річної норми амортизації, яка обчислюється відповідно до строку корисного використання об'єкта і подвоюється.

Згідно описаного вище формула для розрахунку амортизації виглядає таким чином:

$$A = \frac{BV - RV}{T} * 2$$

де A - сума амортизації;

BV - залишкова вартість об’єкта на початок звітного року або первісна вартість на дату початку нарахування амортизації;

RV - залишкова вартість об’єкта (решта вартості) на кінець звітного року;

T - строк корисного використання об’єкта.

Для знаходження місячної суми амортизації при застосуванні методу прискореного зменшення залишкової вартості, необхідно виконати шлях поділу загальної суми амортизації *А* за рік на 12 місяців.

$$A_{міс} = \frac{A}{12}$$

де А - сума амортизації за рік.

Станом на 2024 рік у відкритому доступі немає web-сервісу для обліку транспортних засобів на підприємстві.

До оптимального функціоналу розроблюваної інформаційної системи відноситься таке як:

- технічний стан транспортного засобу;
- тип транспорту;
- готовність до використання;
- призначення ТЗ;
- потрібна категорія водійського посвідчення для управління ТЗ;
- тип використовуваного палива.

Для кожного транспортного засобу необхідно мати повну інформацію щодо його стану, моделі та типу використовуваного палива, а також його зовнішнього вигляду. Також передбачається можливість додавання, редагування та видалення цієї інформації разом із транспортним засобом, якщо він більше не використовується.

1.2 Аналіз інформаційного забезпечення предметної області

Вхідна документація містить первинну, непереробленну інформацію про стан об’єкта управління і заповнюється вручну. Ця документація охоплює загальні групові дані, отримані в результаті оброблення таких документів, як:

- документи про державну реєстрацію та перереєстрацію;
- документи про облік транспортних засобів;

- звітність про стан транспортних засобів.

Вхідні документи поділяються на оперативні і нормативно-довідкові. Оперативні документи відображають факти фінансово-господарської діяльності підприємства, які змінюються при кожній фіксації. Згідно з пунктом 5.1.5 П(С)БО 7 "Основні засоби", транспортні засоби класифікуються окремою групою основних засобів. Облік наявності та руху транспортних засобів ведеться на субрахунку 105 "Транспортні засоби". Витрати підприємства на придбання матеріальних активів, облік яких проводиться на рахунку 10 "Основні засоби", відображаються на субрахунку 152 "Придбання (виготовлення) основних засобів".

Відображення придбаних транспортних засобів на балансі підприємства відбувається шляхом занесення їх на дебет субрахунка 105 "Транспортні засоби" за їх первісною вартістю. Перелік витрат, які включаються у бухгалтерському обліку та враховуються у первісній вартості об'єкта основних засобів, наведений у пункті 8 П(С)БО 7 та схожий на перелік витрат, які включаються до первісної вартості об'єкта основних засобів згідно з ПКУ.

Підставою для внесення основних засобів на баланс є акт приймання-передачі (введення в експлуатацію) основних засобів. Зразок такого акта наведений у зразку 3.

Стандартні форми первинної облікової документації з обліку основних засобів затверджені наказом Мінстату від 29.12.95 року №352, але підприємство може розробити власні документи, які повинні мати обов'язкові реквізити, визначені законодавством про бухгалтерський облік та Положенням №88.

Нарахування амортизації розпочинається з місяця, наступного за місяцем, у якому транспортний засіб (автомобіль) стає придатним для корисного використання (з моменту введення в експлуатацію). Нарахування амортизації за виробничим методом починається з дати, яка настає за датою, коли об'єкт основних засобів став придатним для корисного використання.

1.3 Розробка технічного завдання

Для того, щоб зробити вимоги зрозумілими для обох сторін - замовника і розробника, вони повинні бути оформлені у вигляді документа. Вітчизняна практика використовує "Технічне завдання" (ТЗ), а західна - "Software Requirements Specification" (SRS) - специфікація програмних вимог. Фактично, ці документи є еквівалентними.

Створення технічного завдання є одним із перших і дуже важливих етапів у більшості проектів. Чітке і правильно сформульоване ТЗ допомагає уточнити вимоги замовника до роботи, визначити параметри та характеристики майбутнього продукту і надає основу для перевірки виконаної роботи.

Технічне завдання - це спеціальний документ, узгоджений між замовником і виконавцем, який містить вимоги, параметри і основні характеристики проекту, об'єкта чи системи. Крім того, воно може включати список вимог щодо тестування.

При написанні ТЗ важлива чіткість і конкретність - чим більше деталей у документі, тим краще. Після його розгляду ні замовник, ні команда розробників не повинні мати питань стосовно проекту. Якщо технічне завдання складено правильно, це допомагає уникнути непорозумінь між сторонами, скорочує час узгодження і мінімізує трудовитрати.

Незважаючи на індивідуальний підхід до кожного проєкту, практично будь-який приклад ТЗ на розробку ПО буде включати наступні пункти:

1. Загальні відомості про майбутнє продукту (в тому числі область застосування програмного забезпечення, його завдання, цілі та т. Д.).

2. Функціональні характеристики ПО (в цей же пункт включаються список модулів, віджетів, умови експлуатації та ін.).

3. Характеристика об’єктів автоматизації.

4. Вимоги до документування, включаючи вимоги до вихідного коду.

5. Опис стадій розробки ПО (поетапне зміст робіт і терміни виконання завдань).

6. Порядок контролю (в розділі вказуються умови здачі готового продукту і вимоги до приймання).

При описі призначення програмного забезпечення також необхідно врахувати основні сценарії використання користувачами, детально описати ролі користувачів, їх повноваження та доступи, вказати математичні методи та моделі, а також типові та поточні алгоритми розробки. У випадку потреби у інтеграції з іншими програмами або використання інших ресурсів, слід навести відповідний перелік. Технічне завдання може містити різноманітні зведені таблиці, схеми, діаграми, розрахунки та інші матеріали, що використовуються під час проєктування.

Зміст технічного завдання наведено у додатку В

РОЗДІЛ 2. ПРОЄКТУВАННЯ

2.1 Розробка інфологічної моделі предметної області

Мета концептуального інфологічного проєктування полягає у створенні логічної моделі бази даних, яка відображає об'єкти досліджуваної області та відносини між ними, незалежно від вибору конкретної системи управління базами даних (СУБД). Ця модель узагальнює інформаційні потреби майбутніх користувачів інформаційної системи (ІС).

Процес проектування бази даних починається з аналізу предметної області, під час якого важливо ідентифікувати всі ключові об'єкти, визначити їхні характеристики та встановити відносини між ними. Цей початковий етап називається концептуальним проектуванням бази даних.

Опис предметної області в процесі проектування здійснюється за допомогою трьох основних конструктивних елементів: сутності, атрибута та зв'язку.

Сутність - це узагальнений термін, який використовується для позначення групи схожих об'єктів, дані про які потрібно збирати та зберігати в інформаційній системі. Кожна сутність має унікальну назву та набір атрибутів, які описують її властивості.

Атрибут - це названа характеристика сутності, яка може приймати певні значення з визначеного діапазону дозволених значень. Атрибути представляють властивості сутностей.

Зв'язок - це візуально представлена асоціація, яка вказує на наявні відносини між двома чи більше сутностями.

Концептуальне або інфологічне проєктування - це процес створення моделі інформації, використовуваної на підприємстві, який не залежить від фізичних аспектів її представлення. Цей процес призводить до створення семантичних моделей, які відображають інформаційний зміст певної предметної області або концептуальної моделі даних для аналізованої частини підприємства. Модель розробляється на основі аналізу інформації, взятої зі специфікації користувачів та

сформованої під час підготовки до проектування моделі для користувачів бази даних. Ця модель є повністю незалежною від логічних та фізичних деталей реалізації бази даних. На цьому етапі відбувається сприйняття реальності, її абстрагування, дослідження та опис предметної області, що включає визначення об'єктів, їх властивостей та зв'язків, які будуть відображені у проекті бази даних [2].

Концептуальне проєктування складається з таких кроків:

- аналіз вимог до бази даних з боку користувачів та створення так званої користувацької моделі, що включає вивчення та систематизацію інформації про користувачів, їхні інформаційні потреби, джерела інформації та реальні інформаційні потоки;

- виявлення ключових об'єктів (сутностей), які мають бути представлені в моделі, враховуючи специфіку предметної області, цілі замовника та існуючі бізнес-правила;

- визначення атрибутів сутностей і приведення їх у відповідність до вимог майбутньої об'єктної моделі;

- визначення зв'язків між сутностями та їх властивостей;

- вибір типу інфологічної (об'єктної) моделі і її розробка, зазвичай за допомогою моделі «сутність – зв'язок» (ER-діаграма);

- перевірка моделі даних;

- побудова розподіленої моделі даних;

- вибір моделі організації даних у концептуальній схемі.

Для вирішення завдання у дипломному проєкті використовували такі об’єкти:

Транспорт. Властивості: Марка, модель, рік, реєстраційний номер, стан транспорту, вартість, термін експлуатації (роки), метод розрахунку амортизації.

Таблиця 2.1. Транспорт

Vehicles
Make
Model
Year
RegistationNumber
VehicleStatus
InitalCost
UsefulLife
DeprecationMethod

Зображення транспорту. Властивості: Код транспорту, шлях до зображення, опис.

Таблиця 2.2 Зображення транспорту

VehicleImages
VehicleID
ImagePath
Description

Транспорт водії. Властивості: Код транспорта, код водія.

Таблиця 2.3 Транспорт водії

VehicleDrivers
VehicleID
DriverID

Обслуговування. Властивості: Код транспорта, дата, опис, вартість.

Таблиця 2.4 Обслуговування

Maintenance
VehicleID
Date
Description
Cost

Страхування. Властивості: Код транспорту, номер полісу, дата початку, дата завершення, вартість, компанія.

Таблиця 2.5 Страхування

Insurance
VehicleID
Date
Description
Cost

Паливо. Властивості: Код транспорта, дата, літраж, ціна за літр.

Таблиця 2.6 Паливо

Fuel
VehicleID
Date
Liters
CostPerLiter

Користувачі. Властивості: Логін, пароль, ім’я, прізвище, електронна пошта, дата створення облікового запису, роль.

Таблиця 2.7 Користувачі

Users
UserID
Username
PasswordHash
FirstName
LastName
Email
CreateAt
Role

Водії. Властивості: Ім’я, прізвище, номер водійських прав, контактна інформація.

Таблиця 2.8 Водії

Drivers
FirstName
LastName
LicenseNumber
ContactInfo

Амортизація. Властивості: Код транспорта, Дата амортизації, сума амортизації.

Таблиця 2.9 Амортизація

Depreciation
VehicleID
DepreciationDate
DepreciationAmount

Модель «сутність-зв'язок» (англ. Entity-relationship model або entity-relationship diagram) представляє собою метод моделювання даних, що дозволяє створювати концептуальні схеми з використанням стандартизованих блоків. Як мета-модель, ER-модель служить засобом для опису різних моделей даних. Серед інших підходів до представлення знань, модель «сутність-зв'язок» виділяється як універсальний інструмент для уніфікованого відображення даних, який не залежить від конкретного програмного забезпечення. Особливо значимим є аспект, що з ER-моделі можливо генерувати інші існуючі моделі даних, такі як ієрархічна, мережева, реляційна та об'єктна, що робить її найбільш універсальною з усіх доступних моделей [3]. ER-діаграма даної предметної області наведена на рис. 2.1.

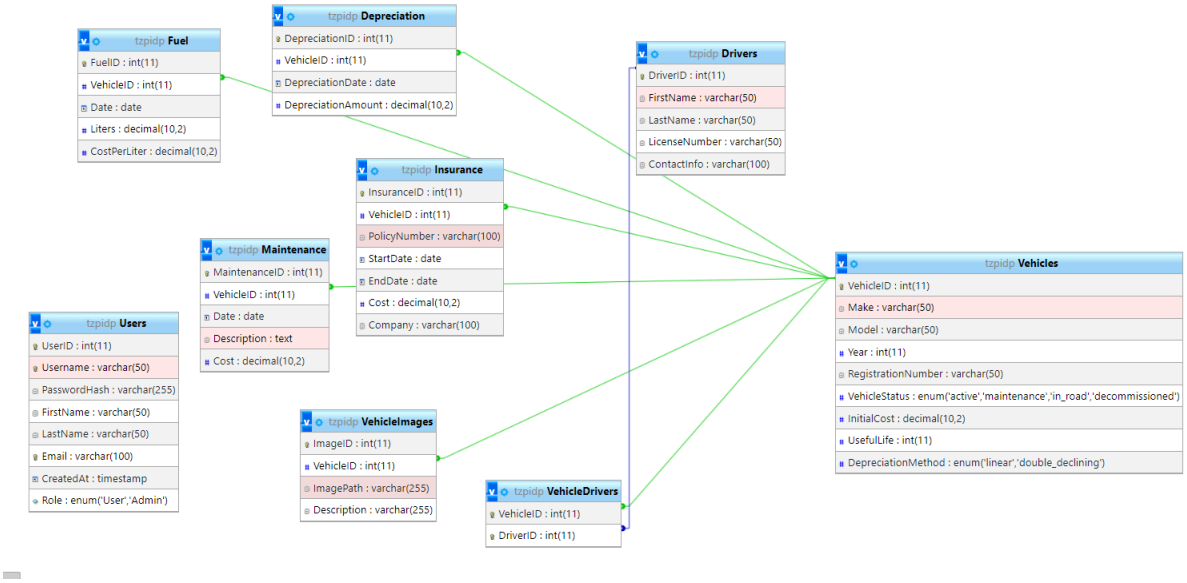

Рисунок 2.1 – ER-діаграма

У даній базі даних визначено наступні зв’язки (табл.2.10).

Таблиця 2.10 – Типи зв’язків

Таблиця 1	Типи зв’язків	Таблиця 2	Опис
Vehicles	1:N	VehicleImages	Один транспортний засіб може мати кілька зображень, але кожне зображення пов'язане тільки з одним транспортним засобом.
Vehicles	1:N	Maintenance	Один транспортний засіб може мати кілька записів обслуговування.
Vehicles	1:N	Insurance	Один транспортний засіб може мати кілька страховок.
Vehicles	1:N	Fuel	Один транспортний засіб може мати кілька записів про заправки.

Подовження таблиці 2.10

Vehicles	1:N	Depreciation	Один транспортний засіб може мати багато записів про амортизацію.
Vehicles	N:N	VehicleDrivers	Один транспортний засіб може бути використаний кількома водіями, і один водій може використовувати кілька транспортних засобів.

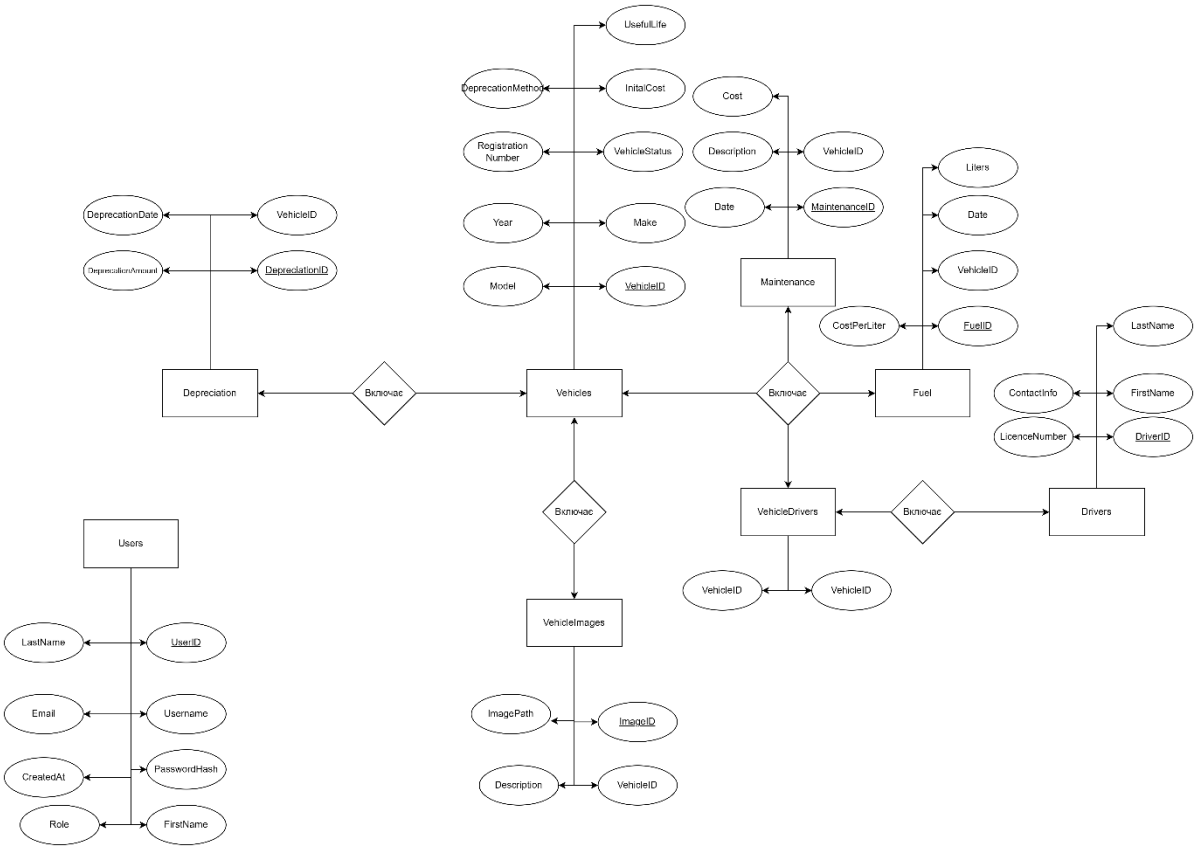

Рис. 2.2 – Інфологічна модель

2.2. Даталогічне проєктування

Даталогічна модель представляє собою базу даних, організовану на логічному рівні та адаптовану до конкретної системи управління базами даних (СУБД). Перед розробкою даталогічної моделі важливо визначитись з вибором СУБД, оскільки кожна система має свої обмеження та особливості, що впливають на структуру логічної моделі. Тому перед проектуванням необхідно детально дослідити можливості та вимоги обраної СУБД, а також фактори, які можуть вплинути на модель даних [4].

На відміну від даталогічної моделі, інфологічна модель включає у себе всі необхідні дані про предметну область, що мають значення для проєктування бази даних. Проте, не всі сутності, які включені в інфологічну модель, потрібно обов'язково відображати в даталогічній моделі. Перший крок у створенні

даталогічної моделі полягає у вирішенні, які дані будуть фактично зберігатися у базі даних.

Основні фактори, які впливають на процес даталогічного проєктування з боку СУБД, включають:

1. Тип логічної моделі, яку підтримує обрана СУБД. В даний час на ринку програмних продуктів та в сфері автоматизації економічних розрахунків переважають реляційні СУБД. Окрім реляційних, також існують єрархічні та мережеві моделі баз даних, кожна з яких має свої особливості та застосування.

2. Особливості фізичної організації даних в середовищі обраної СУБД. Наприклад, в СУБД, таких як Paradox або системи на базі dBASE, база даних структурована як набір взаємопов'язаних файлів у форматах DT і DBF. Всі інші компоненти, такі як форми та звіти, зберігаються у відповідних окремих файлах.

У середовищі phpMyAdmin для управління MySQL, база даних структурована у вигляді окремих таблиць, які зберігаються на сервері. Відмінності від Microsoft Access полягають у тому, що у phpMyAdmin всі дані знаходяться в розподіленій системі на сервері, а не в одному файлі. Це дозволяє керувати великими обсягами даних ефективніше і забезпечує кращу масштабованість та доступність даних.

Під час проектування бази даних у середовищі phpMyAdmin з MySQL важливо враховувати не тільки правила побудови логічної моделі, але й особливості використання SQL для оптимізації зберігання та запитів до даних. Керування індексами, типами даних та структурою таблиць може істотно вплинути на продуктивність роботи з базою даних.

3. Обмеження, накладені СУБД щодо кількості ресурсів. До таких належать максимальна кількість рівнів ієрархії в ієрархічних моделях, можливість створення певної кількості полів, записів або файлів. Поки що не існує стандартизованих методів, які б дозволяли точно провести даталогічне проектування, тому успіх проекту залежить від досвіду та кваліфікації фахівців, які займаються розробкою [5].

В результаті даталогічного проектування можливо отримати декілька варіантів логічної моделі даних, тому критично важливим є процес оцінки цих моделей та вибір найбільш оптимального рішення. Отримані моделі слід оцінювати з огляду на їх відповідність наявним обчислювальним ресурсам. У разі несумісності з цими ресурсами може знадобитися перепроектування бази даних. Також важливо перевірити, наскільки логічна модель задовольняє умовам виконання запитів користувачів і вимогам прикладних програм, тобто її адекватність інформаційній моделі предметної області.

На даний момент реляційна модель даних, яка представляє інформацію у вигляді двовимірних таблиць, залишається найпопулярнішою. Існує велика кількість систем управління реляційними базами даних, які спрощують процес розробки баз даних, складених з декількох таблиць, пов'язаних між собою за допомогою ключових полів. Кожне поле в таблиці реляційної моделі містить фактичні дані, які відносяться до сутностей.

Для бази даних створено наступні таблиці:

1. Водії;
2. Автомобілі;
3. Водії-Автомобілі;
4. Зображення автомобілів;
5. Амортизація;
6. Обслуговування;
7. Страхування;
8. Користувачі;
9. Паливо.

2.3 Розробка логічної схеми бази даних. Нормалізація

Логічне проектування визначається як розробка структури бази даних на логічному рівні. Цей процес має на меті створити структуру інформаційної системи, яка мінімізує дублювання даних і запобігає різним аномаліям, які можуть виникати під час оновлення, видалення або введення даних. Цей процес відомий як нормалізація.

Нормалізація відносин в базах даних - це застосування формальних правил, які обмежують спосіб структурування даних, щоб уникнути їхнього повторення, забезпечити послідовність і знизити витрати на обслуговування бази даних. Хоча надмірність і продуктивність інформаційної системи часто знаходяться у протиріччі, повне видалення надмірності іноді може бути не доцільним через можливі негативні наслідки для ефективності системи.

Процедура нормалізації бази даних зосереджена на усуненні зайвої повторюваності даних та виявленні функціональних залежностей між атрибутами. Завдяки усуненню повторень даних забезпечується більш компактне зберігання, а також запобігаються можливі аномалії при вставці, видаленні чи оновленні даних після їхнього фізичного розміщення в базі. Функціональна залежність описує взаємозв'язок між атрибутами так, що одне значення атрибуту визначає значення іншого, що часто позначається як A→B. Це лежить в основі утворення зв'язків між сутностями бази даних типу один до одного (1:1) або один до багатьох (1:M).

Нормалізація, або правила Кодда, які використовуються для цього процесу, є простими, але строгими. Вони допомагають досягти різних рівнів відповідності теорії реляційних баз даних через послідовне застосування рівнів нормалізації: перша нормальна форма (1НФ), друга нормальна форма (2НФ), третя нормальна форма (3НФ), нормальна форма Бойса-Кодда (БКНФ), четверта (4НФ) та п'ята нормальна форма (5НФ). Однак, через жорсткі вимоги до продуктивності, не всі реляційні СУБД підтримують виконання всіх п'яти форм. На практиці часто обмежуються першими трьома рівнями, оскільки повна нормалізація може знизити

продуктивність системи до неприйнятних рівнів через необхідність з'єднання численних таблиць при виконанні запитів.

Перша нормальна форма (1НФ) досягається, коли всі атрибути в базі даних містять тільки атомарні (неділимі) значення, і кожне поле містить однакову кількість даних у кожному записі без повторень.

Друга нормальна форма (2НФ) вимагає, щоб база даних вже була в 1НФ, і всі неключові атрибути були повністю залежні від первинного ключа, не допускаючи часткових залежностей будь-якого з атрибутів від будь-якої частини складеного ключа.

Процедура нормалізації бази даних полягає у видаленні зайвих дублікатів даних і виявленні функціональних залежностей між атрибутами. Завдяки цьому процесу можливе запобігання таким аномаліям, як вставка, видалення та оновлення, тим самим гарантуючи компактність даних. Функціональна залежність створює відношення між атрибутами, де один атрибут визначає значення іншого, що зазвичай позначається як A→B.

Процедура нормалізації бази даних полягає у видаленні зайвих дублікатів даних і виявленні функціональних залежностей між атрибутами. Завдяки цьому процесу можливе запобігання таким аномаліям, як вставка, видалення та оновлення, тим самим гарантуючи компактність даних. Функціональна залежність створює відношення між атрибутами, де один атрибут визначає значення іншого, що зазвичай позначається як A→B.

Третя нормальна форма (3НФ):

Відношення знаходиться у третій нормальній формі, якщо воно вже відповідає другій нормальній формі і не містить транзитивних залежностей між неключовими атрибутами. Це означає, що жоден неключовий атрибут не залежить від інших неключових атрибутів.

Четверта нормальна форма (4НФ):

Відношення вважається представленим у четвертій нормальній формі, якщо воно вже відповідає нормальній формі Бойса-Кодда і у відношенні відсутні

нетривіальні багатозначні залежності, крім випадків, коли ліва частина залежності є суперключем.

П'ята нормальна форма (5НФ):

П'ята нормальна форма досягається, коли відношення не може бути поділене на більш дрібні відносини без втрати інформації. Це стосується декомпозиції відношень за допомогою проєктування.

Властивості першої нормальної форми (1НФ) включають:

- унікальність кортежів без дублікатів;
- невпорядкованість кортежів і атрибутів;
- відсутність групувань атрибутів з однаковими значеннями;
- атомарність значень атрибутів.

На практиці, функціональні залежності відображають взаємозв'язки, які визначаються у межах предметної області, і вносять додаткові обмеження для збереження даних у базі. Наприклад, знаючи ідентифікаційний номер співробітника, можна визначити його прізвище. Ці залежності вимагають, щоб при модифікації бази даних перевірялись всі обмеження, встановлені функціональними залежностями.

Етапи розробки логічної моделі реляційної бази даних:

1 етап. Приведення до 1НФ: Спочатку визначаються і встановлюються відносини, що відповідають поняттям предметної області, всі з яких автоматично знаходяться в 1НФ.

2 етап. Приведення до 2НФ: Якщо виявляються атрибути, залежні від частини складного ключа, їх виносять в окремі відносини разом з цією частиною ключа.

3 етап. Приведення до 3НФ: Проводиться декомпозиція відносин, в яких неключові атрибути залежать від інших неключових атрибутів, забезпечуючи, що всі неключові атрибути взаємно незалежні [6].

2.4 Об’єктно-орієнтоване моделювання предметної області із використанням CASE-засобів

Для створення діаграм було використано безкоштовний онлайн сервіс draw.io, який надає інструмент для побудови UML-діаграм з зручними функціями, що дозволяють швидко малювати їх. Інтуїтивно зрозумілий інтерфейс, можливість створювати елементи за один клацання миші та автоматизоване з'єднання спрощують процес створення діаграм.

Основні переваги включають доступ до великої бібліотеки фігур для всіх типів UML-діаграм, можливість редагувати професійно розроблені приклади UML-схем навіть миттєво, а також тематичні колірні палітри та стилі для налаштування вигляду UML-діаграм.

UML є графічною мовою для опису архітектури системи. Не кодуючи програму, а описуючи її за допомогою діаграм, UML нагадує шкільну алгоритмічну мову. Діаграми UML складаються з об'єктів і зв'язків між ними.

UML має чотирирівневу архітектуру:

- мета-метамодель;
- метамодель;
- модель;
- користувацькі об'єкти.

Користувацькі об'єкти визначають об'єкти конкретної предметної області, наприклад: телевізор, монітор. Модель є певним поглядом на предметну область.

В UML існують такі моделі:

- Модель випадків використання (use case model): Призначена для опису вимог до системи та підсистем.
- Модель класів (class model): Використовується для опису статичної структури системи, включаючи ієрархію класів і стосунки між ними.

•Модель взаємодій (collaboration model) і сценарії (sequence model): Призначені для опису механізмів взаємодії об'єктів системи, що реалізують певну функцію.

•Поводінкова модель (behaviour model): Діаграми переходів і станів, призначені для опису алгоритмів поведінки об'єктів системи.

•Модель процесів (deployment model): Описує фізичну архітектуру системи та розподіл процесів по процесорах у фізичному проекті системи.

•Модель програмних модулів (component model): Описує розподіл класів і об'єктів системи по модулях у фізичному проекті системи.

•Модель дій (activity model): Призначена для опису алгоритмів системи (для методів класів, для декількох класів) і є варіантом поведінкової моделі без повідомлень.

Створення діаграми прецедентів

Для початку потрібно визначити основних акторів:

- Адміністратор – підтримує дані в ПЗ;
- БД – база даних, що містить інформацію про транспортні засоби;
- Користувач – переглядає інформацію.

Адміністратор виконує роль головної особи, що здійснює основні процеси в системі. База даних є об’єктом, над яким виконуються дії, але вона також має свої функції. Ці два актори взаємодіють між собою, утворюючи взаємозалежні дії.

Далі необхідно з'єднати варіанти використання і акторів зв'язками. У своїй діаграмі я використав два типи зв’язків: залежність (dependency) та односпрямовану асоціацію (unidirectional association). Завдяки цим зв’язкам вдалося створити зв'язок між прецедентами і акторами.

Після розміщення на діаграмі прецедентів всіх об'єктів, необхідно додати до варіантів використання їх специфікації. Діаграма прецедентів представлена на рис. 2.3.

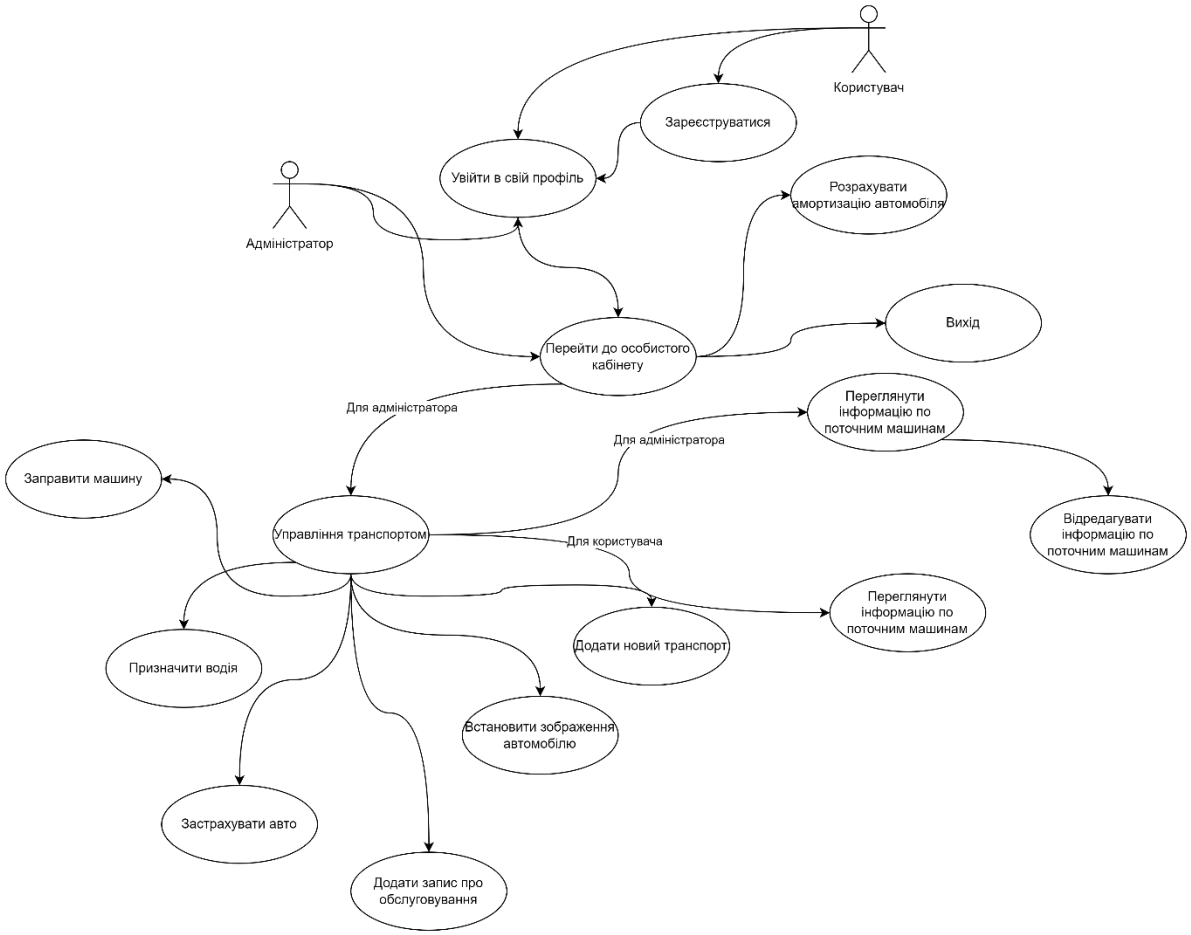

Рис. 2.3 – Діаграма прецедентів

Створення діаграми діяльності

Для моделювання процесу виконання операцій в UML використовують діаграми діяльності (Activity Diagrams), які акцентують увагу на послідовності виконання певних дій (елементарних операцій).

Основною метою використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно зобразити алгоритми їхнього виконання. Кожен стан дії може відповідати виконанню операції деякого класу, а вид діяльності – послідовності таких дій.

На етапі моделювання прецедентів діаграми діяльності зображають потоки функцій керування, зазначають, які гілки процесу керування можуть виконуватись паралельно, і визначають альтернативні шляхи досягнення мети. Потоки керування можуть містити декілька прецедентів або протікати у рамках одного прецеденту.

Діаграма діяльності наведена на рис. 2.4.

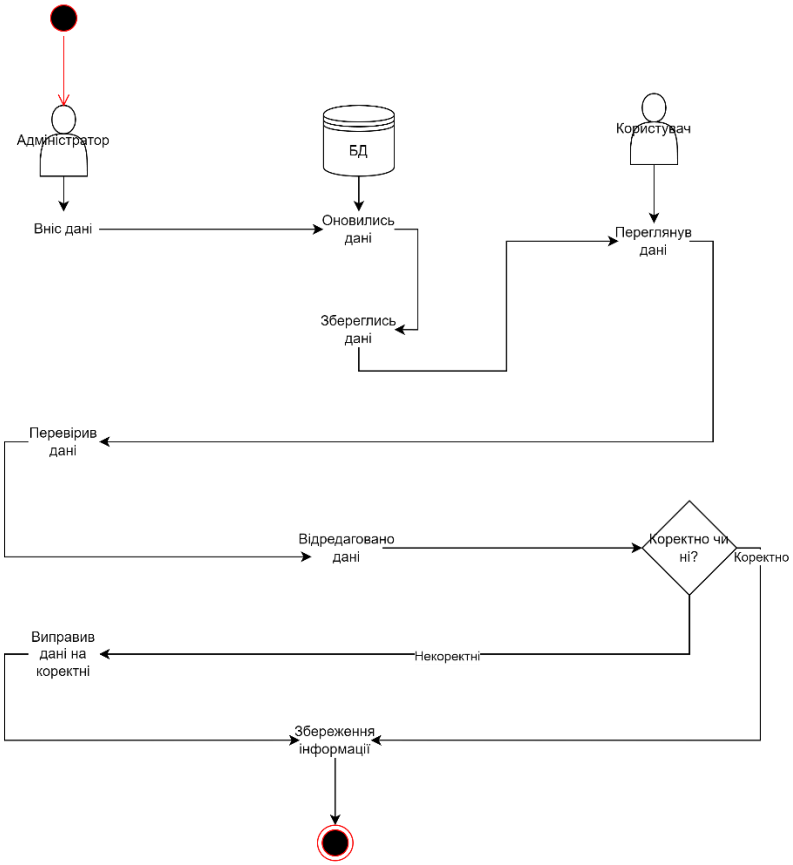

Рис.2.4. – Діаграма діяльності

Діаграма компонентів - це вид структурної діаграми, яка показує організацію системи на рівні її компонентів або модулів, а також зв'язки між ними. Компоненти можуть бути фізичними або логічними частинами програми або системи, які мають чітко визначені межі та інтерфейси.

Основна мета діаграми компонентів полягає в тому, щоб допомогти розробникам та архітекторам зрозуміти, як система організована на рівні її компонентів, як вони взаємодіють між собою та які вони мають функціональність.

Деякі основні елементи, які можуть бути відображені на діаграмі компонентів, включають:

•Компоненти - фізичні або логічні частини системи, які мають свою функціональність та інтерфейси.

•Інтерфейси - місця, де компоненти взаємодіють один з одним або з іншими системами. Інтерфейси можуть бути представлені як методи, параметри, повідомлення або інші способи обміну даними.

•Залежності - зв'язки між компонентами, які вказують, що один компонент використовує інший або залежить від нього.

•Артефакти - додаткові ресурси, такі як бібліотеки, файли конфігурації або інші компоненти, які необхідні для роботи системи.

Діаграми компонентів допомагають зрозуміти структуру системи, допомагають при розробці, тестуванні та супроводженні програмного забезпечення. Вони також можуть служити як основа для подальшого проектування системи та визначення її архітектури.

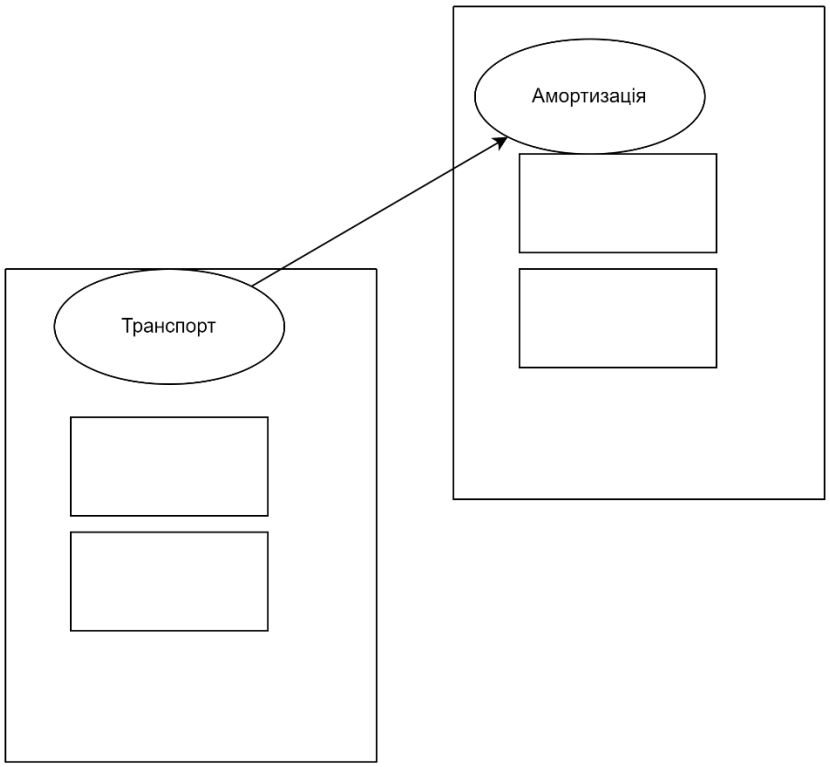

Рис. 2.5 – Діаграма компонентів

РОЗДІЛ 3. РЕАЛІЗАЦІЯ

3.1. Розробка веб-застосунку

Для розробки використовувався Open Server – це платформа для локальної веб-розробки, яка дозволяє швидко налаштувати та запустити веб-сервер, базу даних та інші середовища, необхідні для розробки веб-додатків на PHP.

Основні характеристики Open Server включають:

Вбудовані сервери: Open Server містить в собі Apache та Nginx як сервери веб-додатків, що дозволяє вибрати той, який найкраще підходить для конкретного проекту.

Підтримка PHP та MySQL: Open Server надає можливість використовувати PHP для розробки серверної частини додатків та MySQL для баз даних.

Простий використання: Завдяки інтуїтивно зрозумілому інтерфейсу користувача, Open Server легко встановлюється та налаштовується.

Підтримка віртуальних хостів: Open Server дозволяє створювати та налаштовувати віртуальні хости для розробки кількох веб-сайтів одночасно.

Можливість налаштування PHP та інших середовищ: Користувач може змінювати конфігурації PHP, Apache, Nginx та інших середовищ зручним для себе способом.

Підтримка різних версій PHP: Open Server дозволяє встановлювати та використовувати різні версії PHP для різних проектів.

А також було використано мову програмування PHP та базу даних, що було створено мовою запитів SQL у середовищі phpMyAdmin.

phpMyAdmin - це безкоштовний веб-інтерфейс для управління базами даних MySQL. Він надає зручний інтерфейс користувача для виконання різних операцій з базами даних, таких як створення, редагування, видалення таблиць, введення та редагування даних, виконання SQL-запитів, керування користувачами та привілеями і багато іншого.

Деякі основні функції phpMyAdmin включають у себе:

Створення та управління базами даних: Користувач може створювати нові бази даних або видаляти існуючі, а також переглядати інформацію про них.

Створення та управління таблицями: Користувач може створювати нові таблиці, редагувати їх структуру, видаляти та керувати іншими параметрами.

Введення та редагування даних: За допомогою phpMyAdmin можна додавати нові записи у таблиці, редагувати існуючі дані, видаляти записи тощо.

Виконання SQL-запитів: Користувач може виконувати SQL-запити безпосередньо через інтерфейс phpMyAdmin.

Керування користувачами та привілеями: PhpMyAdmin надає можливість керувати доступом користувачів до баз даних та надавати їм різні привілеї.

Імпорт та експорт даних: Користувач може імпортувати дані у базу даних з різних форматів файлів (наприклад, SQL, CSV тощо) або експортувати дані у зручний формат для збереження або обміну.

phpMyAdmin є потужним інструментом для адміністрування баз даних MySQL через веб-інтерфейс, що робить процес управління базами даних більш зручним та ефективним.

Таким чином, PHP та MySQL - це два ключові компоненти для розробки веб-додатків.

PHP - це скриптова мова програмування загального призначення, яка спеціально призначена для розробки веб-додатків.

Вона використовується для створення динамічного вмісту веб-сторінок, обробки форм, роботи з базами даних, керування сесіями користувачів та багато іншого.

PHP вбудовано в HTML, що дозволяє вставляти PHP-код безпосередньо в HTML-сторінки.

MySQL - це відкрита система управління базами даних (СУБД), яка використовує мову запитів SQL для роботи з даними.

Вона забезпечує можливість зберігати, оновлювати, видаляти та вибирати дані з бази даних.

MySQL часто використовується разом з PHP для розробки веб-додатків, оскільки вони добре інтегруються та дозволяють створювати повноцінні веб-застосунки.

У розробці веб-застосунків з використанням PHP та MySQL вони часто використовуються разом. PHP використовується для обробки запитів користувачів, взаємодії з базою даних та генерації HTML-вмісту, а MySQL - для зберігання та управління даними. Такий підхід дозволяє створювати потужні веб-додатки з можливістю взаємодії з користувачами та обробки великих обсягів даних.

SPA (Single Page Application) - це тип веб-додатку або сайту, який працює в одному HTML-документі, а вміст сторінки динамічно оновлюється за допомогою JavaScript без перезавантаження сторінки. В SPA весь код, що потрібен для роботи додатку, завантажується разом з першим запитом на сервер і потім взаємодіє з користувачем, змінюючи лише ту частину сторінки, яка є необхідною для відображення нового вмісту.

Для розробки SPA зазвичай використовуються такі технології:

Хоча сторінка завантажується один раз, HTML використовується для визначення структури додатку.

CSS використовується для стилізації вмісту та розмітки сторінки.

Весь функціонал SPA реалізується за допомогою JavaScript. Він взаємодіє з користувачем, оброблює події, виконує запити до сервера та оновлює вміст сторінки.

Для розробки SPA часто використовуються фреймворки та бібліотеки, такі як React.js, Angular, Vue.js тощо. Вони надають зручний інструментарій для створення динамічного вмісту, маршрутизації та управління станом додатку.

SPA може взаємодіяти з сервером за допомогою REST або GraphQL API для отримання та відправлення даних без перезавантаження сторінки.

Основні переваги SPA включають покращену продуктивність та швидкодію завдяки відсутності перезавантаження сторінки, зручний інтерфейс користувача та зниження навантаження на сервер. Однак важливо враховувати питання, такі як SEO-оптимізація та керування пам'яттю, при розробці SPA.

JavaScript (JS) - це високорівнева, інтерпретована мова програмування, яка використовується для створення динамічних інтерактивних веб-сайтів. Однак, вона також використовується для розробки мобільних додатків, настільних програм, інтерфейсів користувача та інших сфер програмування.

Основні характеристики JavaScript включають:

JavaScript може виконуватися на різних платформах, таких як веб-браузери, сервери, мобільні пристрої та інші.

JavaScript дозволяє створювати динамічні веб-сайти, які можуть змінюватися під час взаємодії користувача з ними, без необхідності перезавантаження сторінки.

JavaScript надає можливість реагувати на події, такі як кліки мишею, натискання клавіш, завантаження сторінки, тощо, і виконувати певні дії відповідно до цих подій.

JavaScript може змінювати структуру, вміст та стилізацію HTML-елементів на сторінці, що дозволяє створювати динамічні інтерфейси користувача.

JavaScript дозволяє виконувати асинхронні запити до сервера, такі як отримання або відправлення даних без перезавантаження сторінки, за допомогою технологій, таких як XMLHttpRequest (XHR) або fetch API.

JavaScript є мовою з великою кількістю бібліотек та фреймворків, які спрощують розробку різноманітних застосунків, таких як React.js, Angular, Vue.js тощо.

Загалом, JavaScript є однією з найпопулярніших мов програмування у веб-розробці і відіграє ключову роль у створенні сучасних динамічних веб-сайтів та веб-додатків.

3.2. Розробка бази даних

Було створено базу даних у phpMyAdmin мовою запитів SQL. У цій базі було створено 9 таблиць, що дозволяють користувачеві працювати з базою даних, тобто, додавати, редагувати інформацію про наявні на підприємстві транспортні засоби.

Базу даних у phpMyAdmin, як і таблиці можна створити двума шляхами, або за допомогою мови запитів SQL, або за допомогою конструктору.

Для того, щоб створити базу даних за допомогою конструктора (рис. 3.1), то необхідно обрати бази даних та увести ім’я для бази та задати їй кодування.

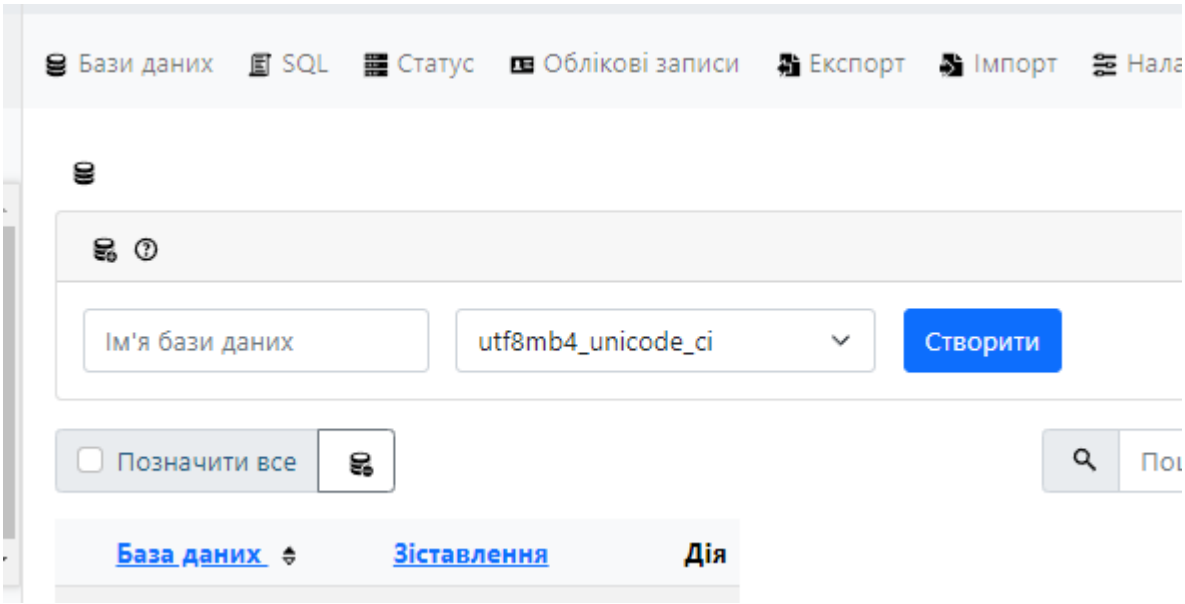

Рис. 3.1 – Створення бази даних за допомогою конструктора

Для того, щоб створити базу даних за допомогою мови запитів SQL необхідно обрати відповідне підменю SQL (рис. 3.2) та записати туди відповідний запит, що відповідає за створення бази даних.

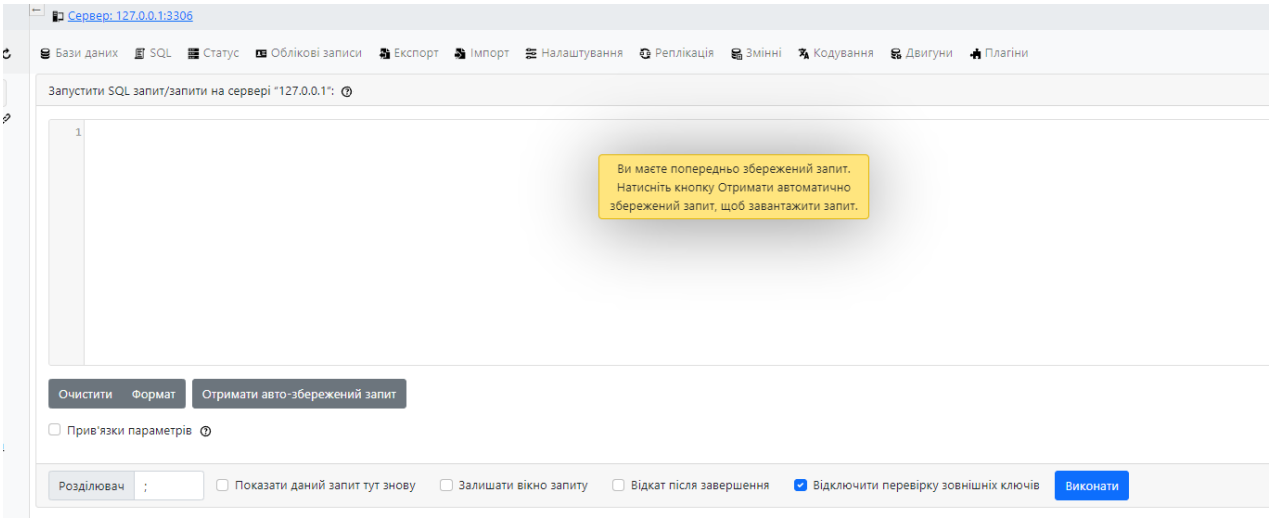

Рис. 3.2 – Створення бази даних за допомогою запитів

Після того, як запит уведено треба натиснути на кнопку «Виконати».

Щойно нова база створиться, треба перейти у неї натиснувши на її назву та створити таблиці. Таблиці можна створити за допомогою конструктора, або за допомогою мови запитів.

За допомогою конструктора (рис.3.3). Треба задати назву таблиці, кількість колонок, що буде у ній та натиснути створити.

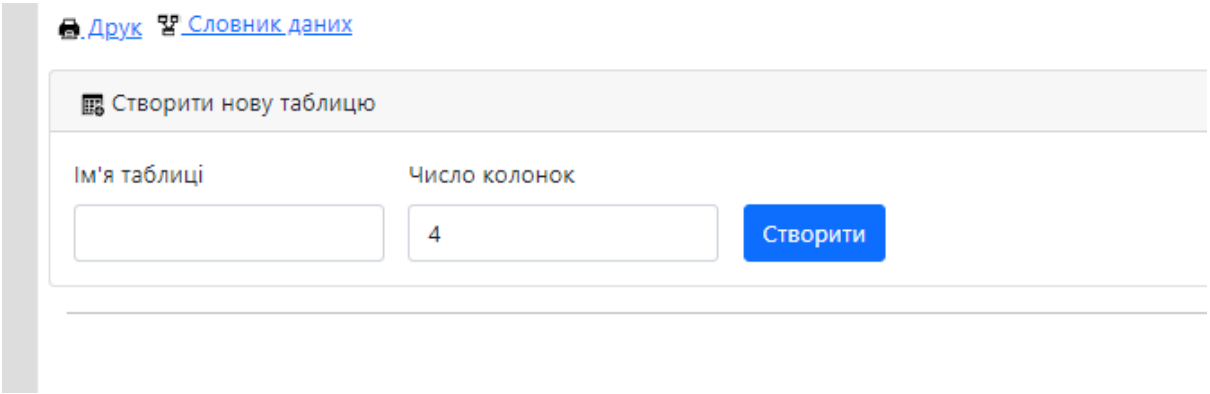

Рис. 3.3 – Створення таблиці за допомогою конструктора

Після натискання «Створити» відкриється меню створення таблиць (рис.3.4), де можна задати ім’я атрибутам, їхній тип даних, кодування, визначити чи буде нульове поле, вказати індекс, вказати первинний ключ чи зовнішній, налаштувати

автоінкремент, тощо. У цьому ж вікні можна обрати тип таблиць, можна здійснити попередній перегляд SQL запиту та зберегти нову таблицю.

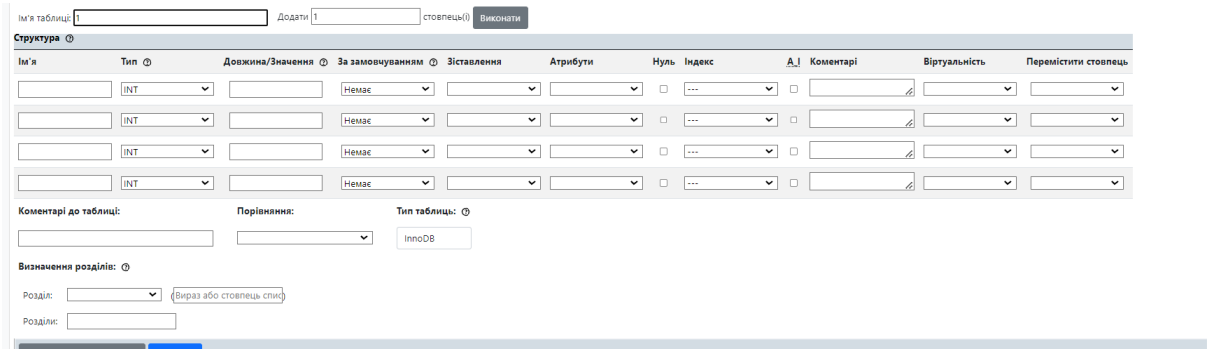

Рис. 3.4 – Створення таблиць

Створення таблиць за допомогою мови запитів (рис.3.5). Для цього необхідно обрати SQL та аналогічно як у базі даних увести відповідний запит та створити таблицю.

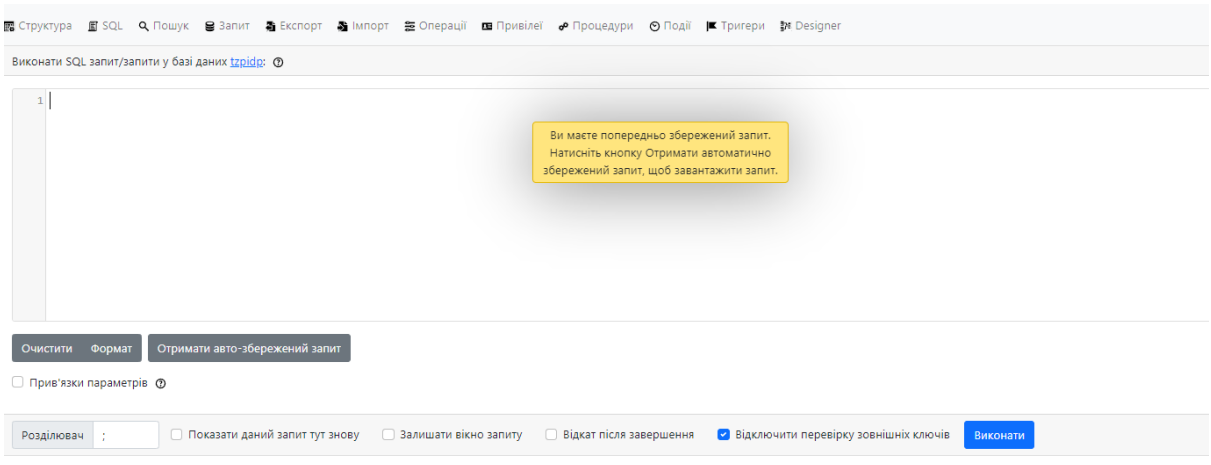

Рис.3.5 – Створення таблиць за допомогою запитів

Після успішного створення всіх необхідних таблиць можна обрати певну таблицю та переглянути її структуру. Структуру всіх створених таблиць наведено на рис. 3.6-3.14.

Структуру створених таблиць визначає ім’я, тип даних, зіставлення (кодування), атрибути, чи може бути поле нульовим, як і так, так і за замовчуванням, коментарі, якщо потрібно та додатково, наприклад, автоікремент.

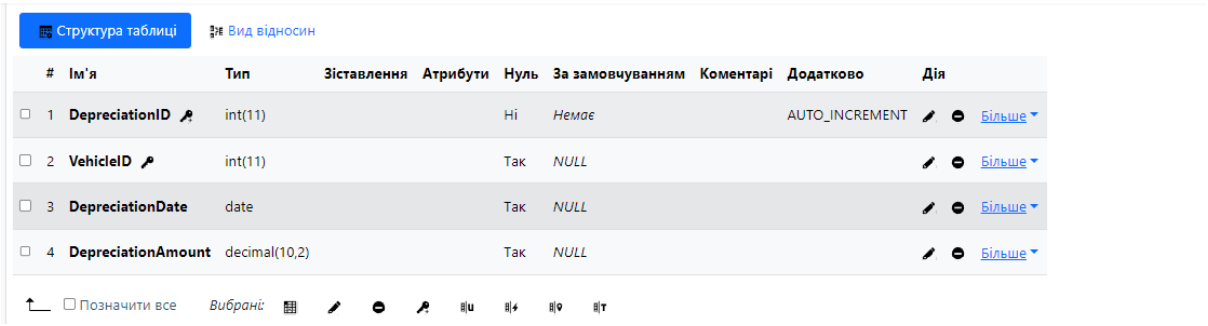

Рис. 3.6 – Структура таблиці «Амортизація»

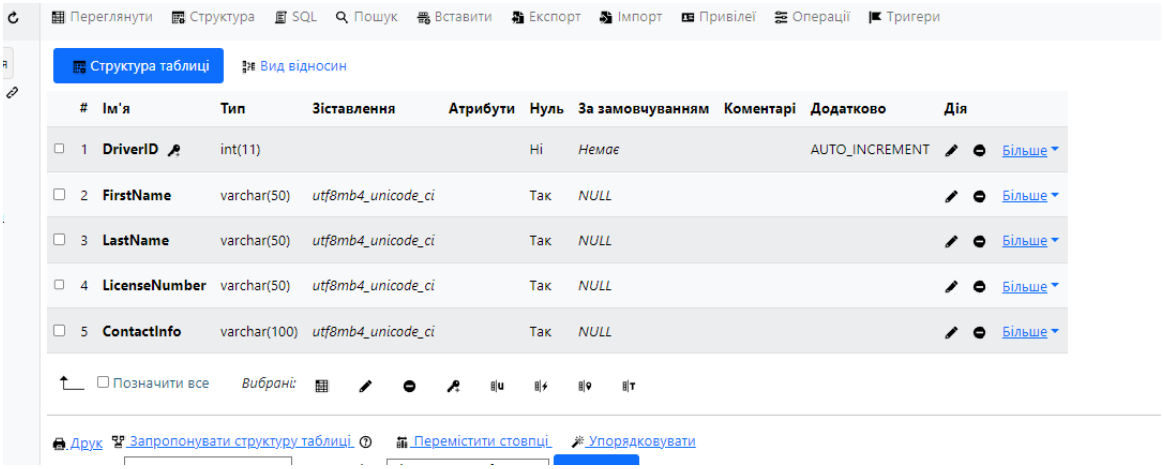

Рис. 3.7 – Структура таблиці «Водії»

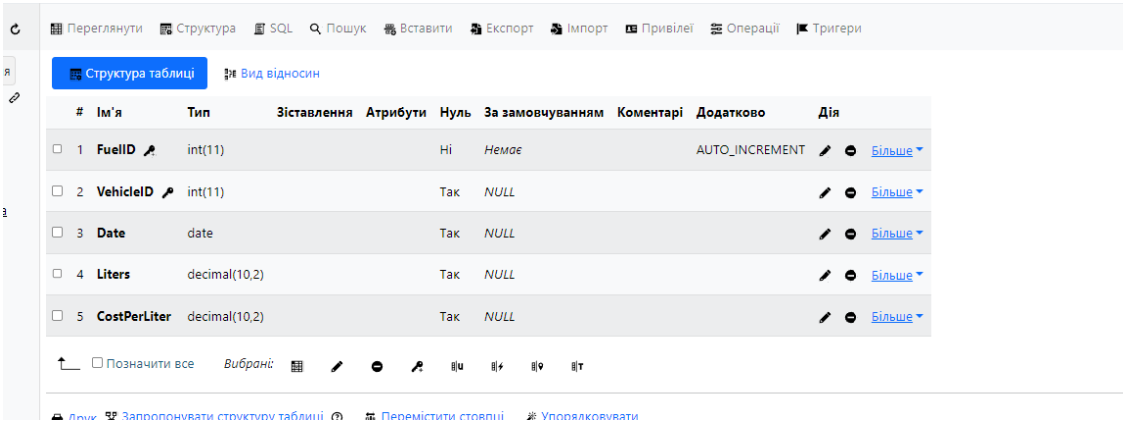

Рис. 3.8 – Структура таблиці «Паливо»

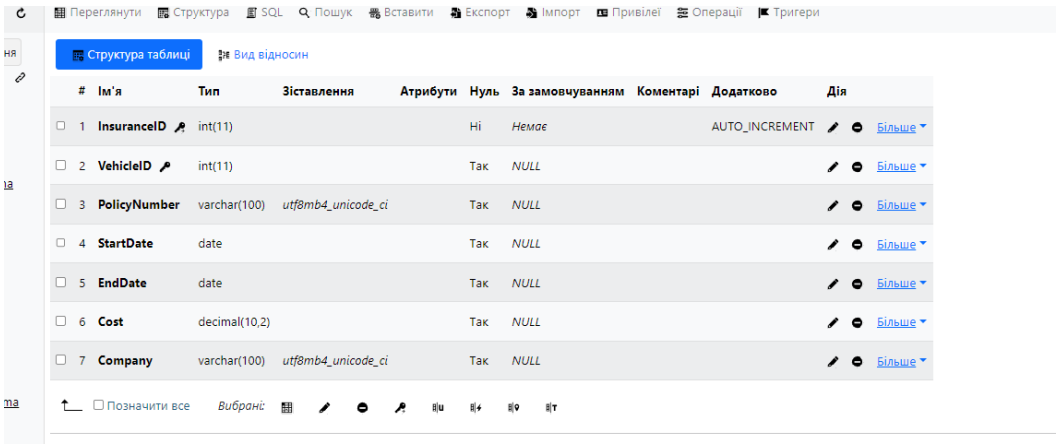

Рис. 3.9 – Структура таблиці «Страховка»

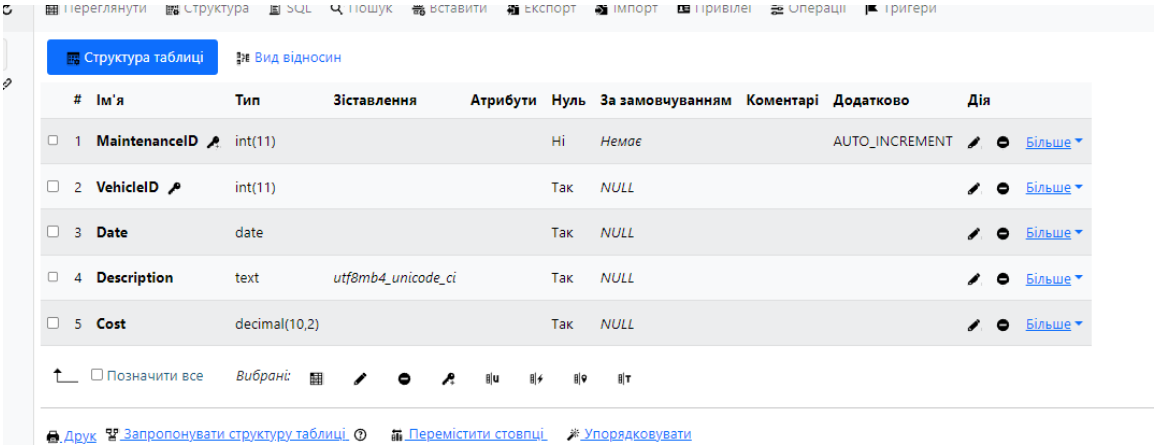

Рис. 3.10 – Структура таблиці «Обслуговування»

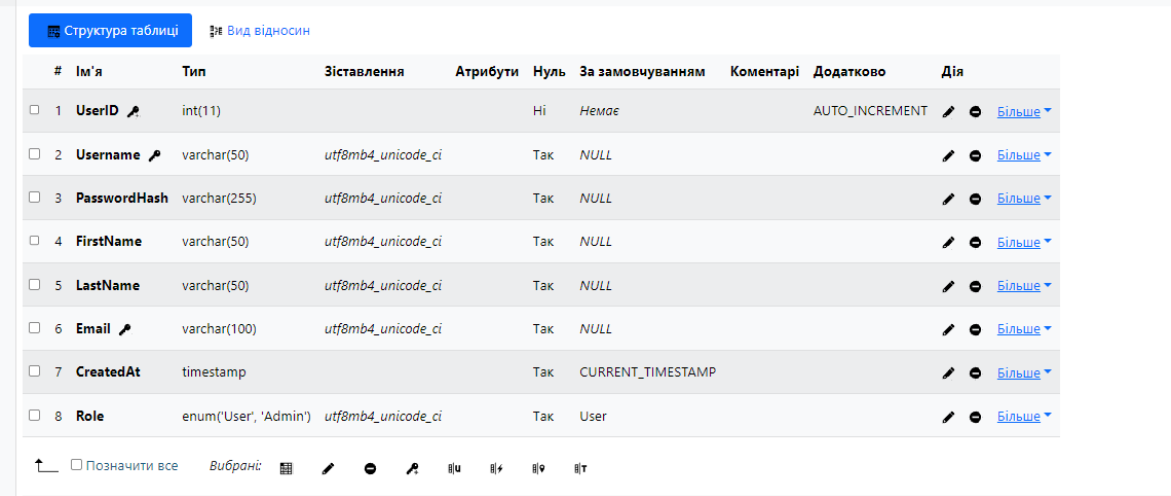

Рис. 3.11 – Структура таблиці «Користувача»

Структура таблиці | Вид відносин

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	VehicleID	int(11)			Ні	*Немає*			Більше
2	DriverID	int(11)			Ні	*Немає*			Більше

Позначити все *Вибрані:*

Рис. 3.12 – Структура таблиці «Водії-транспорт»

Переглянути | Структура | SQL | Пошук | Вставити | Експорт | Імпорт | Привілеї | Операції | Тригери

Структура таблиці | Вид відносин

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	ImageID	int(11)			Ні	*Немає*		AUTO_INCREMENT	Більше
2	VehicleID	int(11)			Так	*NULL*			Більше
3	ImagePath	varchar(255)	*utf8mb4_unicode_ci*		Так	*NULL*			Більше
4	Description	varchar(255)	*utf8mb4_unicode_ci*		Так	*NULL*			Більше

Позначити все *Вибрані:*

Рис. 3.13 – Структура таблиці «Зображення транспорту»

Структура таблиці | Вид відносин

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	VehicleID	int(11)			Ні	*Немає*		AUTO_INCREMENT	Більше
2	Make	varchar(50)	*utf8mb4_unicode_ci*		Так	*NULL*			Більше
3	Model	varchar(50)	*utf8mb4_unicode_ci*		Так	*NULL*			Більше
4	Year	int(11)			Так	*NULL*			Більше
5	RegistrationNumber	varchar(50)	*utf8mb4_unicode_ci*		Так	*NULL*			Більше
6	VehicleStatus	enum('active', 'maintenance', 'in_road', 'decommis...	*utf8mb4_unicode_ci*		Так	active			Більше
7	InitialCost	decimal(10,2)			Так	*NULL*			Більше
8	UsefulLife	int(11)			Так	*NULL*			Більше
9	DepreciationMethod	enum('linear', 'double_declining')	*utf8mb4_unicode_ci*		Так	linear			Більше

Позначити все *Вибрані:*

Рис. 3.14 – Структура таблиці «Транспорт»

Для підключення бази даних до застосунку було створено окремий скрипт database.php, який у всіх подальших файлах викликається за допомогою виклику функцій connectDB().

```
<?php
$mysqli=false;
function connectDB(){
```

```
global $conn;
$conn=new mysqli("localhost","root","","tzpidp");
$conn->query("SET NAMES 'utf8'");
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
}
?>
```

Повний лістинг наведено у додатку Б.

Логіка цього скрипту полягає у наступному:

В першому рядку встановлюється змінна $mysqli і присвоюється значення false. Вона буде використовуватися для зберігання об'єкта з'єднання з базою даних.

Далі визначається функція connectDB(), яка має доступ до глобальної змінної $conn. Функція приймає налаштування для підключення до бази даних (хост, ім'я користувача, пароль, назва бази даних).

У функції виконується створення об'єкту mysqli, якому передаються налаштування для підключення.

Далі встановлюється кодування для з'єднання з базою даних, щоб врахувати специфікації кодування UTF-8.

Перевіряється з'єднання. Якщо з'єднання не вдалося, програма виводить повідомлення про помилку підключення та припиняє виконання за допомогою функції die().

Розрахунок амортизації може здійснюватися за допомогою двух способів:

1) лінійний. Формула розрахунку амортизації за допомогою лінійного способу:

$$A_{\text{поч}} = \frac{\text{В}_\text{т}}{\text{Т}_\text{експл}}$$

де $A_{\text{поч}}$ – початкове значення амортизації (вважатимемо, що воно дорівнює нулю);

$\text{В}_\text{т}$ – вартість транспортного засобу;

$\text{Т}_\text{експл}$ – термін експлуатації ТЗ (у роках).

2) методом подвійного зменшення. Формулу розрахунку амортизації за допомогою методу подвійного зменшення:

$$A_{\text{поч}} = \left(\frac{\text{Т}_{\text{експл}}}{2}\right) * \text{В}_{\text{т}}$$

де $A_{\text{поч}}$ – початкове значення амортизації (вважатимемо, що воно дорівнює нулю);

$\text{В}_{\text{т}}$ – вартість транспортного засобу;

$\text{Т}_{\text{експл}}$ – термін експлуатації ТЗ (у роках).

3.3. Тестування розроблюваної системи

Головне вікно розроблюваної системи наведено на рис. 3.15. Воно поділяється на авторизацію та реєстрацію. Також відображається інформацію про транспортні засоби, які є у базі даних за допомогою слайдера.

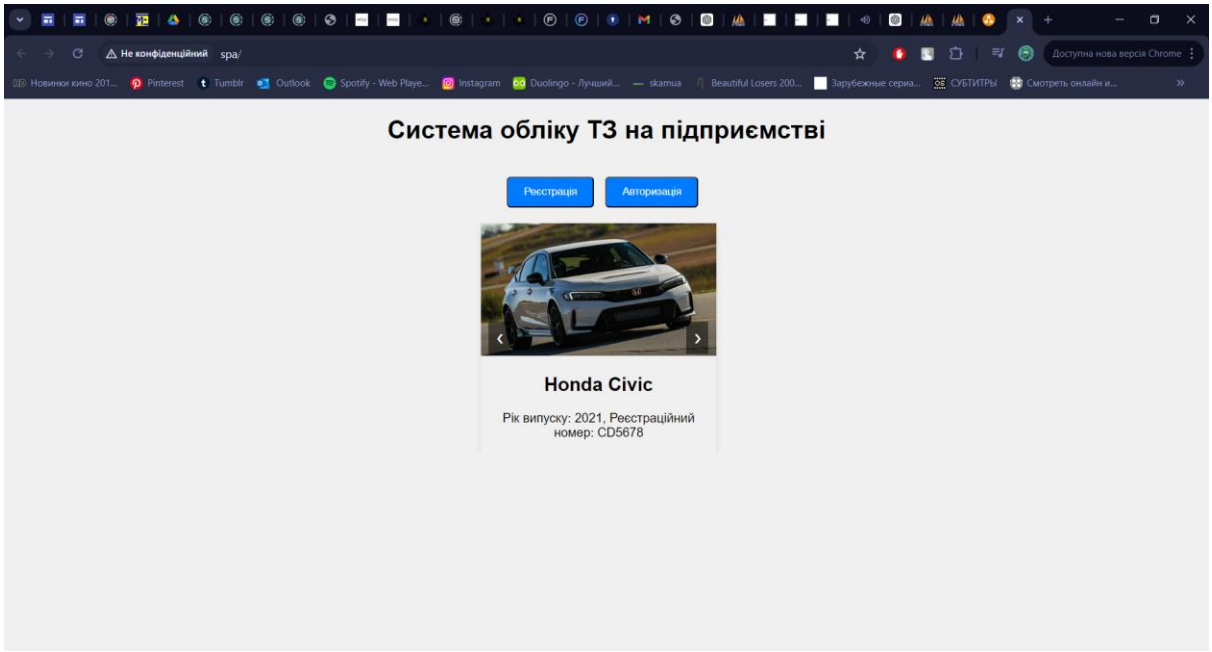

Рис.3.15 – Головне вікно розроблюваної системи

Відкриємо веб-сторінку «Реєстрація» (рис.3.16) та створимо нового користувача (рис.3.17). За замовчуванням всі реєструються з правами звичайного користувача.

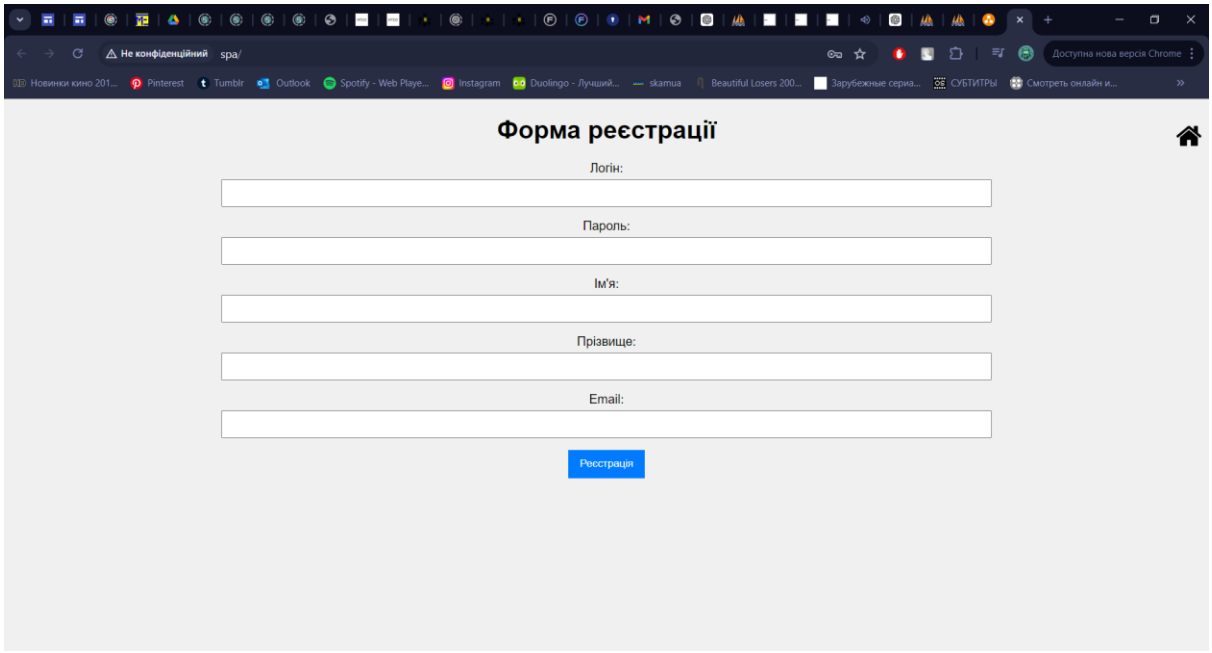

Рис.3.16 – Веб-сторінка «Реєстрація»

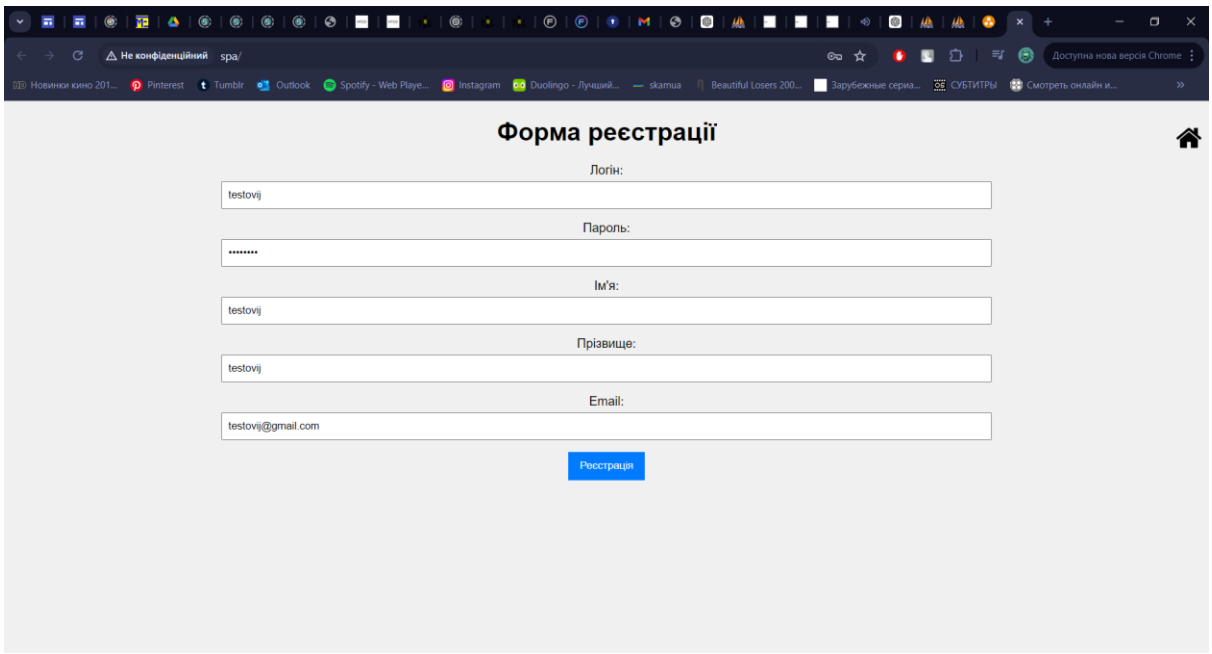

Рис.3.17 – Створення нового користувача

Зайдемо у щойно створений акаунт (рис. 3.18).

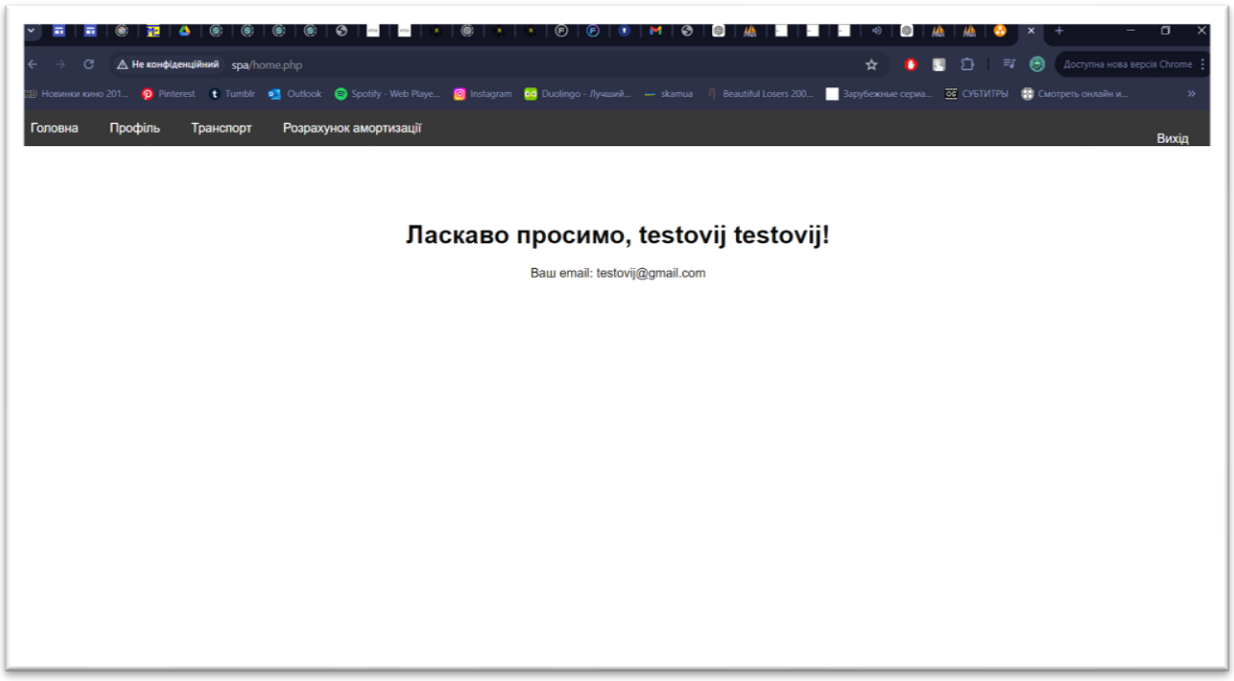

Рис. 3.18 – Авторизація у щойно створеному акаунті

У особистому кабінеті користувача ми можемо переглянути інформацію про свій профіль (рис.3.23), перейти на вкладку транспорт (рис.3.19), з цієї вкладки перейти на перегляд поточного транспорту (рис. 3.20) та переглянути доступний розрахунок амортизації (рис.3.21), але порахувати її він не може.

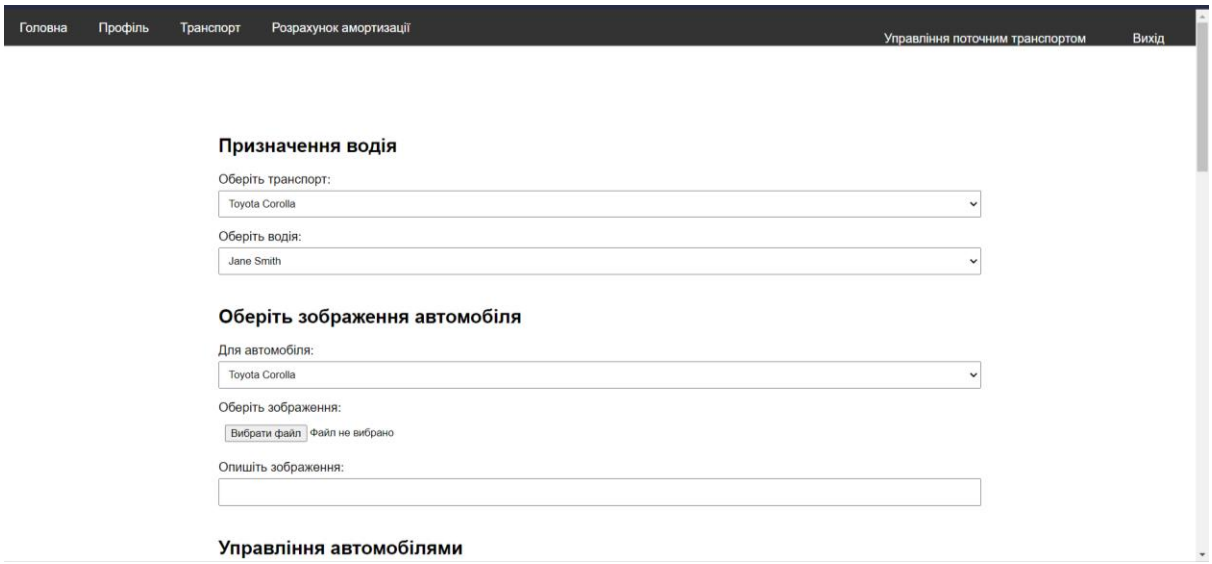

Рис. 3.19 – Вкладка транспорт

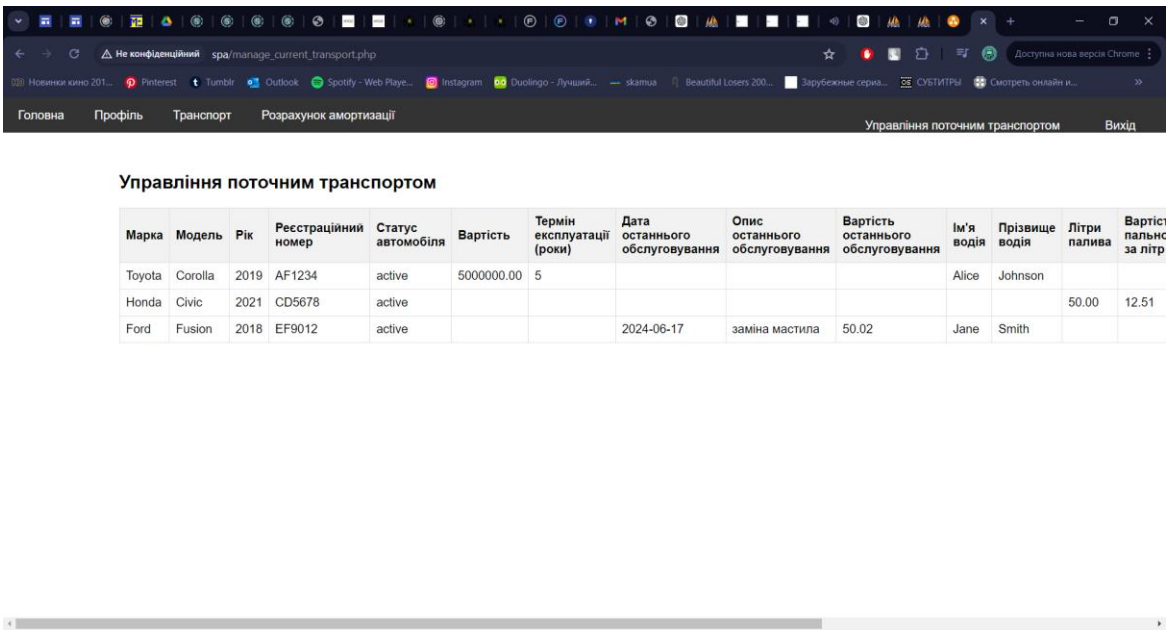

Рис. 3.20 – Управління поточним транспортом

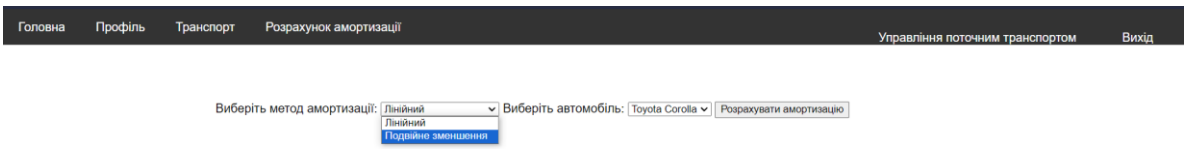

Рис. 3.21 – Розрахунок амортизації

У разі спроби здійснення розрахунку амортизації, як користувач, то буде проінформовано, що доступ заборонено (рис. 3.22).

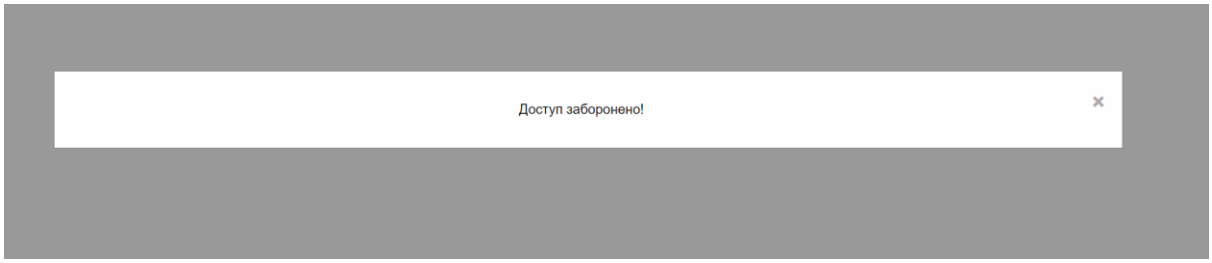

Рис. 3.22 – Повідомлення про заборонений доступ

На рис. 3.23 можна відредагувати інформацію про свій профіль. Змінити можна все, крім логіну.

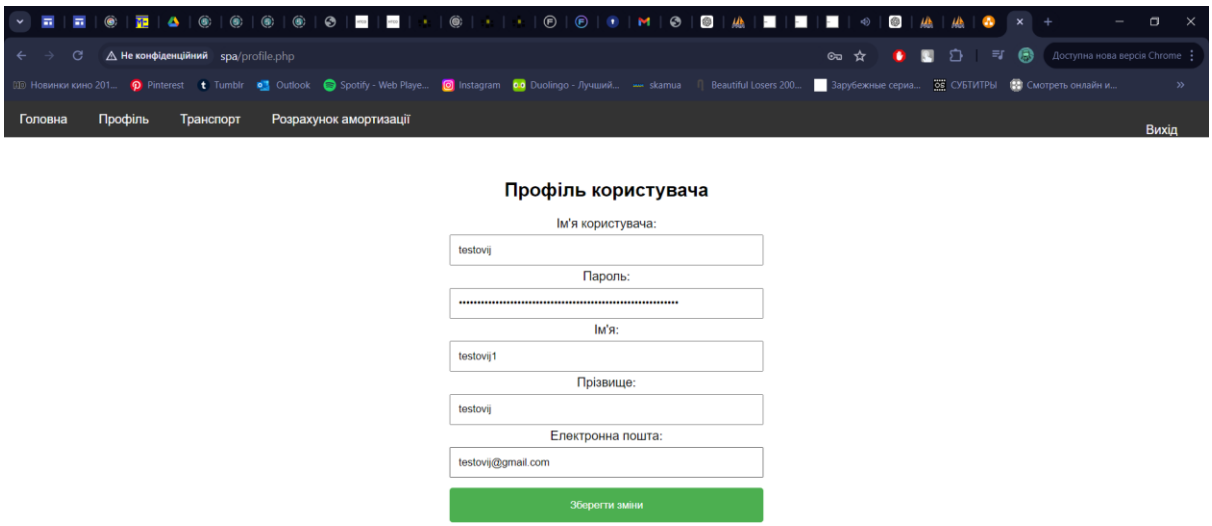

Рис. 3.23 – Профіль користувача

Авторизуємося як адміністратор та відразу перейдемо до вкладки «Транспорт» (рис.3.24). Тут можна призначати водіїв, з урахуванням на те, що водії вже занесені у базу даних, змінювати фотографії транспортному засобу, створювати новий транспортний засіб, фіксувати його обслуговування, фіксувати останню заправку, фіксувати страховку.

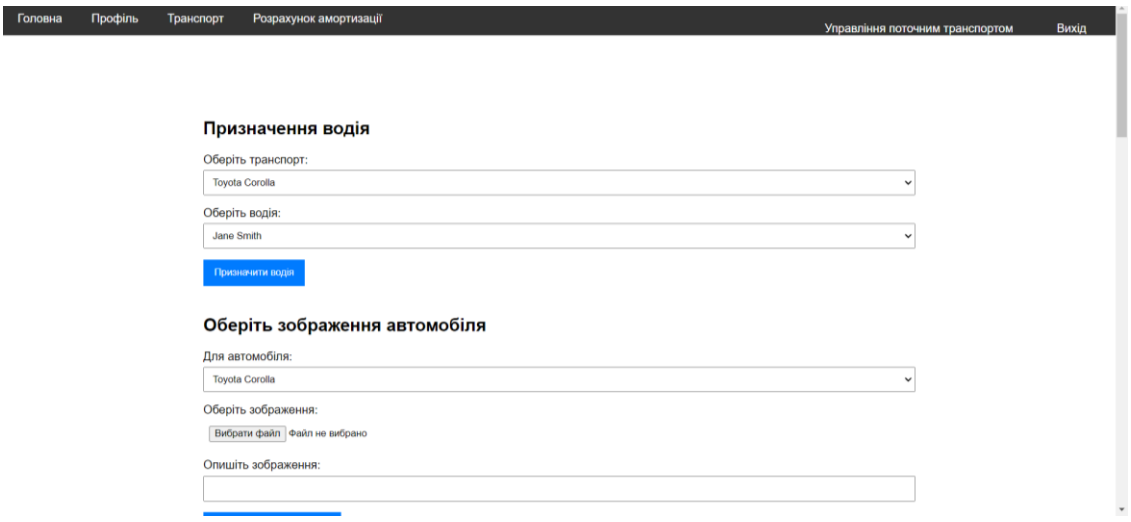

Рис.3.24 – Вкладка «Транспорт»

Наприклад, назначимо для транспортного засобу Honda Civic водія Jane Smith та натиснемо на кнопку «Призначити водія». У разі успішного призначення користувача буде проінформовано (рис. 3.25).

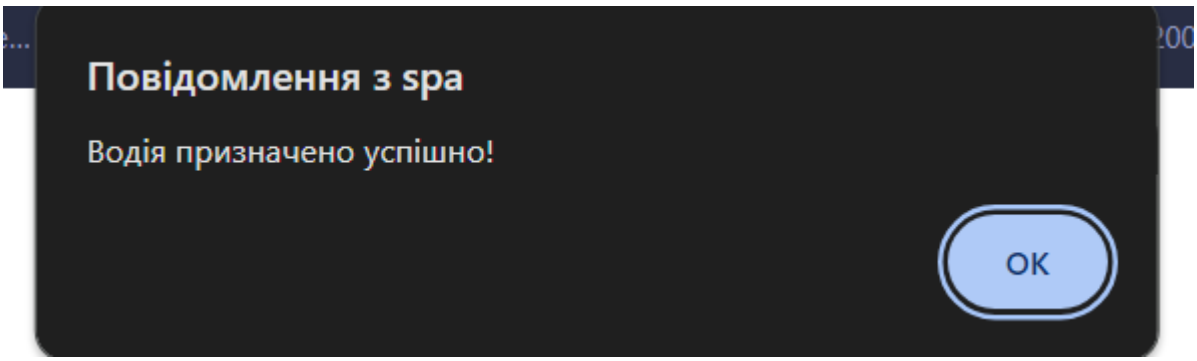

Рис. 3.25 – Повідомлення про успіх

Додамо новий транспортний засіб (рис.3.26).

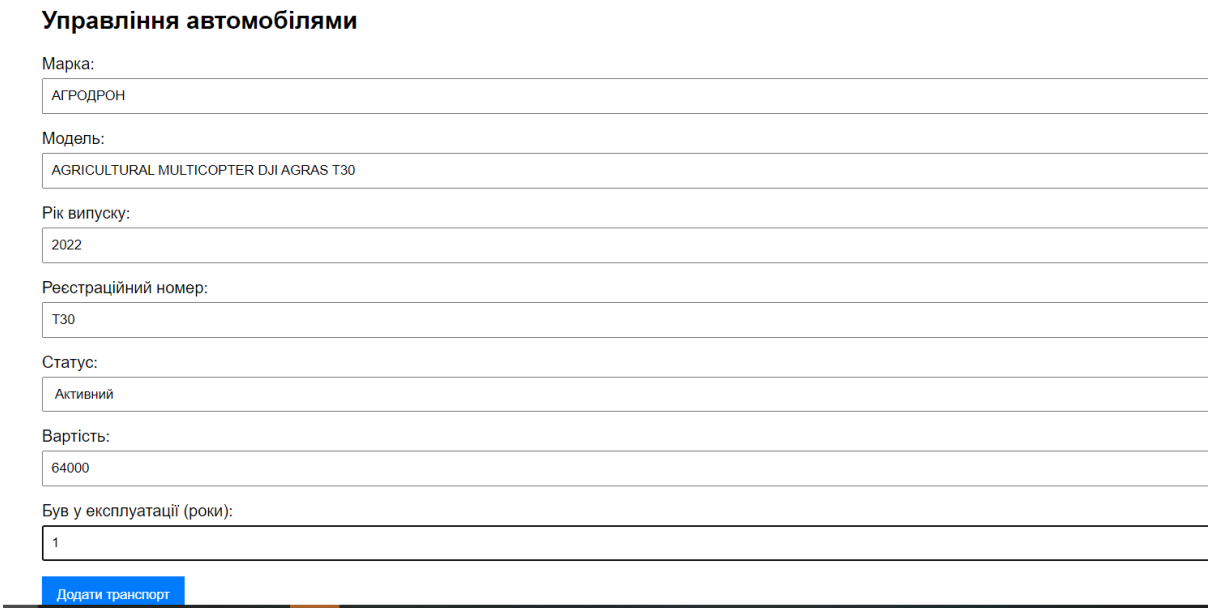

Рис. 3.26 – Додавання нового транспортного засобу

Після всіх маніпуляцій з додавання даних зрештою новий транспортний засіб відобразиться на головній сторінці (рис. 3.27).

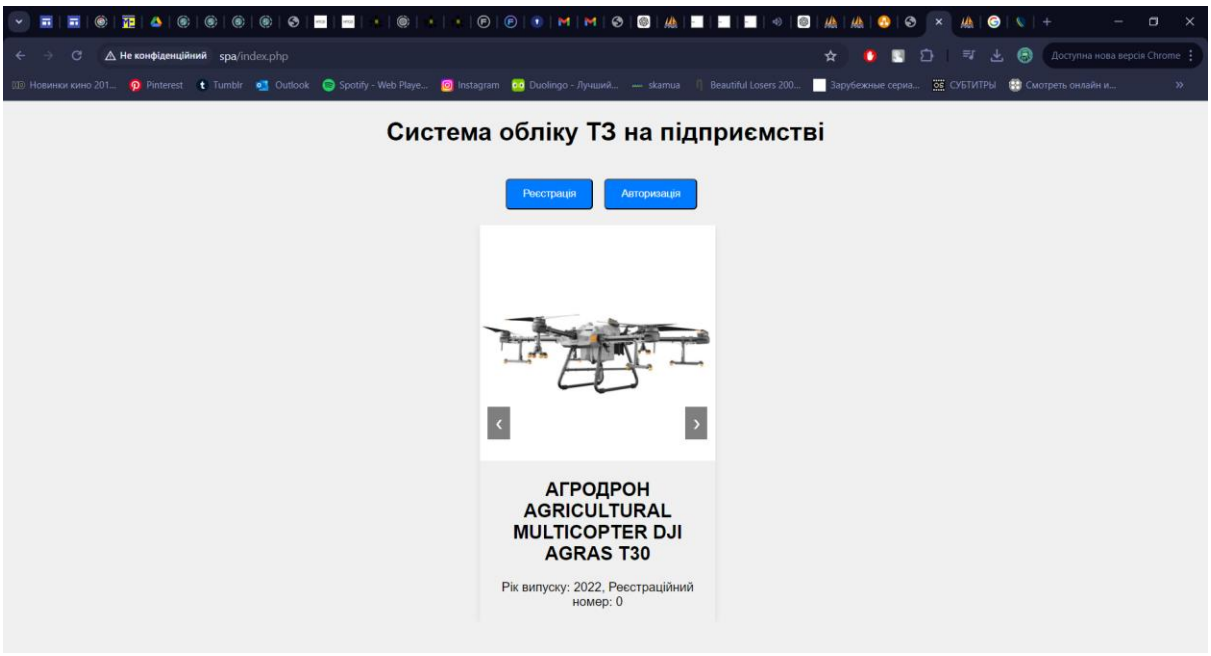

Рис. 3.27 – Інформація про новий транспортний засіб

Для інших транспортних засобів інформація додається аналогічно.

Перейдемо на сторінку «Управління поточним транспортом» (рис. 3.28). На цій сторінці в кінці кожного рядка можна відредагувати значення, але варто врахувати, що порожні значення неможливо відредагувати або внести в них нову інформацію.

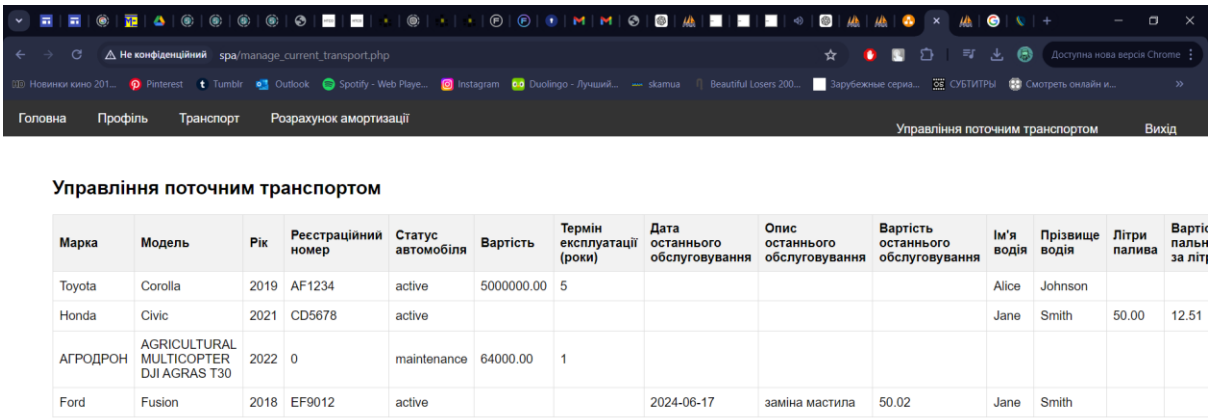

Рис. 3.28 – Управління поточним транспортом

Перейдемо на сторінку «Розрахунок амортизації» (рис.3.29). Оберемо метод розрахунку амортизації та транспортний засіб для якого бажаємо порахувати амортизацію та натиснемо на кнопку «Розрахувати амортизацію». Результати розрахунку автоматично занесуться у таблицю «Амортизація» в базі даних.

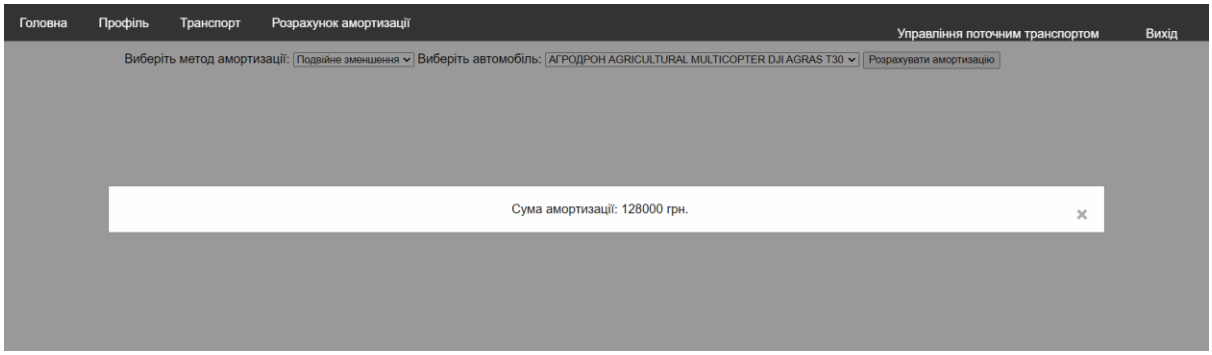

Рис. 3.29 – Сторінка «Амортизація»

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1. Надзвичайні ситуації, викликані пожежами, вибухами, техногенними та природніми причинами

Надзвичайні ситуації природного характеру — це небезпечні геологічні, метеорологічні, гідрологічні морські та прісноводні явища, деградація ґрунтів і надр, природні пожежі, зміни стану повітря, інфекційні захворювання людей та сільськогосподарських тварин, масове ураження сільськогосподарських рослин хворобами чи шкідниками, а також зміни стану водних ресурсів та біосфери.

Надзвичайні ситуації техногенного характеру включають транспортні аварії (катастрофи), пожежі, неспровоковані вибухи чи їх загрозу, аварії з викидом (або загрозою викиду) небезпечних хімічних, радіоактивних та біологічних речовин, раптове руйнування споруд і будівель, аварії на інженерних мережах та об'єктах життєзабезпечення, а також гідродинамічні аварії на греблях і дамбах.

До надзвичайних ситуацій техногенного характеру (код 10000) належать такі групи (в дужках вказано код групи):

- транспортні аварії (катастрофи — 10100);
- пожежі, вибухи (10200);
- аварії з викидом (або загрозою викиду) сильнодіючих отруйних речовин (СДОР) на об'єктах економіки (10300);
- наявність у навколишньому середовищі шкідливих речовин понад гранично допустимі концентрації (10400);
- аварії з викидом (або загрозою викиду) радіоактивних речовин (10500);
- раптове руйнування споруд (10600);
- аварії на електроенергетичних системах (10700);
- аварії на системах життєзабезпечення (10800);
- аварії систем зв'язку та телекомунікацій (10900);
- аварії на очисних спорудах (11000);
- гідродинамічні аварії (11100).

Надзвичайні ситуації природного характеру (код 20000) включають такі групи:

- геологічні (20100);
- метеорологічні (20200);
- гідрологічні морські та гідрологічні прісноводні (20300 та 20400);
- пожежі в природних екосистемах (20500);
- інфекційна захворюваність людей (20600);
- отруєння людей (20700);
- інфекційні захворювання сільськогосподарських тварин (20800);
- масова загибель диких тварин (20900);
- ураження сільськогосподарських рослин хворобами та шкідниками (20950) [7].

Офіційні джерела визначають аварії, стихійні лиха та катастрофи як основні причини надзвичайних ситуацій, що підтверджується визначенням терміна «надзвичайна ситуація» у законах, підзаконних актах та державних стандартах України. Згідно з ними, надзвичайна ситуація (НС) — це порушення нормальних умов життя і діяльності людей на об’єктах або територіях, викликане аварією, катастрофою, епідемією, стихійним лихом, епізоотією, епіфітотією, великою пожежею чи застосуванням засобів ураження, що призводить або може призвести до людських і матеріальних втрат, а також до забруднення навколишнього середовища.

Аварія — це небезпечна подія техногенного характеру, яка створює загрозу для життя і здоров’я людей на об'єкті, території або акваторії, спричиняє руйнування будівель, споруд, обладнання і транспортних засобів, порушення виробничого або транспортного процесу та завдає шкоди довкіллю.

Катастрофа — це масштабна аварія або інша подія, що має тяжкі, трагічні наслідки.

Основним критерієм для визначення будь-якої події як надзвичайної ситуації є встановлені державою кількісні показники наслідків та матеріальних і технічних

ресурсів, необхідних для їх ліквідації, залежно від реальних соціально-економічних умов.

Пожежі - це неконтрольоване горіння, яке розповсюджується в часі та просторі і спричиняє матеріальні збитки, загрозу життю та здоров'ю людей, тварин і рослин, а також руйнування навколишнього середовища. Пожежі можуть виникати як у природних умовах, так і внаслідок діяльності людини.

Пожежі поділяють на лісові, польові, побутові, виробничі, транспортні.

Пожежі виникають через: людський фактор, технічні причини, природні фактори, підпали.

До наслідків пожеж відносять: матеріальні збитки, загроза життю та здоров’ю, екологічні наслідки, соціальні наслідки.

В Україні існує кілька нормативно-правових актів, які регулюють питання надзвичайних ситуацій, що викликані пожежами.

Серед них є:

1) законодавчі акти:

- Кодекс цивільного захисту України – основний закон, який регулює питання цивільного захисту, включаючи запобігання пожежам, організацію пожежної охорони, евакуацію населення та ліквідацію наслідків надзвичайних ситуацій.

- Закон України «Про пожежну безпеку» - визначає правові, соціальні та економічні основи забезпечення пожежної безпеки на об'єктах всіх форм власності, а також обов'язки громадян, підприємств, установ та організацій щодо дотримання вимог пожежної безпеки.

- Закон України "Про правові засади цивільного захисту" – регулює діяльність органів влади, підприємств та організацій щодо захисту населення та територій від надзвичайних ситуацій техногенного та природного характеру, включаючи пожежі.

2) стандарти, які поділяють на державні будівельні норми (ДБН), національні стандарти України (ДСТУ) та на правила пожежної безпеки в Україні (НАПБ).

Актуальним на 2024 рік є ДБН В.1.1-7:2016 Пожежна безпека об’єктів будівництва. Загальні вимоги [8].

Актуальним на 2024 рік є НАПБ А.01.001-2014 Правила пожежної безпеки в Україні [9].

4.2 Планування робіт з охорони праці

Планування заходів з охорони праці - це процес управління, що здійснюється для забезпечення безпечних та здорових умов праці для працівників. Положення, що визначає планування заходів з охорони праці було затверджено 21.02.2003 року [10].

Планування - це процес обґрунтування рішень та розподілу ресурсів (матеріальних, фінансових, людських, інформаційних, часових) для їх реалізації.

Плановані заходи повинні бути конкретними та мати визначені обсяги і джерела фінансування. Грошові та матеріальні ресурси, виділені для виконання заходів з охорони праці, заборонено використовувати для інших цілей.

Планування роботи з охорони праці поділяється на перспективне, поточне та оперативне.

Перспективне планування (на період 3, 5, 10 і більше років) включає найважливіші, трудомісткі та довгострокові заходи з охорони праці. Реалізація заходів перспективного плану повинна бути підтверджена розрахунками матеріально-технічного забезпечення та фінансових витрат. Основною формою перспективного плану з охорони праці є комплексний план покращення умов охорони праці на підприємстві.

Основні напрямки перспективного планування - це розробка комплексної програми поліпшення умов та безпеки праці і санітарно-оздоровчих заходів, що мають бути частиною програми економічного і соціального розвитку підприємства.

Керівництво повинно визначити та документально оформити комплексну програму поліпшення умов охорони праці, зокрема:

- підготовку конкретних заходів програми;

- визначення та придбання необхідних засобів управління виробничими процесами, устаткування, засобів індивідуального і колективного захисту працівників;

- інформування працівників про відповідність робочих місць встановленим вимогам охорони праці та навчання їх навичкам для забезпечення безпеки праці;

- удосконалення та актуалізацію методів управління охороною праці і засобів контролю;

- аналіз перспективних тенденцій в галузі охорони праці, включаючи оцінку можливостей перевищення сучасного технічного рівня забезпечення охорони праці;

- виявлення та контроль шкідливих і небезпечних виробничих чинників і робіт, які потребують проведення медичних оглядів.

Програму рекомендується розробляти на три роки з обговоренням її на зборах трудового колективу, профспілковому органі, комісії з питань охорони праці підприємства або іншому уповноваженому органі.

Метою програми є створення безпечних та здорових умов праці для кожного працівника, досягається шляхом вирішення наступних завдань:

- виявлення причин і чинників, що можуть призвести до погіршення умов праці;

- вибір пріоритетних напрямків для досягнення найкращого результату з мінімальними витратами;

- розробка і реалізація відповідних організаційних, технічних, санітарно-гігієнічних, лікувально-профілактичних, соціально-економічних заходів.

- Програма передбачає вирішення таких завдань:

- максимальне скорочення робочих місць, які не відповідають вимогам охорони праці;

- приведення обладнання, машин і механізмів у відповідність до стандартів;

- закриття виробничих об'єктів, що не гарантують безпеку праці;
- забезпечення необхідної кількості санітарно-побутових приміщень;
- значне скорочення важких фізичних робіт;
- зменшення чисельності працівників, зайнятих ручною працею;
- розвиток лікувально-профілактичних та оздоровчих установ.

Зміст перспективної комплексної програми покращення умов охорони праці включає:

1. Організаційні заходи.
2. Технічні заходи.
3. Заходи з забезпечення належних санітарно-побутових умов та лікувально-профілактичної роботи.
4. Соціально-економічні заходи.
5. Заходи, пов'язані з проведенням науково-дослідних робіт [11].

ВИСНОВКИ

У процесі виконання даної дипломної роботи у першому розділі було розглянуто аналіз предметної області, що включав в себе вивчення основних процесів і вимог, проведення аналізу інформаційного забезпечення предметної області, що дозволило визначити ключові джерела даних і методи їх обробки, розробку технічного завдання, яке чітко формулює цілі та завдання проєкту, вимоги до функціональності системи та її обмеження.

У другому розділі наведено проєктування розроблюваної системи. Було створено інфологічну модель предметної області, що відображає основні об’єкти та їх взаємозв’язки, виконано даталогічне проєктування, що забезпечує структурування даних для ефективного зберігання та обробки, розроблено логічну схему бази даних, з урахуванням нормалізації для уникнення надлишкових даних і забезпечення цілісності, використано об'єктно-орієнтоване моделювання із застосуванням CASE-засобів, що дозволило створити ефективну і гнучку архітектуру системи.

У третьому розділі наведено реалізацію системи, тобто, описано розроблюваний веб-застосунок, що забезпечує інтуїтивно зрозумілий інтерфейс користувача та функціональність відповідно до технічного завдання, створено базу даних, яка відповідає проектним вимогам і забезпечує надійне зберігання та швидкий доступ до даних, проведено тестування розробленої системи, що дозволило виявити і виправити помилки, забезпечити стабільність і надійність роботи.

У четвертому розділі наведено охорону праці у надзвичайних ситуаціях, таких як пожежі, вибухи, техногенні та природні причини, що дозволило сформувати заходи для мінімізації ризиків, а також розглянуто пункт проведення планування робіт з охорони праці, що включає розробку заходів

щодо забезпечення безпечних умов праці та підвищення рівня охорони праці на підприємстві.

В результаті проведеної роботи, було створено автоматизовану інформаційну систему, яка відповідає сучасним вимогам і стандартам, забезпечуючи ефективне управління даними та підтримку бізнес-процесів. Розроблена система сприяє підвищенню продуктивності, покращенню якості послуг та забезпеченню безпеки на підприємстві.

ПЕРЕЛІК ПОСИЛАНЬ

1. Автомобіль на підприємстві URL: http://www.visnuk.com.ua/uk/publication/100005740-avtomobil-na-pidpriyemstvi.

2. Пасічник В.В., Резінченко В.А. Організація баз даних і знань. – К. Видавнича група BHV, 2006 – 384с.

3. Основи баз даних:. І.О. Завадський, 2011. – 190с.

4. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. - Київ. – 2017. – 110 с.

5. Організація баз даних : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. – Одеса : Фенікс,2019. – 246 с.

6. Костенко О. Б. Організація баз даних та знань : конспект лекцій (для студентів денної та заочної форм навчання першого (бакалаврського) рівня вищої освіти за спеціальністю 126 – Інформаційні системи та технології) / О. Б. Костенко, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 92 с

7. Надзвичайні ситуації та їх класифікація. Реферат. URL: https://osvita.ua/vnz/reports/bjd/22895/

8. ДБН В.1.1-7:2016 Пожежна безпека об'єктів будівництва

9. НАПБ А.01.001-2014 Правила пожежної безпеки в Україні.

10. Про порядок планування та фінансування заходів з охорони праці. URL: https://zakon.rada.gov.ua/rada/show/v3577611-03#Text

11. Пожарова О. В. Охорона праці : навчальний посібник / О. В. Пожарова. - Одеса, 2022. - 86 с. URL: https://doi.org/10.32837/11300.18442

ДОДАТКИ

ДОДАТОК А ЛІСТИНГ БАЗИ ДАНИХ

```
  -- phpMyAdmin SQL Dump
-- version 5.2.0
-- https://www.phpmyadmin.net/
--
-- Хост: 127.0.0.1:3306
-- Час створення: Чрв 09 2024 р., 19:41
-- Версія сервера: 5.7.39
-- Версія PHP: 8.1.9

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- База даних: `tzpidp`
--

-- --------------------------------------------------------

--
-- Структура таблиці `Depreciation`
--

CREATE TABLE `Depreciation` (
  `DepreciationID` int(11) NOT NULL,
  `VehicleID` int(11) DEFAULT NULL,
  `DepreciationDate` date DEFAULT NULL,
  `DepreciationAmount` decimal(10,2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

```
-- --------------------------------------------------------

--
-- Структура таблиці `Drivers`
--

CREATE TABLE `Drivers` (
  `DriverID` int(11) NOT NULL,
  `FirstName` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `LastName` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `LicenseNumber` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `ContactInfo` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `Drivers`
--

INSERT INTO `Drivers` (`DriverID`, `FirstName`, `LastName`,
`LicenseNumber`, `ContactInfo`) VALUES
(1, 'Jane', 'Smith', 'S2345678', 'jane.smith@example.com'),
(2, 'Alice', 'Johnson', 'J3456789', 'alice.johnson@example.com');

-- --------------------------------------------------------

--
-- Структура таблиці `Fuel`
--

CREATE TABLE `Fuel` (
  `FuelID` int(11) NOT NULL,
  `VehicleID` int(11) DEFAULT NULL,
  `Date` date DEFAULT NULL,
```

```
  `Liters` decimal(10,2) DEFAULT NULL,
  `CostPerLiter` decimal(10,2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `Fuel`
--

INSERT INTO `Fuel` (`FuelID`, `VehicleID`, `Date`, `Liters`,
`CostPerLiter`) VALUES
(1, 2, '2024-06-09', '50.00', '12.51');

-- --------------------------------------------------------

--
-- Структура таблиці `Insurance`
--

CREATE TABLE `Insurance` (
  `InsuranceID` int(11) NOT NULL,
  `VehicleID` int(11) DEFAULT NULL,
  `PolicyNumber` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `StartDate` date DEFAULT NULL,
  `EndDate` date DEFAULT NULL,
  `Cost` decimal(10,2) DEFAULT NULL,
  `Company` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `Insurance`
--

INSERT INTO `Insurance` (`InsuranceID`, `VehicleID`,
`PolicyNumber`, `StartDate`, `EndDate`, `Cost`, `Company`) VALUES
(1, 1, 'dffdfdf', '2024-06-06', '2024-06-28', '0.13', 'fgfdfd');
```

```
-- --------------------------------------------------------

--
-- Структура таблиці `Maintenance`
--

CREATE TABLE `Maintenance` (
  `MaintenanceID` int(11) NOT NULL,
  `VehicleID` int(11) DEFAULT NULL,
  `Date` date DEFAULT NULL,
  `Description` text COLLATE utf8mb4_unicode_ci,
  `Cost` decimal(10,2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `Maintenance`
--

INSERT INTO `Maintenance` (`MaintenanceID`, `VehicleID`, `Date`,
`Description`, `Cost`) VALUES
(1, 3, '2024-06-17', 'заміна мастила', '50.02');

-- --------------------------------------------------------

--
-- Структура таблиці `Users`
--

CREATE TABLE `Users` (
  `UserID` int(11) NOT NULL,
  `Username` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `PasswordHash` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `FirstName` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `LastName` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
```

```
  `Email` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `CreatedAt` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `Role` enum('User','Admin') COLLATE utf8mb4_unicode_ci DEFAULT
'User'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `Users`
--

INSERT INTO `Users` (`UserID`, `Username`, `PasswordHash`,
`FirstName`, `LastName`, `Email`, `CreatedAt`, `Role`) VALUES
(1, 'test',
'$2y$10$jLltpjESz5LaqFAkDhuXM.5f3G.4rUhsJwPeOv3P7GkPDj0dnqTBu',
'test', 'test', 'test@gmail.com', '2024-06-01 17:13:53', 'Admin'),
(9, 'test11',
'$2y$10$XBL25dums.RuKD7BwU5GtelVb8RMrHtKdsXHdqunW1oTzUq.M1Dm2', 't',
't', 'test11@gmail.com', '2024-06-01 20:18:27', 'User'),
(10, 't1',
'$2y$10$CgI4hHtMCY8xNW1xzwReJOr38yxI.yR6za0M1PMQ/z/VbMCnMzXAK', 't',
't', 't@gmail.com', '2024-06-09 15:53:44', 'User');

-- --------------------------------------------------------

--
-- Структура таблиці `VehicleDrivers`
--

CREATE TABLE `VehicleDrivers` (
  `VehicleID` int(11) NOT NULL,
  `DriverID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `VehicleDrivers`
--
```

```
INSERT INTO `VehicleDrivers` (`VehicleID`, `DriverID`) VALUES
(3, 1),
(1, 2);

-- --------------------------------------------------------

--
-- Структура таблиці `VehicleImages`
--

CREATE TABLE `VehicleImages` (
  `ImageID` int(11) NOT NULL,
  `VehicleID` int(11) DEFAULT NULL,
  `ImagePath` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `Description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `VehicleImages`
--

INSERT INTO `VehicleImages` (`ImageID`, `VehicleID`, `ImagePath`,
`Description`) VALUES
(3, 2, 'img/honda.jpg', 'honda'),
(4, 1, 'img/toyota.png', 'toyota');

-- --------------------------------------------------------

--
-- Структура таблиці `Vehicles`
--

CREATE TABLE `Vehicles` (
  `VehicleID` int(11) NOT NULL,
```

```
  `Make` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `Model` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `Year` int(11) DEFAULT NULL,
  `RegistrationNumber` varchar(50) COLLATE utf8mb4_unicode_ci
DEFAULT NULL,
  `VehicleStatus`
enum('active','maintenance','in_road','decommissioned') COLLATE
utf8mb4_unicode_ci DEFAULT 'active',
  `InitialCost` decimal(10,2) DEFAULT NULL,
  `UsefulLife` int(11) DEFAULT NULL,
  `DepreciationMethod` enum('linear','double_declining') COLLATE
utf8mb4_unicode_ci DEFAULT 'linear'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

--
-- Дамп даних таблиці `Vehicles`
--

INSERT INTO `Vehicles` (`VehicleID`, `Make`, `Model`, `Year`,
`RegistrationNumber`, `VehicleStatus`, `InitialCost`, `UsefulLife`,
`DepreciationMethod`) VALUES
(1, 'Toyota', 'Corolla', 2019, 'AF1234', 'active', '5000000.00',
5, 'linear'),
(2, 'Honda', 'Civic', 2021, 'CD5678', 'active', NULL, NULL,
'linear'),
(3, 'Ford', 'Fusion', 2018, 'EF9012', 'active', NULL, NULL,
'linear');

--
-- Індекси збережених таблиць
--

--
-- Індекси таблиці `Depreciation`
--
ALTER TABLE `Depreciation`
  ADD PRIMARY KEY (`DepreciationID`),
```

```
  ADD KEY `VehicleID` (`VehicleID`);

--
-- Індекси таблиці `Drivers`
--
ALTER TABLE `Drivers`
  ADD PRIMARY KEY (`DriverID`);

--
-- Індекси таблиці `Fuel`
--
ALTER TABLE `Fuel`
  ADD PRIMARY KEY (`FuelID`),
  ADD KEY `VehicleID` (`VehicleID`);

--
-- Індекси таблиці `Insurance`
--
ALTER TABLE `Insurance`
  ADD PRIMARY KEY (`InsuranceID`),
  ADD KEY `VehicleID` (`VehicleID`);

--
-- Індекси таблиці `Maintenance`
--
ALTER TABLE `Maintenance`
  ADD PRIMARY KEY (`MaintenanceID`),
  ADD KEY `VehicleID` (`VehicleID`);

--
-- Індекси таблиці `Users`
--
ALTER TABLE `Users`
  ADD PRIMARY KEY (`UserID`),
  ADD UNIQUE KEY `Username` (`Username`),
  ADD UNIQUE KEY `Email` (`Email`);

--
```

```
-- Індекси таблиці `VehicleDrivers`
--
ALTER TABLE `VehicleDrivers`
  ADD PRIMARY KEY (`VehicleID`,`DriverID`),
  ADD KEY `DriverID` (`DriverID`);

--
-- Індекси таблиці `VehicleImages`
--
ALTER TABLE `VehicleImages`
  ADD PRIMARY KEY (`ImageID`),
  ADD KEY `VehicleID` (`VehicleID`);

--
-- Індекси таблиці `Vehicles`
--
ALTER TABLE `Vehicles`
  ADD PRIMARY KEY (`VehicleID`);

--
-- AUTO_INCREMENT для збережених таблиць
--

--
-- AUTO_INCREMENT для таблиці `Depreciation`
--
ALTER TABLE `Depreciation`
  MODIFY `DepreciationID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=22;

--
-- AUTO_INCREMENT для таблиці `Drivers`
--
ALTER TABLE `Drivers`
  MODIFY `DriverID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=3;

--
```

```
--
-- AUTO_INCREMENT для таблиці `Fuel`
--
ALTER TABLE `Fuel`
  MODIFY `FuelID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT для таблиці `Insurance`
--
ALTER TABLE `Insurance`
  MODIFY `InsuranceID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT для таблиці `Maintenance`
--
ALTER TABLE `Maintenance`
  MODIFY `MaintenanceID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT для таблиці `Users`
--
ALTER TABLE `Users`
  MODIFY `UserID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT для таблиці `VehicleImages`
--
ALTER TABLE `VehicleImages`
  MODIFY `ImageID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT для таблиці `Vehicles`
--
ALTER TABLE `Vehicles`
```

```
       MODIFY `VehicleID` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=5;

     --
     -- Обмеження зовнішнього ключа збережених таблиць
     --

     --
     -- Обмеження зовнішнього ключа таблиці `Depreciation`
     --
     ALTER TABLE `Depreciation`
       ADD CONSTRAINT `depreciation_ibfk_1` FOREIGN KEY (`VehicleID`)
REFERENCES `Vehicles` (`VehicleID`);

     --
     -- Обмеження зовнішнього ключа таблиці `Fuel`
     --
     ALTER TABLE `Fuel`
       ADD CONSTRAINT `fuel_ibfk_1` FOREIGN KEY (`VehicleID`)
REFERENCES `Vehicles` (`VehicleID`);

     --
     -- Обмеження зовнішнього ключа таблиці `Insurance`
     --
     ALTER TABLE `Insurance`
       ADD CONSTRAINT `insurance_ibfk_1` FOREIGN KEY (`VehicleID`)
REFERENCES `Vehicles` (`VehicleID`);

     --
     -- Обмеження зовнішнього ключа таблиці `Maintenance`
     --
     ALTER TABLE `Maintenance`
       ADD CONSTRAINT `maintenance_ibfk_1` FOREIGN KEY (`VehicleID`)
REFERENCES `Vehicles` (`VehicleID`);

     --
     -- Обмеження зовнішнього ключа таблиці `VehicleDrivers`
     --
```

```
      ALTER TABLE `VehicleDrivers`
        ADD CONSTRAINT `vehicledrivers_ibfk_1` FOREIGN KEY
(`VehicleID`) REFERENCES `Vehicles` (`VehicleID`),
        ADD CONSTRAINT `vehicledrivers_ibfk_2` FOREIGN KEY (`DriverID`)
REFERENCES `Drivers` (`DriverID`);

      --
      -- Обмеження зовнішнього ключа таблиці `VehicleImages`
      --
      ALTER TABLE `VehicleImages`
        ADD CONSTRAINT `vehicleimages_ibfk_1` FOREIGN KEY (`VehicleID`)
REFERENCES `Vehicles` (`VehicleID`);
      COMMIT;

      /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
      /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
      /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

ДОДАТОК Б. ЛІСТИНГ ВЕБ-ЗАСТОСУНКУ

calculate_amortization.php

```php
<?php
// Перевірка, чи були передані необхідні дані через POST-запит
if (isset($_POST['method'], $_POST['vehicle_id'])) {
    // Підключення до бази даних
    require_once 'database.php';
    connectDB();

    // Отримання даних з форми
    $method = $_POST['method'];
    $vehicle_id = $_POST['vehicle_id'];

    // Отримання даних про вибраний транспортний засіб з бази даних
    $sql_vehicle = "SELECT InitialCost, UsefulLife FROM Vehicles WHERE VehicleID
= ?";
    $stmt_vehicle = $conn->prepare($sql_vehicle);
    $stmt_vehicle->bind_param("i", $vehicle_id);
    $stmt_vehicle->execute();
    $result_vehicle = $stmt_vehicle->get_result();

    if ($result_vehicle->num_rows > 0) {
        $row_vehicle = $result_vehicle->fetch_assoc();
        $initial_cost = $row_vehicle['InitialCost'];
        $useful_life = $row_vehicle['UsefulLife'];

        // Розрахунок амортизації в залежності від вибраного методу
        $depreciation_rate = 0;
        switch ($method) {
            case 'linear':
                if ($useful_life != 0) {
                    $depreciation_rate = $initial_cost / $useful_life;
                }
                break;
            case 'double_declining':
                // Розрахунок амортизації для методу подвійного зменшення
                $depreciation_rate = (2 / $useful_life) * $initial_cost;
                break;
            default:
                $depreciation_rate = 0;
        }

        // Вставка результату розрахунку амортизації в таблицю Depreciation
        $sql_insert = "INSERT INTO Depreciation (VehicleID, DepreciationDate,
DepreciationAmount) VALUES (?, NOW(), ?)";
        $stmt_insert = $conn->prepare($sql_insert);
        $stmt_insert->bind_param("id", $vehicle_id, $depreciation_rate);
        $stmt_insert->execute();
```

```
        if ($stmt_insert->affected_rows > 0) {
            echo "Сума амортизації: $depreciation_rate грн.";
        } else {
            echo "Помилка: Неможливо вставити результат амортизації в базу
даних.";
        }

        // Закриття підготовлених запитів та з'єднання з базою даних
        $stmt_insert->close();
    } else {
        echo "Помилка: Автомобіль з таким ID не знайдено.";
    }

    $stmt_vehicle->close();
    $conn->close();
} else {
    echo "Помилка: Недостатньо даних для розрахунку амортизації.";
}
?>
```

amortization.php

```
<?php
// Перевірка, чи були передані необхідні дані через POST-запит
if (isset($_POST['method'], $_POST['vehicle_id'])) {
    // Підключення до бази даних
    require_once 'database.php';
    connectDB();

    // Отримання даних з форми
    $method = $_POST['method'];
    $vehicle_id = $_POST['vehicle_id'];

    // Отримання даних про вибраний транспортний засіб з бази даних
    $sql_vehicle = "SELECT InitialCost, UsefulLife FROM Vehicles WHERE VehicleID
= ?";
    $stmt_vehicle = $conn->prepare($sql_vehicle);
    $stmt_vehicle->bind_param("i", $vehicle_id);
    $stmt_vehicle->execute();
    $result_vehicle = $stmt_vehicle->get_result();

    if ($result_vehicle->num_rows > 0) {
        $row_vehicle = $result_vehicle->fetch_assoc();
        $initial_cost = $row_vehicle['InitialCost'];
        $useful_life = $row_vehicle['UsefulLife'];

        // Розрахунок амортизації в залежності від вибраного методу
```

```
        $depreciation_rate = 0;
        switch ($method) {
            case 'linear':
                if ($useful_life != 0) {
                    $depreciation_rate = $initial_cost / $useful_life;
                }
                break;
            case 'double_declining':
                // Розрахунок амортизації для методу подвійного зменшення
                $depreciation_rate = (2 / $useful_life) * $initial_cost;
                break;
            default:
                $depreciation_rate = 0;
        }

        // Вставка результату розрахунку амортизації в таблицю Depreciation
        $sql_insert = "INSERT INTO Depreciation (VehicleID, DepreciationDate,
DepreciationAmount) VALUES (?, NOW(), ?)";
        $stmt_insert = $conn->prepare($sql_insert);
        $stmt_insert->bind_param("id", $vehicle_id, $depreciation_rate);
        $stmt_insert->execute();

        if ($stmt_insert->affected_rows > 0) {
            echo "Сума амортизації: $depreciation_rate грн.";
        } else {
            echo "Помилка: Неможливо вставити результат амортизації в базу
даних.";
        }

        // Закриття підготовлених запитів та з'єднання з базою даних
        $stmt_insert->close();
    } else {
        echo "Помилка: Автомобіль з таким ID не знайдено.";
    }

    $stmt_vehicle->close();
    $conn->close();
} else {
    echo "Помилка: Недостатньо даних для розрахунку амортизації.";
}
?>
```

ДОДАТОК В. ТЕХНІЧНЕ ЗАВДАННЯ

1. Вступ

1.1. Призначення

1.2. Область дії

1.3. Визначення, акроніми і скорочення

1.4 Короткий огляд

2. Повний опис

2.1. Перспектива виробу

2.1.1. Інтерфейси користувача.

2.1.2. Інтерфейси зв’язку

2.1.3. Обмеження пам’яті

2.2. Характеристики користувача

2.3. Список варіантів використання

2.4. Обмеження

2.5. Допущення і залежності

3. Специфічні вимоги

3.1. Вимоги до зовнішніх інтерфейсів

3.1.1. Інтерфейси користувача

3.1.2. Інтерфейси зв’язку

3.2. Функціональні вимоги

3.2.1. Режим редагування

3.2.2. Режим Перегляду

3.3. Вимоги до робочих характеристик

3.4. Проектні обмеження

3.5. Атрибути системи програмного забезпечення

3.5.1. Надійність

3.5.2. Захист

3.5.3. Зручність супроводу

3.5.4. Мобільність

1. Вступ

1.1. Призначення

Система обліку ТЗ на підприємстві призначена для збереження та структуризації інформації про транспортні засоби на підприємстві.

1.2. Область дії

Система називається «Система обліку ТЗ на підприємстві».

Дана система буде надавати інформацію про:

- транспортний засіб: модель, марка, рік випуску, реєстраційний номер, статус, вартість, термін експлуатації (роки) та способ амортизації;
- амортизацію: код транспортного засобу, дату підрахунку амортизації та суму амортизації;
- водія: ім’я, прізвище, номер водійського посвідчення, контактну інформацію;
- паливо: код транспортного засобу, дату останньої заправки, літраж, ціна за літр;
- страховку: код транспортного засобу, страховий поліс, початок страховки, кінець страховки, вартість, компанію;
- обслуговування: код транспортного засобу, дату, опис, вартість;

- користувачів: логін, пароль, ім’я, прізвище, електронну пошту, дату створення, роль;
- водії-транспортний засобів: таблиця зв’язок між водіями та транспортними засобами;
- зображення транспортних засобів: код транспортного засобу, шлях до зображення, опис.

1.3. Визначення, акроніми і скорочення

ПЗ – Програмне забезпечення

БД – База даних

ТЗ – Транспортні засоби

1.4. Короткий огляд

Ця SRS (Software Requirements Specification) структурована таким чином, щоб систему та всі її функціональні можливості описували поетапно. Для цього кожний розділ містить опис функціональних можливостей системи, розпочинаючи з простих дій та функцій і поступово переходячи до складніших.

2. Повний опис

2.1. Перспектива виробу

Система обліку транспортних засобів на підприємстві. Ця система виконує операції у сфері обліку ТЗ, система є самостійною і не належить до жодних вже існуючих систем.

2.1.1. Інтерфейси користувача

Людський фактор відіграє важливу роль у взаємодії між системами. Саме тому в системі передбачено користувацький інтерфейс взаємодії. Щоб користувач міг ефективно працювати з цим інтерфейсом, його апаратне забезпечення повинне відповідати таким вимогам: мінімальна роздільна здатність екрана 1024*768 та наявність підключення до бази даних системи.

Після 30-хвилинного інструктажу користувач може використовувати систему повністю і виконувати всі її функції.

2.1.2. Інтерфейси зв’язку

Для роботи з phpMyAdmin необхідне інтернет підключення.

2.1.3. Обмеження пам’яті

Початковий пакет файлів системи обліку ТЗ на підприємстві займає 1,15 МБ, для бази даних слід забезпечити 2 МБ дискового простору. Щоб система ефективно і швидко функціонувала потрібно забезпечити як мінімум 256 МБ оперативної пам’яті. Система для своєї роботи використовує 15 Мб оперативної пам’яті. За умови якщо вона не виконує громіздких операції.

2.2. Характеристика користувача

Адміністратор – людина, що має найвищий рівень доступу до інформації, може вносити нову інформацію та редагувати існуючу.

База даних – відіграє основну роль в АІС. Оскільки забезпечує обробку вхідної інформації, виступає як сховище даних.

Користувач – людина, що може лише переглядати дані.

2.3. Список варіантів використання

Код	Виконавець	Операція	Опис
А1	Адміністратор	Внесення даних	Внесення нових даних у БД
А1	Адміністратор	Редагування даних	Доповнення/Редагування даних у БД
К1	Користувач	Перегляд даних	Переглядає дані різного роду у БД
БД1	База даних	Надання інформації	Надає дані різного роду у БД
БД1	База даних	Оновлення даних	Оновлення інформації різного роду у БД
БД1	База даних	Збереження інформації	Зберігає внесену інформацію про ТЗ для

			подальшого її використання

2.4. Обмеження

Обмеження для даної система відсутні

2.5. Допущення і залежності

Робота даної система залежить від:

- версії операційної системи;
- встановлених програм;
- режиму роботи операційної системи;
- кількість оперативної пам’яті;
- об’єм вінчестера;
- тактова частота процесора;
- встановлений Open Server чи ні.

3. Специфічні вимоги

3.1. Вимоги до зовнішніх інтерфейсів

3.1.1. Інтерфейси користувача

Мінімальна роздільна здатність екрану повинна бути не менше 1024*768, і частота оновлення зображення повинна бути 60 Гц або вище. Цей екран призначений для відображення інформації. Вікно повинно бути центроване на екрані та займати щонайменше 80% його площі. Оформлення вікна має бути максимально простим.

3.1.2. Апаратні інтерфейси

- монітор з підключення до VGA адаптера;
- VGA адаптер об’ємом 128 Мб і вище;
- клавіатура 102 клавіш підключена за допомогою інтерфейсу PS/2 або USB;
- маніпулятор типу «миша» підключений за допомогою інтерфейсу PS/2 або USB;

- HDD накопичувач об’ємом 30 Гб і вище;

- оперативна пам'ять типу DDR або DDR2 об’ ємом 1 Гб і вище, з частотою шини більше 400 МГц;

- центральний процесор з тактовою частотою 1000 ГГц і вище, розрядність 32 біт, частота шини більше 300 МГц.

3.1.3. Інтерфейси програмного забезпечення

- Операційна система Microsoft Windows 10 і вище;

- Microsoft Net. Framework 6.0;

- Microsoft Office 2003 і вище;

- Microsoft Visual Studio 2005 Redistributable;

- Microsoft Visual Studio 2008 Redistributable;

- Microsoft Visual Studio 2010 Redistributable;

- Open Server 6.0 і вище.

3.1.4. Інтерфейси зв’язку

Для даної системи не потрібного жодних мережевих з’єднань

3.2. Функціональні вимоги

3.2.1. Режим редагування

Система повинна вносити в базу даних інформацію про ТЗ

3.2.2. Режим перегляду

Система повинна виводити основні відомості про ТЗ

3.3. Вимоги до робочих характеристик

Для даної система потрібно один термінал на якому одночасно можуть працювати лише одна особа. Система створена тільки для монопольного використання. Тобто в даний момент часу може працювати лише одна особа.

Виведення списку замовлень повинне відбуватись не більше 1 секунди. В один момент часу може відбуватись лише одна реєстрація замовлення або клієнта.

3.4. Проектні обмеження

Для даної системи відсутні обмеження, які би ускладнювали чи унеможливлювали роботу система.

3.5. Атрибути системи програмного забезпечення

3.5.1. Надійність

Для того аби система стабільно виконувала свої функції слід забезпечити апаратне забезпечення стабільною напругою, в операційній системі під час використання даної система не запускати паралельно з система інших програм у яких потреба у ресурсах перевищує наступні значення: оперативна пам'ять – 40 Мб, ресурси процесора – 4 %.

3.5.2. Захист

Для забезпечення максимального захисту слід дотримуватись наступних рекомендацій:

- Встановити антивірусну програму
- Встановити фаервол

3.5.3. Зручність супроводу

Супровід даної система полягає у тому аби оновлювати версію системи на робочому терміналі, а також вносити необхідні зміни в структуру бази даних не порушуючи її цілісності

3.5.4. Мобільність

Завдяки тому що база даних є зовнішньою, тобто підключається до система її можна перемістити на інший термінал і після встановлення система користуватись тими даними які були на попередньому терміналі.

3.5.5. Специфічні умови

3.5.5.1. Режим перегляду

Режим перегляду дозволяє перегляд всіх записів про ТЗ, пошук певного ТЗ.

3.5.5.2. Режим редагування

У режимі редагування відбувається внесення інформації в БД, тим самим виконавши додання ТЗ. Є можливість змінити ту чи іншу інформацію для кожного ТЗ.

4. Вимоги до програмної документації

4.1. Примірний склад програмної документації

Склад програмної документації повинен включати:

1. технічне завдання;

2. функціональний тест-план;

3. посібник користувача;

5. Техніко-економічні показники

5.1. Економічні переваги розробки

Орієнтовна економічна вартість проекту та оцінка його розміру розраховуються за методикою COCOMO.

6. Стадії і етапи розробки

6.1. Стадії розробки

Розробка має бути проведена відповідно до календарного плану роботи над ДП:

1. Аналітична робота;

2. Проектування АІС;

3. Програмна реалізація АІС;

4. Розробка документації користувача;

5. Розробка заходів з охорони праці при експлуатації АІС.

6.2. Етапи розробки

На стадії аналітичної роботи мають бути виконаний етапи:

- аналіз предметної області,

- дослідження наявних програмно-інформаційних комплексів вирішення завдань даної предметної області;

- розробка технічного завдання, що включає розробку, узгодження і затвердження даного технічного завдання.

На стадії проектування АІС мають бути виконані етапи проектування внутрішньої архітектури та візуального моделювання АІС.

На стадії програмної реалізації виконуються етапи:

- проектування інтерфейсу та узгодження з керівником (робочі форми програмного забезпечення з розташуванням основних робочих елементів),
- кодування,
- розробка бази даних.

На стадії розробки документації користувача створюється посібник користувача та (за потреби) довідник адміністратора.

На стадії розробки заходів з охорони праці при експлуатації АІС виконуються етапи:

- аналіз потенційних небезпек проектованого об’єкту (системи, процесу, пристрою, обладнання, послуг тощо) і можливостей негативного впливу його на оточуюче середовище та обслуговуючий персонал;
- розробка заходів щодо створення безпечних і нешкідливих умов праці;
- оптимізація впливу на оточуюче середовище і раціональне використання природних ресурсів.

7. Порядок контролю і приймання

7.1. Види випробувань (тестувань)

Тестування ПЗ супроводжується заповненням функціонального тест-плану в наступній послідовності:

- Тестування інтерфейсу та компонентів;
- Тестування бази даних.

7.2. Загальні вимоги до приймання роботи

На підставі результатів тестування Виконавець спільно із Замовником підписує Акт приймання-здачі програми в експлуатацію.

ДОДАТОК Г ДИСК З РОБОТОЮ