Пояснювальна записка
до курсової роботи

на тему: Веб-застосунок підтримки роботи СТО

КПІ.ІП-1313.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	–	База даних.
СТО	–	Станція технічного обслуговування
MVC	–	Model-View-Controller (Модель-Вигляд-Контролер)
HMVC	–	Hierarchical Model-View-Controller (Ієрархічні Модель-Вигляд-Контролер)
ООП	–	Об’єктно орієнтована парадигма програмування
IDE	–	Середа розробки програмного забезпечення
СКБД	–	Система керування БД
GUI	–	Графічний інтерфейс користувача (Graphical user interface)

ВСТУП

У сучасному індустріальному середовищі, де автотранспорт є необхідністю, питання збереження і оптимізації роботи автосервісних станцій (СТО) стає все більш актуальним. Зростання обсягів автопарку, технічний прогрес у галузі автомобільної техніки та підвищення вимог до якості обслуговування автовласників визначають потребу в ефективних та інноваційних рішеннях.

Сучасний стан об’єкта розробки, а саме автомобільний ремонт і обслуговування, вимагає впровадження нових підходів та технологій. Існуючі тенденції в світі підкреслюють важливість автоматизації та цифровізації процесів на СТО для підвищення ефективності роботи, зменшення часу обслуговування та уникнення помилок.

Наукові установи та організації вже зосереджують свою увагу на вирішенні цих проблем. Фахівці розробляють інноваційні підходи та технології, спрямовані на автоматизацію та покращення роботи автосервісів.

Мета даної роботи полягає в розробці веб-застосунку підтримки роботи на СТО, який сприятиме автоматизації рутинних процесів, оптимізації взаємодії з клієнтами та підвищенню якості послуг. Проект має на меті впровадження передових технологій у сферу автомобільного обслуговування, сприяючи покращенню якості та швидкості надання послуг, а також забезпеченню комфорту для кінцевого користувача.

Ця робота розгляне актуальні проблеми на автомобільних станціях технічного обслуговування, світові тенденції в їх розв’язанні, а також висвітлить можливі сфери застосування розробленого веб-застосунку, сприяючи подальшому розвитку автомобільної галузі та підвищенню конкурентоспроможності автосервісних підприємств.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

В сучасному автомобільному сервісі, автоматизація та використання програмного забезпечення стає ключовим фактором для оптимізації робочих процесів та підвищення якості послуг. Програмне забезпечення для обслуговування автотранспорту має вирішувати ряд завдань, спрямованих на підвищення ефективності та зручності роботи автосервісів. Для даної розробки було розроблено рушій для веб-застосунку, що спроможний в найкоротший термін обробити запит та віддати результат клієнту (до 200 мс).

1.2 Змістовний опис і аналіз предметної області

На сучасному етапі розвитку IT-технологій, автомобільні сервісні станції (СТО) активно впроваджують програмне забезпечення для оптимізації робочих процесів та покращення якості обслуговування. Проте, існуючий стан речей відзначається деякими недоліками та обмеженнями, які впливають на ефективність функціонування автосервісів.

На сьогоднішній день, багато СТО використовують різні програмні продукти для обліку робіт, управління запасами, та взаємодії з клієнтами. Однак, часто ці системи працюють в ізольованому режимі, не забезпечуючи повноцінної інтеграції всіх аспектів роботи сервісу. Це може викликати дублювання даних, помилки у введенні інформації, а також ускладнює аналіз ефективності діяльності.

Недоліками існуючих програм є також обмежені можливості використання аналітики та технологій штучного інтелекту для прогнозування та управління робочими процесами. Брак інтелектуального аналізу може призвести до непрогнозованих затримок у роботі, а відсутність автоматичного контролю може впливати на якість надання послуг.

У розробці веб-застосунку для підтримки роботи СТО важливо враховувати ці обмеження та недоліки. Метою роботи є покращення ситуації у

сфері IT для автомобільних сервісів. Вибір шляху розвитку спрямований на створення інтегрованої системи, яка об'єднає різні аспекти роботи СТО та забезпечить їх взаємодію в єдиному інформаційному просторі.

Досягнення цієї мети передбачає впровадження сучасних методів аналізу даних, використання технологій штучного інтелекту для покращення управління та прогнозування, а також створення зручного інтерфейсу для користувачів. Такий комплексний підхід сприятиме ефективній роботі автосервісних станцій, зменшенню помилок та підвищенню якості обслуговування автомобільних власників.

1.3 Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку підтримки роботи СТО. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Щодо аналізу технічних рішень, можу зазначити, що проаналізувавши декілька архітектурних патернів, а саме:

- Мікросервісна архітектура [29] – забезпечує гнучкість та модульність кінцевого продукту
- MVC архітектура [30] – відокремлює дані від інтерфейсу, чим забезпечує мінімальну кількість змін в моделі при зміні інтерфейсу та навпаки, аби зміни в моделі мінімізували зміни в інтерфейсі.
- HMVC архітектура [12] – розширення патерну MVC архітектури, що дозволяє вирішити проблему маштабованості класичного шаблону MVC архітектури.

З усіх розглянутих варіантів я обрав HMVC архітектуру, оскільки це дозволить в подальшому гарно маштабувати проект, роблячи кожен окремий модуль незалежним від інших. Також однією з головних переваг цього

архітектурного рішення стали простота розробки даного патерну та унеможливлення «заплутатись» в контролер, моделях та виглядах, а також швидкість розгортання, оскільки, як вже було зазначено вище, даний архітектурний патерн є ієрархічним.

Детальніше розглянувши дану архітектуру можна зазначити, що кожна тріада Модель-Вигляд-Контролер використовується як шар в ієрархічній моделі, надаючи доступ до інших таких шарів, проте залишаючи незалежність кожної тріади.

Також було розглянуто мови програмування, а саме:

- PHP [22] – скриптова мова програмування створена для генерації HTML [13] сторінок.
- C# [11] – ООП мова програмування зі строгою типізацією, виключаючи деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем.
- Java [14] – ООП мова програмування, випущена як основний компонент однойменної платформи Java.
- Python [25] – інтерпретована ООП мова програмування високого рівня із суворою динамічною типізацією.
- JavaScript [15] – динамічна ООП-прототипна мова програмування, що зачасту забезпечує керування браузером.
- Rust Lang [26] – мультипарадигмальна мова програмування загального призначення зі сильною типізацією, сфокусована на безпечній роботі з пам’яттю.

Зі всього переліку мов я зупинив свій вибір на PHP та JavaScript.

PHP є найпопулярнішою мовою програмування в категорії веб-застосунків, при тому лишаючись безкоштовним програмним забезпеченням відкритого типу. PHP інтерпретується вебсервером у HTML код, який передається клієнту. Перевагою даного способу є те, що користувач не бачить програмний код написаний на PHP, а може бачити тільки готовий результат.

Також PHP дозволяє генерувати JavaScript скрипти для більш динамічної взаємодії користувача із сторінкою.

JavaScript є скриптовою мовою програмування, що переважно використовується для написання сценаріїв дії вебсторінок. Має низку властивостей ООП, є безкоштовною та легкодоступною для вивчення.

Разом JavaScript та PHP дають сильну зв'язку для динамічної сторінки та зручної реалізації, з мінімальним порогом входження.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Для розробки зручно використовувати IDE, проаналізувавши доступні мені, я отримав наступний перелік:

- Visual Studio Code [28] – універсальний редактор початкового коду із можливість легкого встановлення безлічі розширень, для полегшення розробки.
- PHPStorm [23] – інтегроване середовище розробки для PHP зроблене в екосистемі JetBrains.
- Visual Studio 2022 [17] – програмне середовище розробки, зроблене з підтримкою екосистеми Microsoft.
- Notepad++ [21] – текстовий редактор, «розумна» версія програми Блокнот.

Із переліченого списку я зупинився на Visual Studio Code оскільки дана платформа не потребує серйозних ресурсів для запуску, зручна у використанні, має безліч розширень для зручної роботи, а також є безкоштовною.

В моєму проекті також потрібно застосовувати різні БД. Проаналізувавши ринок я відібрав наступний список:

- PostgreSQL [24] – СКБД, розробка якої не контролюється однією компанією, тобто мінімізує вплив інтересів компаній та забезпечує впровадження передових технологій.
- MySQL [18] – вільна СКБД, створена для підвищення швидкодії обробки великих баз даних

- SQLite [27] – полегшена СКБД, що втілена у вигляді бібліотеки у багатьох мов програмування
- MS SQL Server [16] – СКБД, що гарно підтримується в екосистемі Microsoft.

Серед даного переліку я вибрав PostgreSQL, оскільки вона є безкоштовною, включає в себе найновіші технології, та є дуже зручною для написання різних SQL запитів різної складності, оскільки має спрощений синтаксис. Також ця СКБД підтримує одночасну модифікацію БД декількома користувачами за рахунок вбудованих технологій. Також дана СКБД має великий функціонал написаний на мові програмування PL/pgSQL, що є вбудованою мовою програмування для цієї СКБД. Особливою перевагою є перспективність даної БД, оскільки її розробкою керують люди, а не великі компанії.

Для зручного управління та перегляду БД також потрібно вибрати зручну для роботи IDE для СКБД. Мною було розглянуто наступний список:

- DataGrip [3] – середовище для роботи з реляційними БД, повноцінно розкариваючись в екосистемі JEtBrains
- PgAdmin [5] – вбудоване GUI для роботи з postgreSQL
- Navicat [19] – програма керування базою даних та розробки програмного забезпечення

З усього переліченого я обрав DataGrip тому, що він є доступний для мене, достатньо зручний у використанні, підтримує мою БД, а саме PostgreSQL та відображає все, що стосується БД на достатньому рівні наглядності.

Із використаних бібліотек я використав вбудовані в PHP бібліотеки pgsql та PDO.

Також для спрощення написання видів, я використаю фраймворк. Із розглянутих мною варіантів я вибрав наступні:

- Bootstrap [10]
- UI Kit [9]
- Bulma [1]

Всі варіанти є досить непоганими фреймворками для роботи з дизайном сайту, проте мій вибір зупинився на найпопулярнішому з них – Bootstrap.

Проаналізувавши деякі можливості запуску серверу я зупинив свій вибір на nginx [20], оскільки даний сервіс є безкоштовним та не потребує особливих зусиль для розгортання в будь-якому режимі, а також дозволяє працювати з будь-якими базами даних та мовами програмування.

Також розглянувши різні рушії для веб-серверу я прийняв рішення, що жоден із знайдених мною мене не влаштовує, тому я прийняв рішення написати свій. Перевагами мого рушія є легкість в роботі, швидкість відпрацювання, безпечність та можливість детального налаштування, що є дуже важливим для створення проектів схожої конфігурації. Також перевагами мого рушія є легке розгортання та малий об’єм пам’яті для його зберігання, а саме 33 КБ, що сильно менше від його аналогів

1.3.3 Аналіз відомих програмних продуктів

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Веб-застосунок підтримки роботи СТО	RemOnline	Пояснення
Ведення бази даних клієнтів	Наявне, додаткова можливість прописувати знижки кожному клієнту окремо	Наявне	Введення клієнтської бази даних
Ведення бази даних автомобілів	Наявне	Наявне	Можливість додавати нові автомобілі при замовленні

Продовження таблиці 1.1

Ведення розкладу роботи для працівників	Наявне, можна вести детальний розклад для кожного робочого дня	Наявне	Можливість переглядати розклад кожному працівнику окремо
Ведення інформації про працівників	Наявне	Наявне	Можливість створювати та редагувати інформацію про працівників
Ведення послуг доступних для замовлення	Наявне	Наявне	Можливість створювати нове замовлення та редагувати вже створене
Підтримка роботи для мережі	Наявне	Наявне	Можливість створювати мережу СТО
Ведення записів для аналітики зайнятості працівників	Наявне	Наявне	Можливість призначати та вибирати яке завдання робить кожен працівник
Середа запуску	Локальний сервер	Хмара	Середа запуску програмного забезпечення

Продовження таблиці 1.1

Відстеження історії замовлень	Наявне	Наявне	Введення історії замовлень
Замовлення запчастин	Наявне	Відсутнє	Можливість замовити запчастини
Вести облік витрат та прибутку	Відсутнє	Наявне	Можливість ведення обліку витрат та прибутків

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є створення та відстеження замовлень в роботі СТО, більше функцій можна побачити на рисунку 1.3.

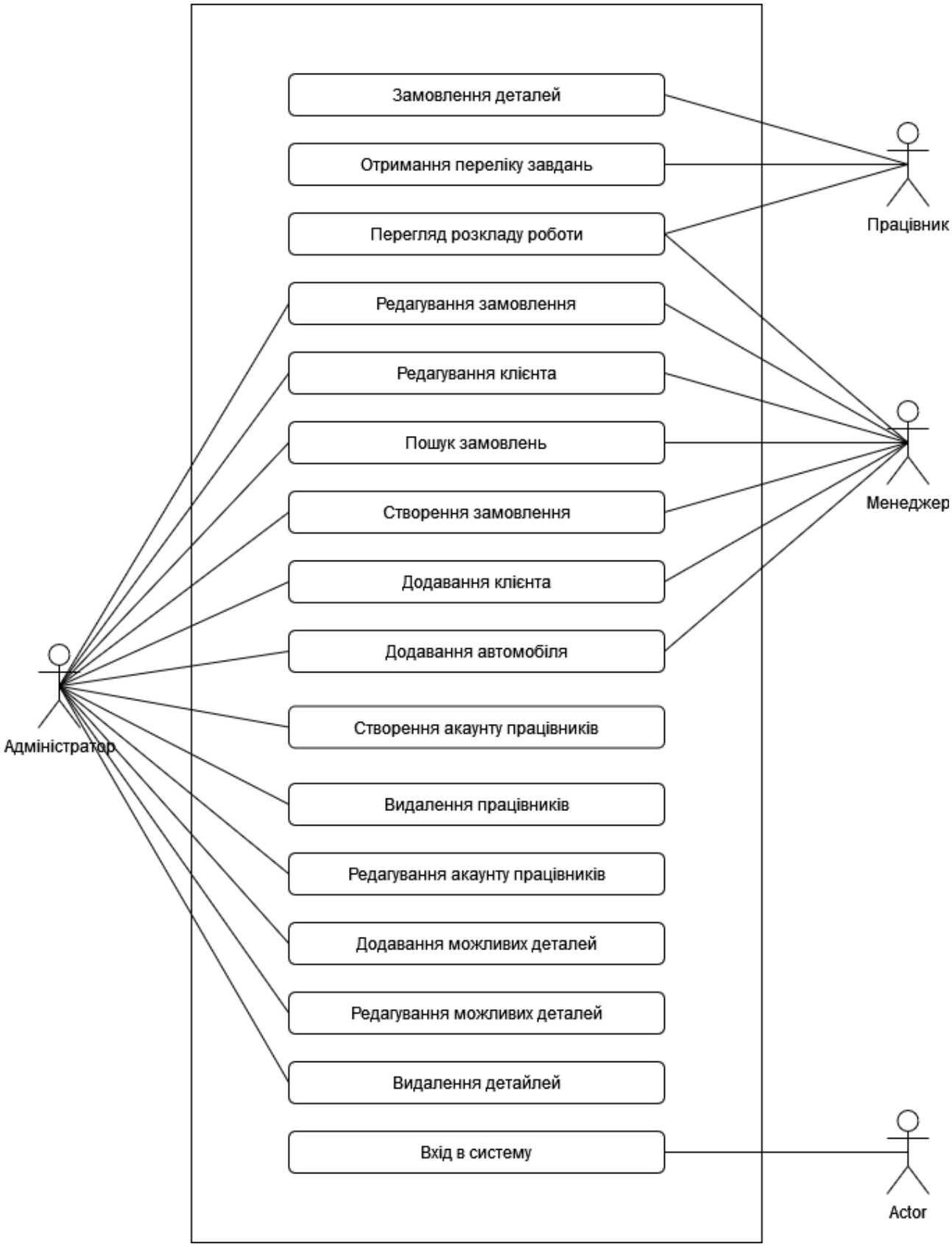

Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.2 - 1.13 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Вхід користувача
Use case ID	UC-01
Goals	Вхід зареєстрованого користувача до системи
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає увійти
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку входу, водить логін та пароль у системі, натискає кнопку «Увійти». Після натискання кнопка стає неактивною
Extension	У випадку введення некоретного логіна чи пароля висвічується повідомлення про помилку та розблоковується кнопка для повторного входу.
Post-Condition	Перехід на головну сторінку користувача

Таблиця 1.3 - Варіант використання UC-2

Use case name	Замовлення деталей
Use case ID	UC-02
Goals	Замовлення деталей працівником
Actors	Працівник
Trigger	Користувач бажає зробити замовлення деталей
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на сторінку замовлення деталей, в пошуку вводить назву деталі та нажимає додати. При натисканні на кнопку додати додається товар в корзину замовлень. В корзині є можливість редагувати кількість товару. Якщо кількість товару стає меншою за нуль – товар видаляється з корзини. Після формування корзини користувач натискає кнопку оформити замовлення, яка надсилає замовлення на сервер та очищує список замовлень
Extension	У випадку вводу некоректної назви товару через пошук товар не додається, корзине не очищується
Post-Condition	-

Таблиця 1.4 - Варіант використання UC-3

Use case name	Отримання переліку завдань
Use case ID	UC-03
Goals	Користувач хоче отримати перелік завдань та відмічати виконані завдання
Actors	Працівник
Trigger	Користувач бажає переглянути список всіх своїх завдань
Pre-conditions	Користувач зареєстрований у системі
Flow of Events	Користувач переходить на сторінку перегляду завдань, отримує від сервера список всіх своїх невиконаних завдань. Кожне завдання користувач має мати зможу відмітити виконаним. Якщо по замовленню завдання, яке було відмічене не залишилося більше активних завдань – замовлення відмічається виконаним
Extension	-
Post-Condition	-

Таблиця 1.5 - Варіант використання UC-4

Use case name	Перегляд розкладу роботи
Use case ID	UC-04
Goals	Перегляд розкладу роботи в календарі
Actors	Працівник, менеджер
Trigger	Користувач бажає переглянути власний розклад роботи
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на вкладку Розклад роботи, виводиться календар з поточним розкладом роботи. Користувач має змогу дивитися розклад роботи на інший проміжок часу, маючи кнопки для перемикання на попередній , поточний та наступний місяці.
Extension	-
Post-Condition	-

Таблиця 1.6 - Варіант використання UC-5

Use case name	Створення замовлення
Use case ID	UC-05
Goals	Створення нового замовлення
Actors	Адміністратор, Менеджер
Trigger	Користувач хоче створити нове замовлення
Pre-conditions	Користувач авторизований
Flow of Events	Користувач на сторінці замовлень натискає кнопку Нове замовлення, яка переносить користувача на сторінку нового замовлення. Користувач заповнює поля замовлення (клієнта, автомобіль, СТО) та додає завдання до замовлення. При натисканні кнопки Зберегти замовлення зберігається. При натисканні кнопки назад – замовлення видаляється
Extension	Не вдалося зберегти замовлення через невказані, чи неправильно вказані відповідні поля
Post-Condition	Користувач переноситься на сторінку замовлень

Таблиця 1.7 - Варіант використання UC-6

Use case name	Створення клієнта
Use case ID	UC-06
Goals	Додавання клієнта до БД
Actors	Адміністратор, Менеджер
Trigger	Користувач хоче додати нового клієнта
Pre-conditions	Користувач авторизований
Flow of Events	Користувач натискає на кнопку новий клієнт. Відкривається форма, де користувач заповнює ПІБ, номер телефону та знижку (опціонально) для нового клієнта. При натисканні кнопки створити створюється запис в БД. При натисканні кнопки відміна – закривається форма
Extension	Не вдалося створити користувача через неправильно заповнені поля – видається повідомлення про помилку
Post-Condition	Закривається форма для вводу даних клієнта

Таблиця 1.8 - Варіант використання UC-7

Use case name	Додавання автомобіля
Use case ID	UC-07
Goals	Додавання автомобіля до БД
Actors	Адміністратор, Менеджер
Trigger	Користувач хоче додати новий автомобіль в БД при створенні замовлення
Pre-conditions	Користувач авторизований
Flow of Events	Користувач натискає кнопку новий автомобіль. Відкривається форма, де користувач вводить марку, модель, рік, номер та опис (опціонально) автомобіля. При натисканні кнопки Створити – створюється запит БД з відповідними даними.. При натисканні кнопки відміна – форма закривається
Extension	Не вдалося створити користувача через неправильно заповнені поля – видається повідомлення про помилку
Post-Condition	Форма закривається

Таблиця 1.9 - Варіант використання UC-8

Use case name	Створення акаунту працівника
Use case ID	UC-08
Goals	Створення нового акаунту працівнику
Actors	Адміністратор
Trigger	Користувач хоче створити новий акаунт для працівника
Pre-conditions	Користувач авторизований у системі
Flow of Events	Користувач переходить на сторінку перегляду працівників та натискає кнопку Новий працівник. Відкривається форма, де користувач заповнює роль, ПІБ, номер телефону, тариф, СТО, відділ, посаду, логін та пароль. Всі поля мають обмеження від 3 символів до максимально можливої кількості, передбаченою БД. Номер телефону валідується в міжнародному форматі. При натисканні кнопки Створити – робиться відповідний запис в БД. При натисканні кнопки Відміна – форма закривається
Extension	При неправильно заповнених полях – виводиться повідомлення про помилку
Post-Condition	Форма закривається

Таблиця 1.10 - Варіант використання UC-9

Use case name	Редагування профілю користувача
Use case ID	UC-09
Goals	Редагування даних в БД про користувача
Actors	Адміністратор
Trigger	Користувач хоче редагувати профіль уже наявного користувача
Pre-conditions	Користувач авторизований
Flow of Events	Користувач натискає на кнопку редагування в таблиці навпроти рядка відповідного користувача. Користувача переносить на сторінку редагування. Користувач редагує відповідні поля, які витягнуті з БД, окрім паролю. Якщо користувач ввів пароль – створюється відповідний запит на зміну паролю та інших введених даних. Якщо користувач залишив поле пароль пустим – створюється відповідний запит без зміни паролю користувача. На кнопку зберегти – зберігаються дані в БД
Extension	Неправильно введені поля
Post-Condition	Переадресація на сторінку користувачів

Таблиця 1.11 - Варіант використання UC-10

Use case name	Редагування клієнта
Use case ID	UC-10
Goals	Редагування даних клієнта
Actors	Адміністратор, Менеджер
Trigger	Користувач хоче змінити дані про клієнта
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на сторінку редагування відповідного клієнта. Користувач має можливість редагувати відповідні поля щодо клієнту. Користувач має можливість переглядати та редагувати всі замовлення стосовно даного клієнта. При натисканні кнопки Зберегти – зберігається редагована інформація. При натисканні кнопки Назад – переадресація на сторінку клієнтів
Extension	Поля заповненні не правильно – висвічується повідомлення про помилку
Post-Condition	Переадресація на сторінку клієнтів

Таблиця 1.12 - Варіант використання UC-11

Use case name	Видалення працівника
Use case ID	UC-11
Goals	Видалення акаунту працівника з БД
Actors	Адміністратор
Trigger	Користувач хоче видалити працівника
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переходить на сторінку редагування працівника. Користувач натискає кнопку видалення працівника
Extension	-
Post-Condition	Переадресація на сторінку перегляду працівників

Таблиця 1.13 - Варіант використання UC-12

Use case name	Пошук працівника
Use case ID	UC-12
Goals	Пошук працівника по заданим фільтрам
Actors	Адміністратор
Trigger	Користувач хоче знайти працівника
Pre-conditions	Користувач авторизований
Flow of Events	Користувач вводить у поля пошуку необхідні йому фільтри для пошуку. Сторінка реагує на натискання кнопки пошуку відповідною дією, видаючи користувачеві тільки тих працівників, які підходять по заданому фільтру
Extension	-
Post-Condition	-

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. Нижче (таблиця 1.14) наведено загальну модель вимог, а в таблицях 1.15 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити в таблиці 1.28.

Таблиця 1.14 – модель вимог

№	Дія	Код
1.	Авторизація користувача	

Продовження таблиці 1.14

№	Дія	Код
1.1.	Ввід логіна	
1.1.1.	Перевірка довжини (мінімум 3 символи)	
1.1.2.	Перевірка довжини (максимум 20 символів)	
1.2.	Від пароля	
1.2.1.	Перевірка довжини (мінімум 3 символи)	
1.2.2.	Перевірка довжини (максимум 20 символів)	
2.	Перегляд графіку роботи	
2.1.	Вибір місяцю перегляду	
3.	Оформленн замовлення	
3.1.	Вибір клієнта	
3.1.1.	Вибір створеного клієнта	
3.1.2.	Створення нового клієнта	
3.1.2.1.	Ввід прізвища	
3.1.2.1.1.	Перевірка довжини (мінімум 3 символи)	
3.1.2.1.2.	Перевірка довжини (максимум 40 символів)	
3.1.2.2.	Ввід імені	
3.1.2.2.1.	Перевірка довжини (мінімум 3 символи)	
3.1.2.2.2.	Перевірка довжини (максимум 40 символів)	
3.1.2.3.	Ввід прізвища	
3.1.2.3.1.	Перевірка довжини (мінімум 3 символи)	
3.1.2.3.2.	Перевірка довжини (максимум 40 символів)	
3.1.2.4.	Ввід номеру телефону	
3.1.2.4.1.	Валідація номеру телефона	
3.2.	Вибір автомобіля	
3.2.1.	Вибір створеного автомобіля	
3.2.2.	Створення нового автомобіля	
3.2.2.1.	Ввід марки автомобіля	
3.2.2.1.1.	Перевірка довжини (мінімум 3 символи)	
3.2.2.1.2.	Перевірка довжини (максимум 80 символів)	
3.2.2.2.	Ввід моделі автомобіля	
3.2.2.2.1.	Перевірка довжини (мінімум 3 символи)	
3.2.2.2.2.	Перевірка довжини (максимум 80 символів)	
3.2.2.3.	Ввід року автомобіля	
3.2.2.3.1.	Перевірка довжини (дорівнює 4)	
3.2.2.4.	Ввід номеру автомобіля	
3.2.2.4.1.	Перевірка довжини (до 10 символів)	
3.2.2.5.	Ввід опису автомобіля	
3.3.	Вибір СТО	
3.4.	Вибір завдання	
3.4.1.	Вибір працівника	

Продовження таблиці 1.14

№	Дія	Код
3.4.2.	Вибір завдання	
4.	Пошук замовлення	
4.1.	Пошук замовлення по клієнту	
4.2.	Пошук замовлення по автомобілю	
5.	Перегляд списку завдань	
5.1.	Вивід списку завдань	
5.2.	Відмічення завдань виконаними	
5.2.1.	Відмічення замовлення виконаним	
5.2.1.1.	Перевірка виконаності всіх завдань для замовлення	
6.	Замовлення деталей	
6.1.	Пошук деталі по назві	
6.2.	Додавання деталі до списку замовлення	
6.3.	Замовлення деталі	
7.	Пошук працівника	
8.	Редагування працівника	
9.	Видалення працівника	
10.	Пошук послуги	
11.	Редагування працівника	
12.	Видалення працівника	

Таблиця 1.15 – Функціональна вимога FR-1

Назва	Авторизація користувача
Опис	Система повинна надавати можливість авторизації користувачу шляхом введення логіну та паролю

Таблиця 1.16 – Функціональна вимога FR-2

Назва	Перегляд розкладу
Опис	Система повинна надавати можливість перегляду, за наявності, користувачу його розклад роботи

Таблиця 1.17 – Функціональна вимога FR-3

Назва	Оформлення замовлення
Опис	Система повинна надавати можливість оформлення замовлення

Таблиця 1.18 – Функціональна вимога FR-4

Назва	Створення клієнта
Опис	Система повинна надавати можливість створення клієнта в БД

Таблиця 1.19 – Функціональна вимога FR-5

Назва	Створення автомобіля
Опис	Система повинна надавати можливість створення автомобіля в БД

Таблиця 1.20 – Функціональна вимога FR-6

Назва	Вибір послуги до замовлення
Опис	Система повинна надавати можливість вибирати послуги при створенні замовлення

Таблиця 1.21 – Функціональна вимога FR-7

Назва	Пошук замовлень
Опис	Система повинна надавати можливість пошуку створених замовлень

Таблиця 1.22 – Функціональна вимога FR-8

Назва	Перегляд списку задач
Опис	Система повинна надавати можливість перегляду списку задач

Таблиця 1.23 – Функціональна вимога FR-9

Назва	Виконання завдань
Опис	Система повинна надавати можливість відмічення завдань як виконані

Таблиця 1.24 – Функціональна вимога FR-10

Назва	Оформлення замовлення деталей
Опис	Система повинна надавати можливість оформлювати замовлення деталей

Таблиця 1.25 – Функціональна вимога FR-11

Назва	Пошук працівників
Опис	Система повинна надавати можливість пошуку працівників

Таблиця 1.26 – Функціональна вимога FR-12

Назва	Створення працівників
Опис	Система повинна надавати можливість створення акаунту працівників

Таблиця 1.27 – Функціональна вимога FR-13

Назва	Видалення працівників
Опис	Система повинна надавати можливість видаляти акаунти працівників

Таблиця 1.27 – Функціональна вимога FR-13

Назва	Редагування клієнта
Опис	Система повинна надавати можливість редагувати клієнта

Таблиця 1.27 – Функціональна вимога FR-13

Назва	Пошук клієнтів
Опис	Система повинна надавати можливість пошуку клієнтів

Таблиця 1.28 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12
FR-1	+											
FR-2				+								
FR-3					+							
FR-4					+	+						
FR-5					+		+					
FR-6					+							
FR-7					+							
FR-8			+									
FR-9			+									
FR-10		+										
FR-11									+			+
FR-12								+				
FR-13											+	
FR-14										+		
FR-15										+		

1.4.2 Розроблення нефункціональних вимог

Нефункціональною вимогою даного проекту є підтримка української мови.

1.5 Постановка задачі

Мета курсової роботи полягає у розробці веб-застосунку для підтримки роботи автосервісних станцій (СТО), спрямованого на оптимізацію та

автоматизацію ключових процесів, що відбуваються на таких станціях, шляхом розробки власного рушія, що є значно оптимізованішим за його аналоги. Розроблений продукт має на меті покращення управління робочими процесами, взаємодії з клієнтами та використання аналітики для підвищення ефективності обслуговування автотранспорту.

- Автоматизація робочих процесів: Розробка механізмів для ефективного планування та контролю робіт на СТО, враховуючи завдання та ресурси, що вимагаються для їх виконання.
- Інтеграція з базами даних: Реалізація системи управління базами даних для зберігання та обробки інформації про клієнтів, транспортні засоби, використані матеріали та виконані роботи.
- Створення зручного інтерфейсу користувача: Розробка інтуїтивно зрозумілого та зручного інтерфейсу для користувачів, що дозволить легко взаємодіяти з системою та моніторити стан робочих процесів.
- Забезпечення безпеки даних: Реалізація механізмів захисту конфіденційної інформації клієнтів та оптимізація доступу до даних відповідно до рівнів доступу користувачів.
- Тестування та валідація: Проведення тестування розробленого веб-застосунку для перевірки його функціональності та валідації відповідності вимогам.

Реалізація цих завдань спрямована на створення високоефективного інструменту для оптимізації робочих процесів на СТО, що полегшить керування сервісним центром, покращить обслуговування клієнтів та підвищить ефективність діяльності автомобільних сервісів.

Висновки до розділу

В процесі роботи над проектом в сфері авторемонту виявлено значну залежність від програмного забезпечення для покращення робочих процесів та підвищення якості послуг. Втім, існуючі програмні системи часто працюють ізольовано, що призводить до надмірного дублювання даних та появи помилок.

Також бракує передової аналітики та технологій штучного інтелекту, що викликає непередбачувані затримки та відсутність автоматичного контролю.

Для вирішення цих викликів було розроблено веб-застосунок, що об'єднує всі аспекти роботи СТО в єдиний інформаційний простір. Основна мета цього рішення полягає в підвищенні ефективності, зменшенні кількості помилок та покращенні якості надання послуг.

При виборі технічних рішень було обрано архітектуру HMVC через її масштабованість та модульність. Мови програмування PHP та JavaScript вибрані через їхню популярність та здатність сприяти динамічній взаємодії. Використання патерну HMVC дозволяє створювати незалежні модулі та ієрархічну організацію, забезпечуючи доступність до інших рівнів із збереженням при цьому незалежності. PHP, як широко використовувана мова веб-застосунків, може генерувати динамічну взаємодію з користувачем за допомогою JavaScript, який використовується для написання сценаріїв веб-сторінок.

Автори обговорюють інструменти та технології, які вони віддають перевагу, рекомендуючи Visual Studio Code як інтегроване середовище розробки завдяки простоті використання, наявності розширень та доступності. Базою даних обрано PostgreSQL через передові технології, спрощений синтаксис, підтримку одночасної модифікації та перспективність. Для зручності розробки для PostgreSQL використовується DataGrip. Автори також згадують про використання бібліотек pgsql і PDO в PHP, а також фреймворку Bootstrap для написання представлень. Вибір nginx як веб-сервера зумовлено простотою розгортання, сумісністю з різними базами даних та мовами програмування. Рішення розробки власного движка веб-сервера було прийняте через його простоту використання, швидкість розробки, безпеку та можливості кастомізації.

Основна мета проекту полягає в розробці веб-додатку, який оптимізує та автоматизує ключові процеси на станції технічного обслуговування

автомобілів. Серед цілей проекту - поліпшення управління робочим процесом, взаємодія з клієнтами та використання аналітики для підвищення ефективності обслуговування автомобілів. Завдання, пов'язані з досягненням цих цілей, включають ефективні механізми планування та контролю, інтеграцію з базами даних для зберігання та обробки інформації, розробку зручного інтерфейсу, забезпечення безпеки даних, а також ретельне тестування та валідацію. Загалом, мета цього проекту - оптимізація робочих процесів, полегшення управління сервісними центрами, покращення обслуговування клієнтів та підвищення ефективності автомобільного сервісу.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу створення замовлення програмного забезпечення використовується BPMN модель (рисунок 2.1-2.2).

Адміністратор або менеджер можуть створити замовлення в системі. Для цього вони повинні заповнити форму замовлення, яка містить інформацію про клієнта, автомобіль, проблему та інші деталі. Після заповнення форми замовлення, система автоматично створює нове замовлення та надсилає повідомлення працівникам СТО про нове замовлення.

Опис послідовності створення замовлення:

- Користувач заходить вибирає функцію створення нового замовлення
- Користувач натискає кнопку Вибрати напроти графу Замвоник.
- Користувач за допомогою поля пошук намагається знайти користувача
- У разі успіху всі поля про користувача заповнюються
- У разі провалу видається повідомлення та користувач заповнює всі повідомлення мануально
- Користувач натискає кнопку Додати.
- Заповнює поле Замовника
- Відкривається поле для заповнення автомобіля
- Користувач натискає кнопку Вибрати напроти графу Автомобіль
- Користувач за допомогою поля пошук намагається знайти автомобіль
- У разі успіху всі поля про користувача заповнюються
- У разі провалу видається повідомлення та користувач заповнює всі повідомлення мануально

- Користувач натискає кнопку Додати.
- Користувач вибирає СТО із наданого йому списку
- Користувач додає завдання для працівників по даному замовленню
- У працівників автоматично з’являється завдання
- Працівник виконує завдання
- Працівник відмічає виконання завдання в програмі
- Сервер перевіряє, якщо всі завдання по замовленню були виконані – завдання відмічається як виконане

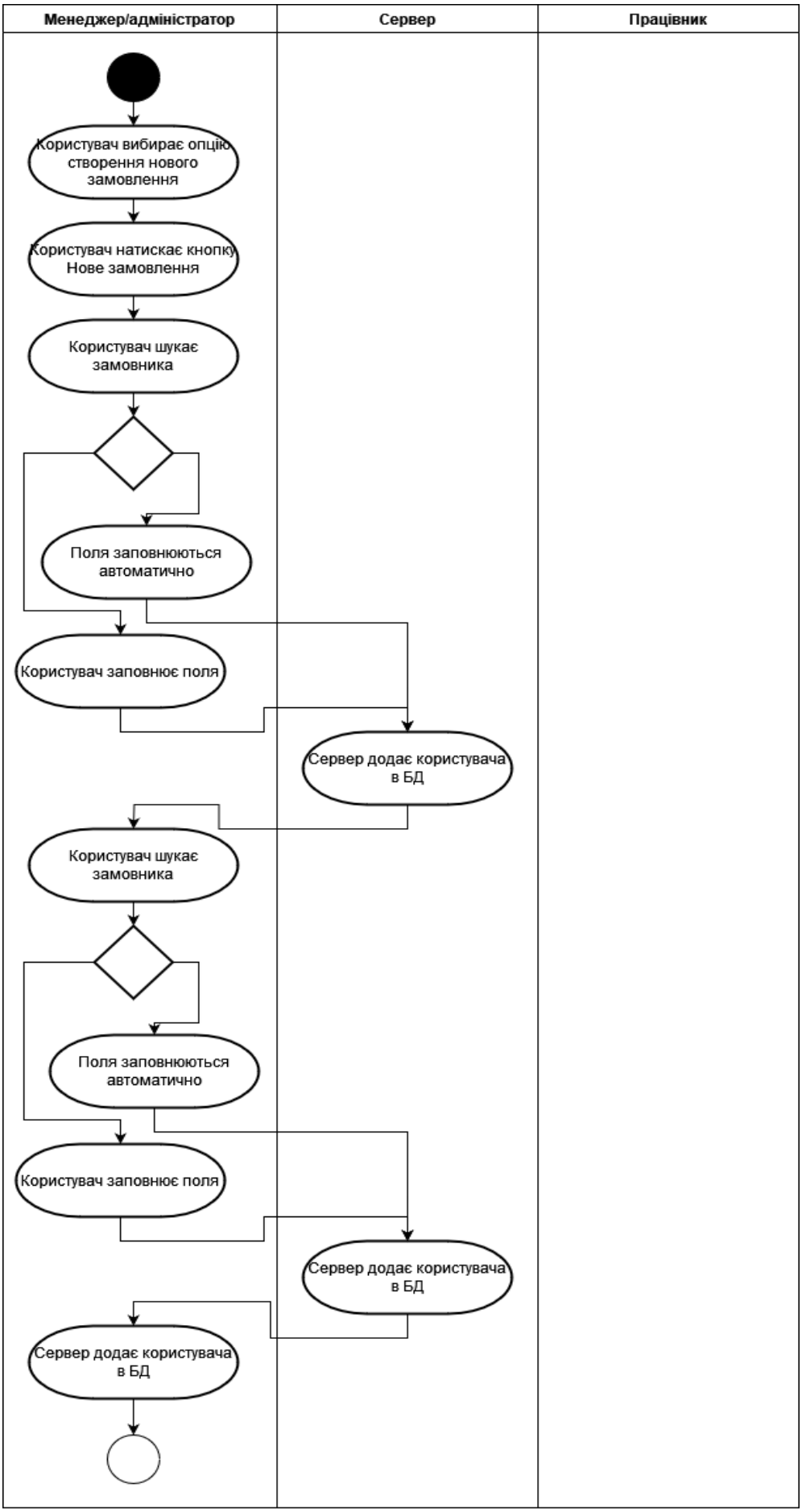

Рисунок 2.1 – BPMN модель

Рисунок 2.2 – BPMN модель

2.2 Архітектура програмного забезпечення

Веб-застосунок для підтримки роботи СТО реалізований за допомогою архітектурного патерну HMVC. Розглянемо даний патерн по різним характеристикам

- Продуктивність. Дане архітектурне рішення дозволяє оптимізувати використання ресурсів, оскільки розділяє логіку на модулі та компоненти
- Надійність. При некоректній роботі одного з модулів, всі інші модулі залишаються робочими.
- Маштабованість. Дане архітектурне рішення дозволяє широко маштабувати проект, без змін старого коду.
- Безпека. Дане архітектурне рішення дозволяє чітко розділити межі доступу до конкретних модулів, що унеможливлює потрапляння користувача на модулі, де йому доступ заборонений

Розглянувши загальні положення архітектури проекту можна розглянути детальну реалізацію. В моєму ПЗ я розробляю трьохрівневе дерево, де посилання складається з трьох частин, а саме: modul/class/action.

Рівень modul надає доступ до модуля працівника, наприклад: модуль admin дає можливість доступу тільки адміністраторам. Також важливо зазначити, що модуль index є доступним для всіх зареєстрованих користувачів.

Рівень class є підрівнем рівня modul та передбачає розподіл на великі функції користувача, наприклад: працівник має три функції: робота з деталями – detail, перегляд завдань – task та загальний базовий підмодуль – index.

Наступний і останній рівень розподілу тріад – action. Даний підклас є назвою конкретного контролера для виконання конкретної дії.

Дана схема дозволяє зручно зберігати та контролювати всі модулі та наглядно розподіляти функції користувачів.

Нижче наведена побудоване дерево модулів та опис їх роботи

- admin – даний модуль є модулем Адміністратора та предназначений для адміністрування клієнтів, замовлень, деталей та прайсу.
    - client – даний модуль предназначений для роботи з клієнтами. Його функціональьність дозволяє додавати, видаляти та редагувати клієнтів СТО. Також для зручності адміністрування передбачено можливість пошуку клієнтів.
    - employee – даний модуль передбачений для адміністрування працівниками СТО. Його функціональність дозволяє додавати нових працівників, редагувати та видаляти вже занесених в БД працівників. Також до було реалізовано пошук працівників. До того ж передбачена можливість зміни паролю працівникові.
    - index – даний модуль створений для перенаправлення на базову сторінку адміністратора, що дозволяє збільшити модульність проекту.
    - order – даний модуль передбачений для створення та редагування замовлення
    - service – даний модуль передбачає створення нової послуги та редагування та видалення уже доданої до БД.
- employee – модуль, забезпечуючий роботу працівника
    - detail – передбачає можливість замолення деталей
    - index – даний модуль був створений для перенаправлення на головну сторінку працівника та перенаправлення на сторінку розкладу працівника
    - task – даний модуль передбачає можливість працівникові переглянути його задачі та при виконанні відмітити їх як виконані
- index – загальний модуль проекту, створення для всі користувачів

    - index – містить функціонал виведення сторінки про помилку запиту, сторінки для авторизації користувача та перенаправлення на сторінки кожної ролі користувачів
    - schedule – модуль створений для роботи з розкладом користувачів. Передбачає перегляд розкладу за різний період часу
- manager – модуль для роботи менеджера на СТО
    - index – створений для перенаправлення на головну сторінку менеджера та перенаправлення на сторінку перегляду розкладу працівника
    - order – створений для можливості додавання, редагування та пошуку замовлень

2.3 Конструювання програмного забезпечення

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання користувачів, а також даних про їх замовлення та автомобілі. Також для зберігання працівників та вієї інформації про них. Опис таблиць бази даних наведено у таблицях 2.1 - 2.16. Модель бази даних наведена на рисунку 2.3.

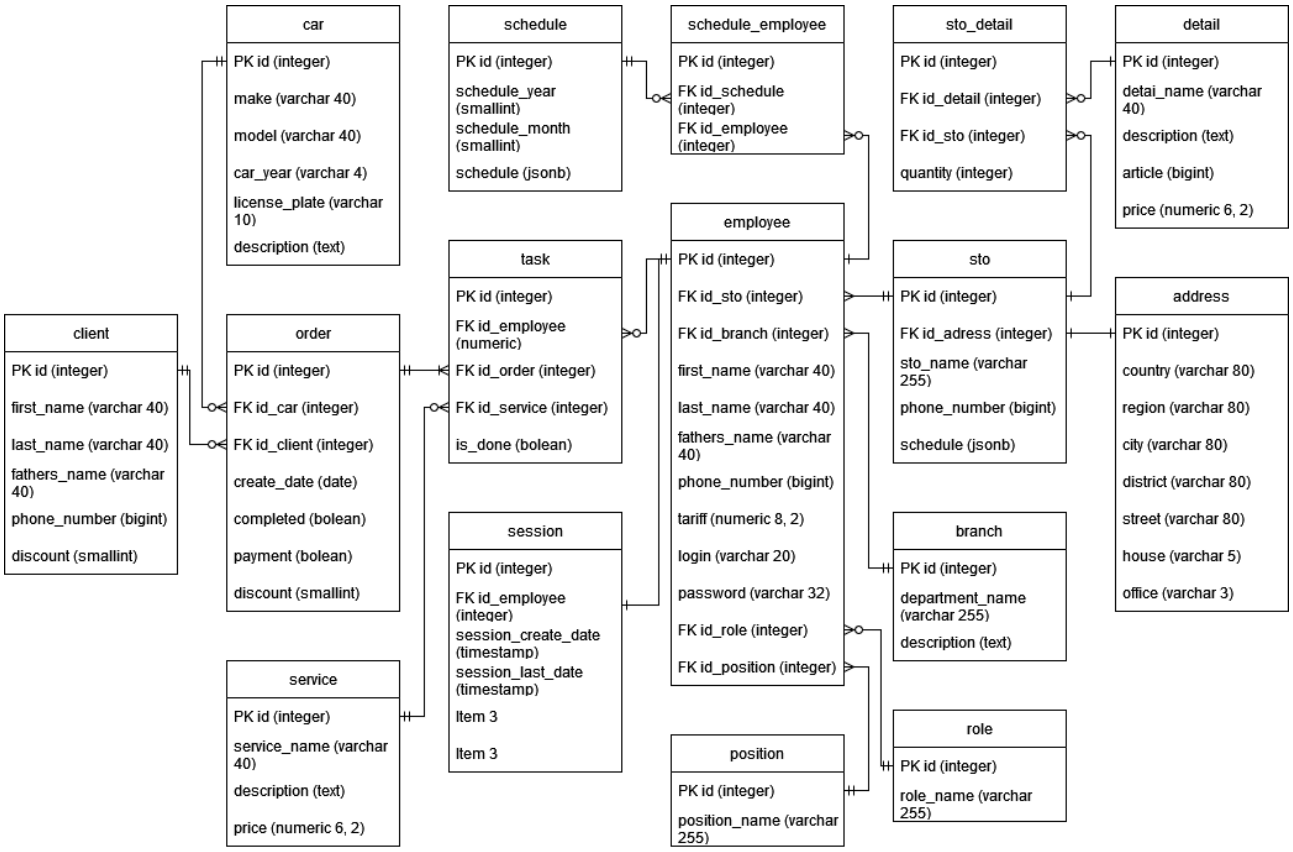

Рисунок 2.3 – ER-модель БД

Таблиця 2.1 – Опис таблиці employee

Таблиця	Назва поля	Тип даних	Опис
employee	id	serial	ідентифікаційний номер працівника
	first_name	VARCHAR(40)	Прізвище працівника
	last_name	VARCHAR(40)	Ім’я працівника
	fathers_name	VARCHAR(40)	Патронім працівника
	phone_number	BIGINT	Номер телефону працівника
	tariff	NUMERIC(8, 2)	Заробтня плата працівника

Продовження таблиці 2.1

	id_sto	INTEGER	Ідентифікаційний номер СТО
	id_branch	INTEGER	Ідентифікаційний номер відділу
	id_position	INTEGER	Ідентифікаційний номер посада
	id_role	INTEGER	Ідентифікаційний номер
	login	VARCHAR(20)	Логін користувача
	password	VARCHAR(32)	Пароль користувача збережений в md5

Таблиця 2.2 – Опис таблиці branch

Таблиця	Назва поля	Тип даних	Опис
branch	id	serial	Ідентифікатор відділу
	department_name	VARCHAR(255)	Назва відділу
	description	TEXT	Опис відділу

Таблиця 2.3 – Опис таблиці position

Таблиця	Назва поля	Тип даних	Опис
position	id	serial	Ідентифікатор посади
	position_name	VARCHAR(255)	Назва посади

Таблиця 2.4 – Опис таблиці role

Таблиця	Назва поля	Тип даних	Опис
role	id	serial	Ідентифікатор ролі
	role_name	VARCHAR(255)	Назва ролі

Таблиця 2.5 – Опис таблиці order

Таблиця	Назва поля	Тип даних	Опис
order	id	serial	Ідентифікатор замовлення
	id_client	INTEGER	Ідентинтифікатор клієнта
	create_date	DATE	Дата створення замоввлення
	completed	BOOLEAN	Чи було замовлення виконаним
	payment	BOOLEAN	Чи було замовлення оплаченим
	discount	SMALLINT	Зничка на замовлення
	id_car	INTEGER	Ідентифікатор машини

Таблиця 2.6 – Опис таблиці task

Таблиця	Назва поля	Тип даних	Опис
task	id	serial	Ідентифікатор завдання

Продовження таблиці 2.6

	id_order	INTEGER	Ідентифікатор замовлення
	id_service	INTEGER	Ідентифікатор послуги
	id_employee	INTEGER	Ідентифікатор працівника
	is_done	BOOLEAN	Чи виконане завдання

Таблиця 2.7 – Опис таблиці sto

Таблиця	Назва поля	Тип даних	Опис
sto	id	serial	Ідентифікатор замовлення
	id_address	INTEGER	Ідентифікатор адреси СТО
	sto_name	VARCHAR(255)	Назва СТО
	phone_number	BIGINT	Номер телефону СТО
	schedule	JSONB	Розклад СТО

Таблиця 2.8 – Опис таблиці client

Таблиця	Назва поля	Тип даних	Опис
client	id	serial	Ідентифікатор клієнта
	first_name	VARCHAR(40)	Прізвище клієнта

Продовження таблиці 2.8

	last_name	VARCHAR(40)	Ім’я клієнта
	fathers_name	VARCHAR(40)	Патронім клієнта
	phone_number	BIGINT	Номер телефону клієнта
	discount	SMALLINT	Знижка клієнта

Таблиця 2.9 – Опис таблиці car

Таблиця	Назва поля	Тип даних	Опис
car	id	serial	Ідентифікатор автомобіля
	make	VARCHAR(40)	Марка автомобіля
	model	VARCHAR(40)	Модель автомобіля
	car_year	VARCHAR(4)	Рік випуску автомобіля
	license_plate	VARCHAR(10)	Номер автомобіля
	description	TEXT	Опис автомобіля

Таблиця 2.10 – Опис таблиці service

Таблиця	Назва поля	Тип даних	Опис
service	id	serial	Ідентифікатор послуги
	service_name	VARCHAR(40)	Назва послуги
	description	TEXT	Опис послуги
	price	NUMERIC(6, 2)	Ціна послуги

Таблиця 2.11 – Опис таблиці detail

Таблиця	Назва поля	Тип даних	Опис
detail	id	serial	Ідентифікатор деталі
	detail_name	VARCHAR(40)	Назва деталі

Продовження таблиці 2.11

	Description	TEXT	Опис деталі
	article	BIGINT	Артикл деталі
	price	NUMERIC(6, 2)	Ціна деталі

Таблиця 2.12 – Опис таблиці sto_detail

Таблиця	Назва поля	Тип даних	Опис
sto_detail	id	serial	Ідентифікатор таблиці
	id_detail		Ідентифікатор деталі
	id_sto		Ідентифікатор СТО
	quantity		Кількість деталей

Таблиця 2.13 – Опис таблиці address

Таблиця	Назва поля	Тип даних	Опис
address	id	serial	Ідентифікатор адреси
	country	VARCHAR(80)	Країна
	region	VARCHAR(80)	Область

Продовження таблиці 2.13

	City	VARCHAR(80)	Місто
	district	VARCHAR(80)	Район
	street	VARCHAR(80)	Вулиця
	house	VARCHAR(5)	Будинок
	office	VARCHAR(3)	Офіс

Таблиця 2.14 – Опис таблиці schedule

Таблиця	Назва поля	Тип даних	Опис
schedule	id	serial	Ідентифікатор розкладу
	schedule_year	SMALLINT	Рік розкладу
	schedule_month	SMALLINT	Місяць розкладу
	schedule	JSONB	Розклад

Таблиця 2.15 – Опис таблиці schedule_employee

Таблиця	Назва поля	Тип даних	Опис
schedule_ employee	id	serial	Ідентифікатор таблиці
	id_employee	INTEGER	Ідентифікатор працівника
	id_schedule	INTEGER	Ідентифікатор розкладу

Таблиця 2.16 – Опис таблиці session

Таблиця	Назва поля	Тип даних	Опис
session	id	serial	Ідентифікатор сесії
	session_create_date	TIMESTAMP WITH TIME ZONE	Дата створення сесії
	session_last_date	TIMESTAMP WITH TIME ZONE	Дата останньої активності сесії

Продовження таблиці 2.16

	id_employee	INTEGER	Ідентифікатор працівника
	session_key	VARCHAR(128)	Ключ сесії
	data	TEXT	Тимчасова інформація для сесії

Оскільки даіграма класів була б надто велика для повного зображення в даній частині, на рисунку 2.4 – діаграма класів, наведена тільки для базових класів. Всі інші класи моделі наслідуються клас ActionClassModel, а класи контролерів наслідуються ActionClass. Їхні атрибути є атрибутами базового класу, а методи залежать від конкретних потреб і реалізацію.

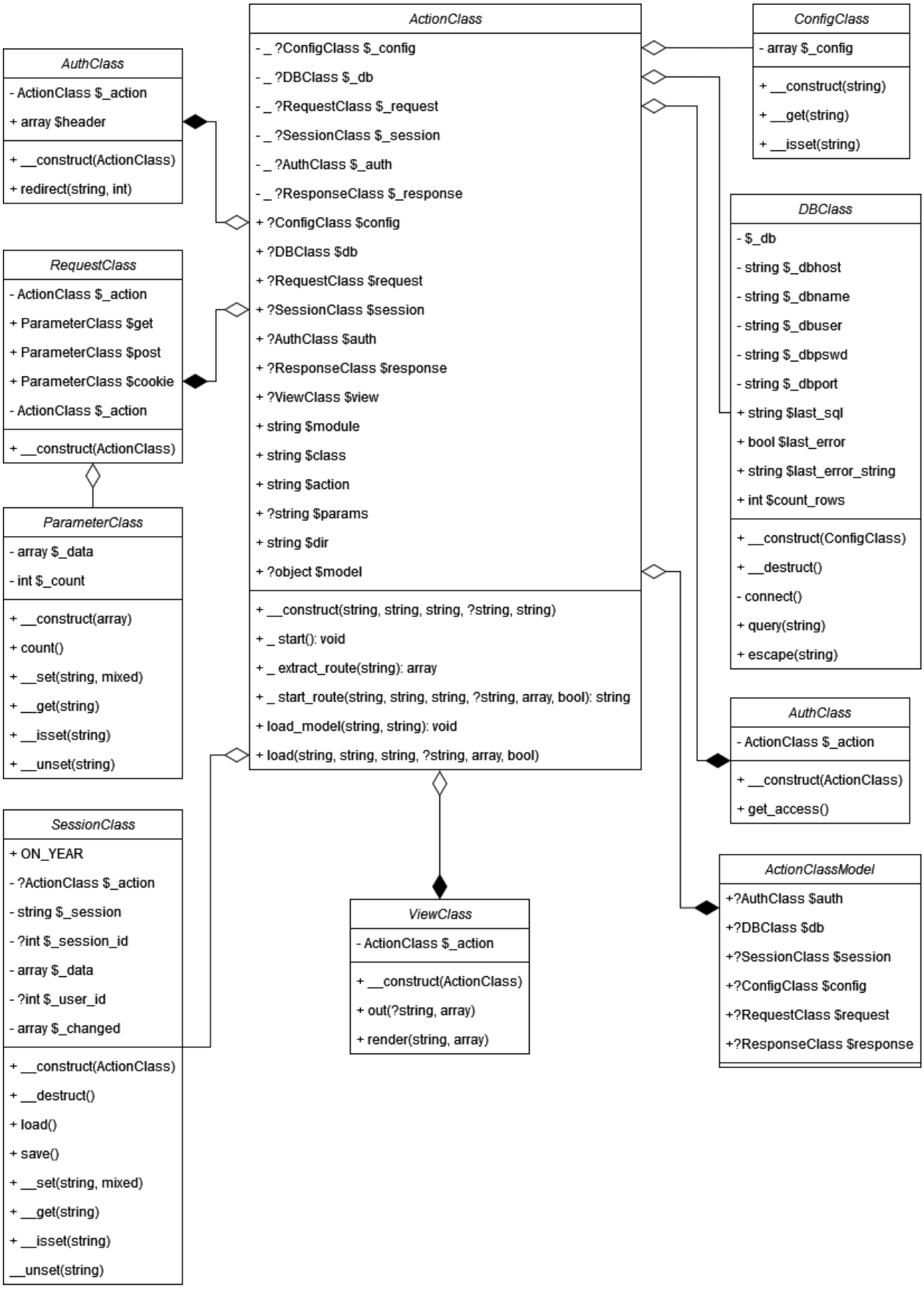

Рисунок 2.4 – Діаграма класів

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.17.

Таблиця 2.17 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Visual Studio Code	Головне середовище розробки програмного забезпечення серверної частини курсової роботи.
2	DataGrip	Програмне забезпечення для зручної зв’язки з БД

2.4 Аналіз безпеки даних

Для зберігання паролей в БД я використовую алгоритм хешування md5. Серед його переваг є швидкість, що робить його ефективним при великій кількості даних. Також він реалізований у більшості мов програмування. Із недоліків даного алгоритму можна зазначити, що він вважається застарілим в стандартах безпеки, проте досі залишається одним із найпопулярніших рішень. Також із недоліків – відсутність солі та вразливість до атак зіткненням.

Для зберігання ключа сесії я використовую алгоритм SHA-512. Його перевагами є стійкість до зіткнення, високий рівень безпеки. Ще його особливістю є велика довжина хешу, що з точки зору безпеки ускладнює його взлом, проте з цим велика довжина хешу є недоліком при неможливості зберігати великі обсяги пам’яті.

Висновки до розділу

Під час виконання роботи над системою для станції технічного обслуговування автомобілів використовувалась модель BPMN для відображення бізнес-процесу створення замовлення на програмне забезпечення. Процес ініціюється адміністратором або менеджером, який заповнює форму замовлення із даними про клієнта, автомобіль, проблему та інші деталі. Після заповнення форми система генерує нове замовлення і автоматично повідомляє працівників СТО.

В процесі створення замовлення користувач виконує послідовні кроки: входить в систему, вибирає опцію створення нового замовлення і вказує клієнта. Під час вибору клієнта виконується автоматичний пошук, і у разі успіху дані автоматично заповнюються, в іншому випадку вводяться вручну. Аналогічні кроки повторюються для введення даних про автомобіль. Далі користувач обирає станцію технічного обслуговування і додає завдання для працівників. Система автоматично призначає завдання працівникам, які після виконання позначають його у програмі. Сервер перевіряє завершені завдання та відзначає замовлення як виконане.

Веб-додаток для СТО розроблено з використанням архітектурного патерну HMVC, що забезпечує оптимізацію використання ресурсів, надійність та можливість масштабування системи без змін існуючого коду. Розробка програмного забезпечення передбачає структуру з трьох рівнів - модулів, класів та дій. Ця структура дозволяє чітко розділити функціональні можливості та підвищити безпеку, регламентуючи доступ до конкретних модулів.

Реалізація системи включає модулі для адміністраторів, працівників та менеджерів, кожен з яких відповідає за певні функції. Наприклад, модуль адміністратора управляє клієнтами, замовленнями, деталями та прайс-листами. Модуль працівника відповідає за інформацію про працівників та їх завдання. Така різноманітність модулів дозволяє кожному користувачеві здійснювати лише ті дії, які відповідають його ролі в системі.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для аналізу якості коду скористаємося програмою PHPMetrics[7]. Проаналізувавши нею наш код ми можемо отримати наступні результати:

- LOC
  - Lines of code 4044
  - Logical lines of code 2535
  - Comment lines of code 1509
  - Average volume 1264.5
  - Average comment weight 42.71
  - Average intelligent content 42.71
  - Logical lines of code by class 82
  - Logical lines of code by method 15
- Object oriented programming
  - Classes 31
  - Interface 0
  - Methods 170
  - Methods by class 5.48
  - Lack of cohesion of methods 1.1
- Coupling
  - Average afferent coupling 1.23
  - Average efferent coupling 1.35
  - Average instability 0.78
  - Depth of Inheritance Tree 1.2
- Package
  - Packages 2
  - Average classes per package 15.5
  - Average distance 0.46

    - Average incoming class dependencies 11.5
    - Average outgoing class dependencies 3.5
    - Average incoming package dependencies 1
    - Average outgoing package dependencies 1
- Complexity
    - Average Cyclomatic complexity by class 13.84
    - Average Weighted method count by class 18.32
    - Average Relative system complexity 45.39
    - Average Difficulty 12
- Bugs
    - Average bugs by class 0.42
    - Average defects by class (Kan) 0.86
- Violations
    - Critical 0
    - Error 12
    - Warning 12
    - Information 4

Оцінивши результат по даним метрикам та по більш детальному звіту, який надав сервіс можна зробити висновки, що хоч і є деякі класи, які перевантажені функціоналом, що в подальшому може скласти проблеми в підтримуванні коду, проте загалом результат можна вважати хорошим.

Також ще одним із параметрів можна вважати швидкість завантаження сторінки. Нижче наведено список деяких сторінок та час їх завантаження

- Сторінка входу користувача – 129мс
- Сторінка адміністратора/працівники – 197мс
- Сторінка адміністратора/перегляду замовлень – 138 мс
- Сторінка працівника/перегляд поточних задач – 152 мс

З огляду на дані цифри можу відзначити чудовий результат, оскільки дані цифри я більше ніж достатніми для швидкої роботи сайту .

3.2 Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.30.

Таблиця 3.1 – Тест 1.1

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Логін, пароль
Опис проведення тесту	У відповідні поля вводяться: коректний логін, який до цього був зареєстрований адміністратором в системі від 3 до 20 символів, пароль від 3 до 20 символів. Після цього натискає кнопку Увійти
Очікуваний результат	Користувач заходить у систему
Фактичний результат	Користувач заходить у систему

Таблиця 3.2 – Тест 2.1

Тест	Додавання працівника
Модуль	Адміністратор/працівники
Номер тесту	2.1
Початковий стан системи	Користувач зареєстрований, має роль admin та знаходить на сторінці перегляду працівників
Вхідні данні	Роль, прізвище, ім’я, по-батькові, номер телефону, тариф, СТО, Відділ, Посада, Логін, Пароль

Перенесення таблиці 3.2

Опис проведення тесту	Користувач натискає на кнопку Новий працівник. Відкривається модальне вікно, де користувач заповнює всі надані йому поля. Потім натискає кнопку Створити. Модальне вікно закривається.
Очікуваний результат	Працівник додається в таблицю працівників
Фактичний результат	Працівник додається в таблицю працівників

Таблиця 3.3 – Тест 2.2

Тест	Редагування послуги
Модуль	Адміністратор/послуги
Номер тесту	2.2
Початковий стан системи	Користувач зареєстрований, має роль admin та знаходить на сторінці перегляду працівників
Вхідні данні	Послуга, яку потрібно змінити, нова назва, новий опис та нова ціна
Опис проведення тесту	Користувач натискає на кнопку редагування відповідного замовлення. Вводить потрібні для редагування дані, натискає кнопку зберегти
Очікуваний результат	Замовлення було відредаговано
Фактичний результат	Замовлення було відредаговано

Таблиця 3.4 – Тест 2.3

Тест	Додавання послуги

Продовження таблиці 3.4

Модуль	Адміністратор/послуги
Номер тесту	2.3
Початковий стан системи	Користувач зареєстрований, має роль admin та знаходить на сторінці перегляду послуг
Вхідні данні	Назва, опис та ціна послуги
Опис проведення тесту	Користувач натискає на кнопку Нова послуга. Вводить потрібні дані. Натискає на кнопку Додати
Очікуваний результат	Послуга додається до БД
Фактичний результат	Послуга додається до БД

Таблиця 3.5 – Тест 2.4

Тест	Редагування замовлення
Модуль	Адміністратор/замовлення
Номер тесту	2.4
Початковий стан системи	Користувач зареєстрований, має роль admin та знаходить на сторінці перегляду працівників
Вхідні данні	Замовлення, знижка, послуги, які потрібно додати
Опис проведення тесту	Користувач натискає на кнопку редагування відповідного замовлення. Вводить потрібні дані. Натискає на кнопку зберегти
Очікуваний результат	Замовлення відредаговано
Фактичний результат	Замовлення відредаговано

Таблиця 3.6 – Тест 2.5

Тест	Видалення працівника
Модуль	Адміністратор/працівники
Номер тесту	2.5
Початковий стан системи	Користувач зареєстрований, має роль admin та знаходить на сторінці перегляду працівників
Вхідні данні	Працівник
Опис проведення тесту	Користувач натискає на кнопку редагування відповідного користувача. Користувач натискає на кнопку Видалити
Очікуваний результат	Працівник видаляється
Фактичний результат	Працівник видаляється

Таблиця 3.7 – Тест 3.1

Тест	Створення нового клієнта
Модуль	Менеджер/замовлення
Номер тесту	3.1
Початковий стан системи	Користувач зареєстрований, має роль manager та знаходить на сторінці перегляду замовлень
Вхідні данні	Прізвище, ім’я, по-батькові клієнта, номер телефону та знижка
Опис проведення тесту	Користувач натискає на кнопку Нове замовлення. Користувач натискає на кнопку Вибрати у графі Клієнт. Користувач натискає на кнопку створити замовлення. Користувач натискає на кнопку Додати

Перенесення таблиці 3.7

Очікуваний результат	Клієнта додано
Фактичний результат	Клієнта додано

Таблиця 3.8 – Тест 3.2

Тест	Оформлення замовлень
Модуль	Менеджер/замовлення
Номер тесту	3.2
Початковий стан системи	Користувач зареєстрований, має роль manager та знаходить на сторінці перегляду замовлень
Вхідні данні	Замовник, автомобіль, СТО, список завдань
Опис проведення тесту	Користувач натискає на кнопку Нове замовлення. Користувач натискає на кнопку Вибрати в графі Замовник. Користувач заповнює відповідні поля по клієнту. Користувач натискає кнопку Додати. Користувач натискає на кнопку Вибрати в графі Автомобіль. Користувач заповнює відповідні поля для автомобіля. Користувач натискає кнопку Додати. Користувач вибирає СТО. Користувач натискає кнопку Нове завдання. Користувач вводить відповідні поля. Користувач натискає кнопку створити. Користувач натискає кнопку зберегти
Очікуваний результат	Замовлення додано
Фактичний результат	Замовлення додано

Таблиця 3.9 – Тест 4.1

Тест	Замовлення деталей
Модуль	Працівник/деталі
Номер тесту	4.1
Початковий стан системи	Користувач зареєстрований, має роль employee та знаходить на сторінці оформлення замовлення деталей
Вхідні данні	Деталі які потрібно замовити
Опис проведення тесту	Користувач вводить в пошук назву деталі. Користувач натискає кнопку додати. Користувач вводить кількість. Користувач натискає кнопку замовити
Очікуваний результат	Замовлення сформоване
Фактичний результат	Замовлення сформоване

Таблиця 3.10 – Тест 4.2

Тест	Виконання завдань
Модуль	Працівник/завдання
Номер тесту	4.2
Початковий стан системи	Користувач зареєстрований, має роль employee та знаходить на сторінці перегляду завдань
Вхідні данні	-
Опис проведення тесту	Користувач бачить свій список поточних завдань. Користувач натискає на чекбокс відповідного виконаного завдання
Очікуваний результат	Завдання відмічене як виконане
Фактичний результат	Завдання відмічене як виконане

Висновки до розділу

Після аналізу якості коду за допомогою програми PHPMetrics та оцінки різноманітних метрик можемо зробити наступні висновки:

Lines of Code (LOC): Загальна кількість рядків коду - 4044, з яких 2535 є логічними рядками коду, а 1509 - коментарями. Середня кількість логічних рядків на клас складає 82, а на метод - 15.

Object Oriented Programming: Проект включає 31 клас та 170 методів. Виявлено певний розподіл методів за класами, з невеликим показником втрати зв'язності методів (Lack of cohesion of methods) - 1.1.

Coupling: Загальний показник зв'язаності (coupling) є невеликим, що свідчить про добру архітектурну структуру проекту. Середні значення зв'язку (afferent та efferent coupling) досить низькі.

Package: Проект поділений на 2 пакети з середньою кількістю класів у пакеті - 15.5. Існує деяка взаємодія між пакетами.

Complexity: Середні значення цикломатичної складності та інших метрик складності вказують на те, що код проекту в цілому є легким для розуміння та модифікації.

Bugs: Середні значення багів та дефектів на клас свідчать про те, що код є досить надійним.

Violations: Загальна кількість порушень (violations) є невеликою, але варто звернути увагу на 12 помилок та 12 попереджень.

Швидкість завантаження сторінок: Час завантаження сторінок дуже задовільний і вказує на високу ефективність роботи веб-додатку. Найбільше часу потрібно для сторінки адміністратора/працівників, проте навіть це значення (197мс) є дуже швидким.

Загалом, не дивлячись на деяке перевантаження функціоналом деяких класів, загальна якість коду є задовільною. Швидкість завантаження сторінок

також свідчить про високий рівень оптимізації. Важливо продовжувати вдосконалювати код та реагувати на виявлені помилки та порушення для підтримки високого рівня ефективності та якості продукту.

Також було проведене мануальне тестування, що забезпечило додаткову перевірку якості коду

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання програми потрібно завантажити сервер nginx [4] версії 1.24 для Windows. Також потрібно завантажити php [6] версії 8.3. Для зв’язки php та nginx потрібно використати програму RunHiddenConsole [2], яку можна завантажити на сайті серверу. Також для успішної роботи потрібно встановити базу даних PostgreSQL [8] версії 14.

Для початку потрібно повстановлювати всі програми у відповідні дерикторії. Далі потрібно налаштувати сервер: в папці, де знаходиться сервер/conf/nginx.conf потрібно вставити наступний текст

```
events {
    worker_connections  1024;
}

http {
     server {
          listen       80;
        server_name  sto.curswork;
          root c:/web/www;

          location /data/ {
            alias C:/web/www/data/;
        }

          location / {
               try_files $uri $uri/ /index.php?$args;
               break;
          }

          location ~ /index.php$ {
               fastcgi_pass   127.0.0.1:9123;
               fastcgi_index  c:/web/www/index.php;
               fastcgi_param                  SCRIPT_FILENAME
$document_root$fastcgi_script_name;
               include        fastcgi_params;
               break;
          }
        index  index.php;
     }
}
```

Всі директорії потрібно вказати відповідно до системних директорій та розташування серверу.

Наступним кроком потрібно створити файли запуску для серверу. З директорії серверу потрібно створити файл з назвою start.cmd, у нього вставити наступний код

```
@echo on
echo Starting servers...
set PHP_FCGI_MAX_REQUESTS=0
cd c:\web\bin\nginx\
c:\web\bin\RunHiddenConsole.exe c:\web\bin\nginx\nginx.exe
c:\web\bin\RunHiddenConsole.exe  c:\web\bin\php\php-cgi.exe  -b
127.0.0.1:9123 -c c:/web/bin/php/php.ini
cd ..
```

Всі директорії потрібно вказати відповідно до системних директорій та розташування серверу.

Також для зупинки серверу потрібно створити файл з назвою stop.cmd, та наповнити його наступним кодом

```
@echo on
echo Shutting down servers...
taskkill /IM nginx.exe /F
taskkill /IM php-cgi.exe /F
```

Також для успішного запуску потрібно в файлі php.ini підключити модуль pgsql, розкоментувавши відповідний рядок.

При успішному виконанні команд, подвійним кліком на файл start.cmd запуститься веб-сервер

Наступним кроком потрібно клонувати репозиторій з github у відповідну директорію

Задля безпеки з репозиторію був видалений файл config.php. Даний файл потрібно створити в директорії. Даний файл потрібно наповнити наступними даними:

```
<?
$dbhost = '';
$dbname = '';
$dbuser = '';
$dbpswd = '';
$dbport = '';

$timeZone = '';
```

```
$sessionKey = '';
$salt = '';
$projectName = '';
```

Для успішного підключення до БД потрібно заповнити усі поля

- dbhost – розташування БД
- dbname – ім’я БД
- dbuser – користувач БД
- dbpswd – пароль БД
- dbport – порт БД
- timeZone – таймзона
- sessionKey – назва ключа для сесії
- salt – сіль для алгоритмів захисту
- projectName – ім’я проекту

Останнім пунктом потрібно створити БД відповідно схемі описаній в розділах вище та наповнити їх даними.

Перезапустити сервер в програма буде готова до роботи.

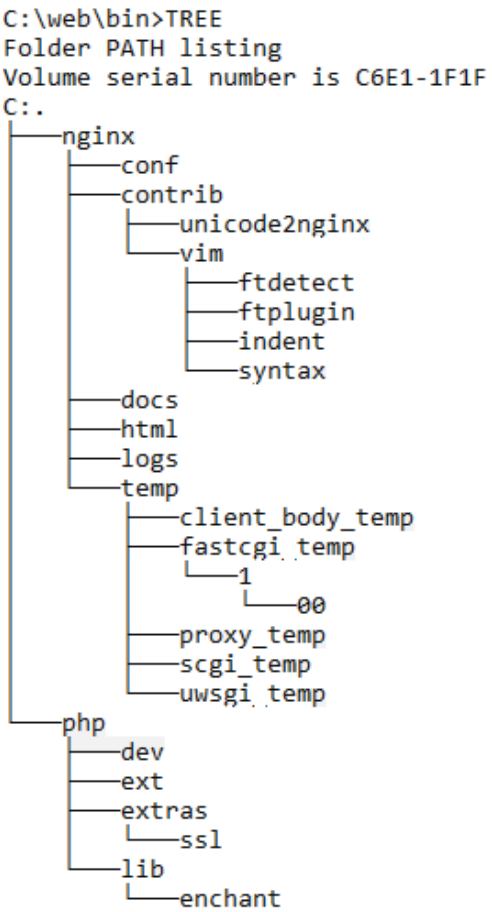

Рисунок 4.1 – дерево директрорій для файлів

4.2 Підтримка програмного забезпечення

Для оновлення проекту користувачі повинні відслідковувати вихід нової версії на github. При виявленні нової версії системний адміністратор має завантажити всі файли та розмістити їх відповідно інструкції. При необхідності зміни структури проекту користувачів буде повідомлено у патчноуті.

Користувач має мати можливість вибору версії для роботи шляхом завантаження різних версій з github.

Підтримка програмного забезпечення має передбачати оновлення до більш сучасних версій використовуваного програмного забезпечення, такого як СКБД, IDE та інших сторонніх системах.

Документація також має бути забезпечена актуальним та своєчасним оновленням

Нові версії програми мають нумеруватись як v*.*.*., де замість * - цифри, для задання відповідного номеру версії

Також вихід версій передбачає pull request в гілку main.

Наступними кроками в подальшій розробці програмного забезпечення стане написання таких модулів, як:

- клієнт, що дозволить працювати клієнту із веб-застосунком;
- директор, що дозволить роздавати завдання та визначати розклад директору підлеглим;
- склад, що дозволить працівнику складу слідкувати за деталями та приймати замовлення

Висновки до розділу

Розгортання програмного забезпечення вимагає виконання кількох кроків, які були детально описані в інструкції. Для ефективного функціонування веб-додатку необхідно завантажити та налаштувати сервер Nginx версії 1.24 для Windows, а також встановити PHP версії 8.3 та базу даних

PostgreSQL версії 14. Для зв'язку PHP та Nginx використовується програма RunHiddenConsole.

При налаштуванні серверу, у файлі nginx.conf вказуються параметри, такі як максимальна кількість одночасних підключень та шляхи до необхідних директорій. Створюються файли запуску та зупинки сервера, що полегшує процес управління сервером.

Для успішного використання додатку необхідно створити файл конфігурації `config.php` та заповнити його необхідними даними для підключення до бази даних та інших налаштувань.

Окремим кроком є клонування репозиторію з GitHub та створення бази даних відповідно до схеми, описаної вище.

Швидкість завантаження сторінок є дуже задовільною, а зупинка та перезапуск сервера виконується за допомогою створених файлів команд.

Для підтримки програмного забезпечення та оновлення проекту слід відслідковувати нові версії на GitHub та виконувати оновлення відповідно до інструкцій. Оновлення може включати в себе вдосконалення використовуваного програмного забезпечення, вирішення помилок та додавання нових функцій. Наявність документації та планування патчноутів є ключовим елементом підтримки.

Користувачі можуть вибирати версії для роботи та спростити вибір за допомогою відповідних засобів. Оновлення програмного забезпечення також може включати оновлення використовуваних інструментів, таких як СКБД та IDE, для забезпечення сучасності та безпеки системи.

Документація повинна бути завжди актуальною та доступною користувачам для ефективного використання програмного забезпечення.

ВИСНОВКИ

Проектування та розгортання веб-додатку було виконано з використанням сучасних технологій, таких як Nginx, PHP 8.3 та PostgreSQL 14. Використання метрик якості коду свідчить про високий рівень проекту.

Розгорнутий веб-додаток може знайти застосування в галузі автосервісів для ефективного керування замовленнями та підтримки персоналу. Впровадження даних технологій може позитивно вплинути на роботу підприємств цієї галузі.

Розроблена система допомагає вдосконалити процеси управління та планування на автосервісах, що має важливе значення для підвищення продуктивності та якості обслуговування клієнтів.

Продовження досліджень у напрямку оптимізації та розширення функціоналу системи може сприяти її подальшому вдосконаленню. Дослідження нових технологій та їх впровадження може зробити систему ще більш конкурентоздатною.

В результаті виконання курсової роботи було спроектовано веб-застосунок для підтримки роботи СТО

В якості середовища розробки обрано Nginx та PHP

У якості БД використано PostgreSQL

Після реалізації застосунку він був протестований в різних веб-браузерах, щоб переконатися у працездатності застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Bulma: Free, open source, and modern CSS framework based on Flexbox. Bulma: Free, open source, and modern CSS framework based on Flexbox. URL: https://bulma.io/ (дата звернення: 31.12.2023).
2) GitHub - wenshui2008/RunHiddenConsole: Hide console window for windows programs. GitHub. URL: https://github.com/wenshui2008/RunHiddenConsole (дата звернення: 31.12.2023).
3) JetBrains. DataGrip: The Cross-Platform IDE for Databases & SQL by JetBrains. JetBrains. URL: https://www.jetbrains.com/datagrip/ (дата звернення: 31.12.2023).
4) nginx. nginx news. URL: https://nginx.org/en/ (дата звернення: 31.12.2023).
5) pgAdmin - PostgreSQL Tools. pgAdmin - PostgreSQL Tools. URL: https://www.pgadmin.org/ (дата звернення: 31.12.2023).
6) PHP: Hypertext Preprocessor. PHP: Hypertext Preprocessor. URL: https://www.php.net/ (дата звернення: 31.12.2023).
7) PhpMetrics, static analysis for PHP - by Jean-François Lépine. PhpMetrics, static analysis for PHP - by Jean-François Lépine. URL: https://phpmetrics.org/ (дата звернення: 31.12.2023).
8) PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 31.12.2023).
9) UIkit. UIkit. URL: https://getuikit.com/ (дата звернення: 31.12.2023).
10) Учасники проектів Вікімедіа. Bootstrap – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Bootstrap (дата звернення: 31.12.2023).
11) Учасники проектів Вікімедіа. C Sharp – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 31.12.2023).
12) Учасники проектів Вікімедіа. HMVC – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HMVC (дата звернення: 31.12.2023).
13) Учасники проектів Вікімедіа. HTML – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення: 31.12.2023).

14) Учасники проектів Вікімедіа. Java – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Java (дата звернення: 31.12.2023).
15) Учасники проектів Вікімедіа. JavaScript – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 31.12.2023).
16) Учасники проектів Вікімедіа. Microsoft SQL Server – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення: 31.12.2023).
17) Учасники проектів Вікімедіа. Microsoft Visual Studio – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio (дата звернення: 31.12.2023).
18) Учасники проектів Вікімедіа. MySQL – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення: 31.12.2023).
19) Учасники проектів Вікімедіа. Navicat – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Navicat (дата звернення: 31.12.2023).
20) Учасники проектів Вікімедіа. nginx – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Nginx (дата звернення: 31.12.2023).
21) Учасники проектів Вікімедіа. Notepad++ – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Notepad++ (дата звернення: 31.12.2023).
22) Учасники проектів Вікімедіа. PHP – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/PHP (дата звернення: 31.12.2023).
23) Учасники проектів Вікімедіа. PhpStorm – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/PhpStorm (дата звернення: 31.12.2023).
24) Учасники проектів Вікімедіа. PostgreSQL – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/PostgreSQL (дата звернення: 31.12.2023).
25) Учасники проектів Вікімедіа. Python – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 31.12.2023).
26) Учасники проектів Вікімедіа. Rust (мова програмування) – Вікіпедія. Вікіпедія. URL:

https://uk.wikipedia.org/wiki/Rust_(мова_програмування) (дата звернення: 31.12.2023).

27) Учасники проектів Вікімедіа. SQLite – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/SQLite (дата звернення: 31.12.2023).

28) Учасники проектів Вікімедіа. Visual Studio Code – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 31.12.2023).

29) Учасники проектів Вікімедіа. Мікросервіси – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Мікросервіси (дата звернення: 31.12.2023).

30) Учасники проектів Вікімедіа. Модель-вид-контролер – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер (дата звернення: 31.12.2023).