**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Факультет кібербезпеки та програмної інженерії**
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Нестеренко К.С.
“____”_____2023 р.

# КВАЛІФІКАЦІЙНА РОБОТА
## (ПОЯСНЮВАЛЬНА ЗАПИСКА)

**ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ**
**МАГІСТРА**

**Тема:** «Система для автоматизації роботи АЗС»

**Виконавець:** Семенченко Дмитро Сергійович

**Керівник:** Горський Олексій Миколайович

**Нормоконтролер:**

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

**Факультет** кібербезпеки та програмної інженерії
**Кафедра** інженерії програмного забезпечення
**Освітній ступінь** магістр
**Спеціальність** 121 «Інженерія програмного забезпечення»
**Освітньо-професійна програма** «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Нестеренко К.С.
«___» _____ 2023 р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Семенченка Дмитра Сергійовича

1. Тема кваліфікаційної роботи: «Система для автоматизації роботи АЗС» затверджена наказом ректора від 04.10.2023 р. № 2034/ст

2. Термін виконання проекту: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні данні до проекту: інформаційна система для автоматизації АЗС, наявність модуля авторизації, наявність бази даних, наявність модуля для взаємодії з базою даних, наявність оригінального інформаційного наповнення.

4. Зміст пояснювальної записки:
   1. Аналіз розробки системи для автоматизації роботи АЗС
   2. Проектування ІС для колонки самообслуговування на АЗС
   3. Програмна реалізація інформаційної системи для автоматизації АЗС
   4. Тестування програмного застосунку

5. Перелік обов’язкових слайдів презентації:
   1. Існуюче програмне забезпечення
   2. Функціональні та нефункціональні вимоги
   3. Структура системи

6. Календарний план-графік

| № пор | Завдання | Термін виконання | Відмітка про виконання |
|---|---|---|---|
| 1. | Складання та затвердження графіку роботи дипломного проектування Написання 1 розділу, представлення керівнику | 02.10.23 – 10.10.23 | |
| 2. | Попередній друк 1 розділу та допоміжних сторінок (чорновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, 1-й нормо-контроль. | 11.10.23 – 15.10.23 | |
| 3. | Написання 2 розділу, представлення керівнику | 16.10.23 – 30.10.23 | |
| 4. | Написання 3 розділу, представлення керівнику | 01.11.23 – 22.11.23 | |
| 5. | Написання 4 розділу, представлення керівнику | 23.11.23 – 22.12.23 | |
| 6. | Загальне редагування та друк пояснювальної записки, графічного матеріалу | 23.12.23 – 24.12.23 | |
| 7. | Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки. | 25.12.23 – 27.12.23 | |
| 8. | Розробка тексту доповіді. Оформлення графічного матеріалу для презентації | 28.12.23 – 28.12.23 | |
| 9. | Отримання відгуку керівника, рецензії. | 29.12.23 – 29.12.23 | |
| 10. | Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці | 30.12.23 – 31.12.23 | |

7. Дата видачі завдання 02.10.2023

Керівник: Олексій ГОРСЬКИЙ

Завдання прийняв до виконання: Дмитро СЕМЕНЧЕНКО

# РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Система для автоматизації роботи АЗС»: 53 сторінки, 9 рисунків, 0 таблиць, 29 використаних джерел, 1 додаток.

ІНФОРМАЦІЙНА СИСТЕМА, ПРОГРАМНИЙ ПРОДУКТ, АЗС, ІНТЕЛЕКТУАЛЬНІ ТЕХНОЛОГІЇ, АВТОМАТИЗАЦІЯ

**Об'єкт дослідження** – методи і засоби розробки інформаційних

**Мета дипломної роботи** – дослідження і розробка інформаційної системи для автоматизації АЗС.

**Метод дослідження** – дослідити предметну область; виконати аналіз існуючих варіантів розв'язання досліджуваної задачі; виконати аналіз методів розв'язання задачі; виконати аналіз систем управління базами даних; розробити базу даних; розробити інформаційну систему.

**Наукова новизна одержаних результатів.** Застосування алгоритмів аналізу великих даних дозволяє системі надавати корисні інсайти та прогнозувати попит на пальне, що є критичним для оптимізації управління запасами та забезпеченням популярних продуктів. Використання методів моделювання та оптимізації бізнес-процесів може дозволити системі працювати більш ефективно, швидко адаптуючись до змінних умов ринку.

**Практичне значення одержаних результатів.** Полягає в можливості використання результатів дослідження та розробки застосунку у легшому способі автоматизації процесів АЗС.

**Особистий внесок випускника.** Розробка програмної частини інформаційної системи, яка включає в себе модулі для управління заправочними точками, складським обліком, обліком продажів, інвентаризацією тощо. Забезпечення того, щоб розроблена система не лише відповідала технічним вимогам, але й враховувала бізнес-потреби АЗС.

# ABSTRACT

Explanatory note to the diploma project "System for automating gas station operation": 53 pages, 9 figures, 0 tables, 29 used sources, 1 appendix.

INFORMATION SYSTEM, SOFTWARE PRODUCT, GAS STATION, INTELLIGENT TECHNOLOGIES, AUTOMATION

**The object of research** is methods and means of information development

**The aim of the thesis** is research and development of an information system for gas station automation.

**The research method** is to investigate the subject area; perform an analysis of existing options for solving the problem under study; perform an analysis of problem solving methods; perform an analysis of database management systems; develop a database; to develop an information system.

**Scientific novelty of the obtained results.** The application of big data analysis algorithms allows the system to provide useful insights and predict fuel demand, which is critical for optimizing inventory management and providing popular products. The use of methods of modeling and optimization of business processes can allow the system to work more efficiently, quickly adapting to changing market conditions.

**Practical significance of the obtained results.** It consists in the possibility of using the results of research and application development in an easier way to automate gas station processes.

**Personal contribution of the graduate.** Development of the software part of the information system, which includes modules for managing gas stations, warehouse accounting, sales accounting, inventory, etc. Ensuring that the developed system not only meets the technical requirements, but also takes into account the business needs of the gas station.

**ЗМІСТ**

# ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

БД – Бази Даних

ПП – Програмний Продукт

СКБД – Система Керування Базами Даних

ТЗ – Технічне Завдання

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

IT – Information Technology

JS –JavaScript

MySQL – Structured Query Language

PHP – Hypertext Preprocessor

URL – Uniform Resource Locator

# ВСТУП

Актуальність теми. На сьогоднішній день розробка інформаційних систем для автоматизації АЗС (автозаправних станцій) залишається дуже актуальною і важливою.

Інформаційні системи дозволяють автоматизувати облік продажу пального, що підвищує точність та надійність даних про обсяги продажу. Системи можуть автоматично вести облік залишків пального, що полегшує виконання інвентаризації та зменшує ймовірність помилок. Інформаційні системи дозволяють налаштовувати системи контролю доступу, що забезпечує безпеку управління станцією та обмежує несанкціонований доступ. Можливість моніторингу всіх операцій в режимі реального часу дозволяє вчасно виявляти аномалії або підозрілі дії.

Автоматизація дозволяє оптимізувати робочі процеси, що може призводити до зменшення витрат на оплату праці та зниження втрат через людські помилки. Системи аналізу даних можуть допомагати в прогнозуванні попиту на пальне, що дозволяє ефективно управляти запасами та уникнути зайвих витрат. Забезпечення взаємодії інформаційної системи АЗС з іншими бізнес-системами дозволяє підприємству оперативно та точно управляти фінансами та ресурсами. Системи можуть автоматично формувати звіти та забезпечувати відповідність законодавчим вимогам та стандартам галузі.

З урахуванням швидкого темпу технологічного розвитку та зростання вимог до оптимізації бізнес-процесів, розробка інформаційних систем для автоматизації АЗС залишається важливим напрямком для підприємств у сфері пального.

Метою роботи є дослідження і розробка інформаційної системи для автоматизації АЗС.

Вимоги до функціоналу інформаційної системи що будується:

- Обирати колонку;
- Обирати тип палива;
- Обирати кількість літрів;
- Обирати тип оплати;

- Формування чеку;
- Меню додаткових послуг;
- Можливість придбати паливо дистанційно;
- Простий та зрозумілий інтерфейс;
- Можливість перегляду локалізації АЗС;
- Формування звіту по замовленню для підтвердження;
- Формування QR-коду транзакції для миттєвої оплати;
- Видача решти на банківську картку або на рахунок користувача. Вимоги до надійності:

В процесі проектування повинні бути використані по можливості типові проектні рішення. На стадії технічного проектування повинні прийматися рішення, що зменшують спростять експлуатації системи для користувача.

Умови експлуатації:

Розроблюване програмне забезпечення буде створено за допомогою мов програмування php, js та мов розмітки html, css.

Об’єктом дослідження є методи і засоби розробки інформаційних систем.

Предметом розробки є процес розробки інформаційних систем для автоматизації АЗС.

Наукова новизна одержаних результатів. Застосування алгоритмів аналізу великих даних дозволяє системі надавати корисні інсайти та прогнозувати попит на пальне, що є критичним для оптимізації управління запасами та забезпеченням популярних продуктів. Використання методів моделювання та оптимізації бізнес-процесів може дозволити системі працювати більш ефективно, швидко адаптуючись до змінних умов ринку.

Практичне значення одержаних результатів. Полягає в можливості використання результатів дослідження та розробки застосунку у легшому способі автоматизації процесів АЗС.

# РОЗДІЛ 1
# АНАЛІЗ РОЗРОБКИ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ АЗС

## 1.1 Аналіз історії розвитку системи автоматизації роботи АЗС

У цьому розділі дипломної роботи проводиться докладний аналіз системи автоматизації роботи АЗС (автомобільних заправних станцій). Системи автоматизації вже давно є важливою частиною управління заправними станціями, і їх значення постійно зростає в контексті розвитку технологій та вимог ринку.

Історія розвитку систем автоматизації АЗС досить цікава та пов’язана зі стрімкими змінами технологій та вимог ринку. Нижче наведено докладний огляд історії розвитку цих систем:

- Ранні заправні станції без автоматизації (початок 20 століття)

Перші заправні станції, що з’явилися на початку 20 століття, мали ручне управління. Це означало, що співробітникам доводилося вручну відмірювати та заливати паливо, а також стежити за кількістю залитого палива. Цей процес був недосконалим і збільшував ризик помилок та шахрайства.

- Електронна революція (1960-1970-ті роки):

У 1960-1970-ті роки заправні станції почали переходити на електронну автоматику. Спеціальні пальномірники були оснащені електронікою, яка автоматично вимірювала кількість палива та передавала цю інформацію до системи обліку. Це суттєво підвищило точність обліку та відбилося на управлінні АЗС.

- Використання карткових систем обліку (1980-1990-ті роки).

У 1980-х та 1990-х роках з’явилися карткові системи обліку, які дозволяли клієнтам розплачуватися кредитними чи дебетовими картками. Це принесло зручність та безпеку в оплату заправок та сприяло відмові від готівки.

- Розвиток систем автоматизації АЗС (2000-тепер):

У 21 столітті системи автоматизації АЗС стали ще більш досконалими та функціональними. До їх складу входять різні компоненти, такі як системи контролю запасів палива, системи безпеки, системи відеоспостереження, а також програмне забезпечення для обліку операцій та аналізу даних. Одним із ключових трендів є

використання сучасних технологій для моніторингу та управління обладнанням АЗС у режимі реального часу.

- Зелена автоматизація (сучасність).

Сучасні системи автоматизації АЗС також активно реалізують екологічні рішення. Це включає використання біопалива, сонячних батарей, переробку відходів і скорочення викидів CO2. Такі заходи спрямовані на зниження впливу АЗС на довкілля.

- Перспективи у майбутнє.

Майбутнє систем автоматизації АЗС може бути пов’язане з розвитком автономних транспортних засобів (автомобілів) та більшою інтеграцією з мобільними мережами для забезпечення швидкого та зручного обслуговування клієнтів.

Підсумовуючи, історія розвитку систем автоматизації АЗС відображає тенденцію до поступового впровадження технологій, що полегшують роботу та покращують обслуговування клієнтів на АЗС.

Як наслідок, для вивчення кращого способу автоматизації системи АЗС, слід розглянути її основні компоненти:

- Система контролю і обліку пального, яка включає в себе датчики виміру пального, системи обліку і виведення інформації на екран.
- Система оплати інформації: Для зручності клієнтів на АЗС використовуються різні методи оплати, такі як готівка, кредитні картки, мобільні платежі.
- Системи безпеки: Охорона роботи АЗС важлива як для співробітників, так і для клієнтів. Використання камер спостереження, сигналізації і контролю доступу допомагає забезпечити безпеку на АЗС.

Системи автоматизації АЗС (автомобільних заправних станцій) мають свої переваги і недоліки. Давайте розглянемо їх детально:

- Переваги систем автоматизації АЗС:

1) Підвищена точність обліку пального: системи автоматизації дозволяють з високою точністю виміряти кількість заправленого пального, що допомагає

уникнути помилок в обліку та втрат пального. Це важливо для якості обліку та управління запасами.

2) Ефективне управління АЗС: автоматизація систем дозволяє відстежувати рівень запасів пального, робити замовлення для поповнення запасів і планувати обслуговування обладнання. Це сприяє зниженню часу простою обладнання і покращує продуктивність АЗС.

3) Зручність для клієнтів: системи автоматизації роблять оплату для клієнтів зручнішою. Їх можна оплатити готівкою, кредитною або дебетовою карткою, а також за допомогою мобільних платіжних систем, що в кілька разів спрощує платіжну систему, дає клієнтам вибір більш зручного способу оплати для них та приваблює швидкістю проведення платіжних операцій.

4) Покращення безпеки: автоматизовані системи безпеки, такі як камери спостереження, сигналізація і контроль доступу – також є частиною систем автоматизації, які в свою чергу допомагають забезпечити безпеку на АЗС для співробітників і клієнтів.

5) Віддалений моніторинг і управління: сучасні системи дозволяють власникам АЗС віддалено моніторити та керувати операціями станції через Інтернет. Це забезпечує більшу гнучкість та контроль над бізнесом.

- Недоліки систем автоматизації АЗС:

1) Високі витрати на впровадження: початкові інвестиції у системи автоматизації можуть бути значними, особливо для невеликих заправних станцій. Сюди входять витрати на апаратне забезпечення, програмне забезпечення та системну інтеграцію. Потреба в технічній підтримці: автоматизовані системи вимагають регулярної технічної підтримки і обслуговування, щоб забезпечити їх правильну роботу. Це може бути витратним та часомістким процесом.

2) Ризик кібератак: оскільки системи автоматизації АЗС підключені до Інтернету, вони стають потенційною метою кібератак. Недостатня кібербезпека може призвести до витоку даних та втрати контролю над системами. Залежність від технологій: висока залежність від технологій може створювати проблеми у разі

виходу з ладу обладнання або програмного забезпечення. Завжди існує ризик втрати доступу до системи.

3) Потенційний вплив на робочі місця: Автоматизація може призвести до зниження потреби в робочій силі, оскільки деякі процеси автоматизовані. Це може мати соціальні наслідки для працівників АЗС.

## 1.2 Аналіз актуальності розробки системи автоматизації роботи АЗС

Аналіз актуальності розробки системи автоматизації роботи АЗС є ключовим етапом дослідження, оскільки він визначає, наскільки важливо та доцільно впроваджувати дану технологію в сучасному світі. У цьому розділі ми проаналізуємо фактори, які роблять цю розробку актуальною, а також її потенційні переваги для АЗС та галузі в цілому.

Для початку, важливо розглянути актуальні тенденції у галузі нафтової промисловості. Сучасний ринок нафти та пального піддається впливу наступних факторів:

- Збільшення об’ємів виробництва та постачання пального: Попит на пальне не зменшується, і нафтова промисловість продовжує збільшувати об’єми видобутку та постачання нафти та газу.
- Зростаюча конкуренція: Галузь стикається зі зростаючою конкуренцією як на світовому, так і на регіональному ринку. Автоматизація може допомогти знизити витрати та підвищити конкурентоспроможність АЗС.
- Екологічні вимоги: Зростаюча увага до проблем екології та позитивний вплив на навколишнє середовище вимагають впровадження більш екологічних підходів до роботи АЗС, що може бути досягнуто завдяки автоматизації.

Технологічний прогрес розвивається швидко, і це створює нові можливості для автоматизації АЗС. Досягнення в галузі сенсорів, Інтернету речей (IoT), штучного інтелекту (AI) і аналітики даних можуть бути використані для створення більш ефективних та продуктивних систем автоматизації.

Споживчі вимоги та зручність клієнтів є ключовими аспектами при розгляді актуальності розробки систем автоматизації на автомобільних заправках.

Сьогоднішні споживачі покладають великі надії на обслуговування заправних станцій та очікують, що ця послуга буде швидкою, ефективною та зручною.

По-перше, споживачі очікують швидкого та ефективного обслуговування. Вони цінують свій час і не хочуть витрачати його на довгі черги або затримки під час заправки. Система автоматизації дозволяє скоротити час очікування, прискорити процес заправки та оплати, адже практично не використовує людський фактор, який негативно впливає на швидкість, а також, не менш важливо, дозволяє зменшити ймовірність здійснення людської помилки, адже система повністю автоматизована. Проте, не варто забувати, що вона також може підводити в роботі, як і людський фактор.

По-друге, для клієнтів важлива точність обліку палива. Вони розраховують, що кількість заправленого палива вимірюватиметься з високою точністю та без помилок. Системи автоматизації забезпечують точний облік та калькуляцію витрат.

По-третє, зручність для користувача повинна включати різні функції, такі як можливість налаштування улюблених варіантів заправки, доступ до історії заправок і рекламним акціям, що дозволяють клієнтам насолоджуватися персоналізованим обслуговуванням.

По-четверте, важливими аспектами є безпека та захист даних клієнтів. Клієнти довіряють АЗС свої особисті та фінансові дані, тому важливо, щоб системи автоматизації мали високі стандарти безпеки та захисту даних, забезпечуючи конфіденційність та захист інформації про клієнтів.

Підсумовуючи, споживчі вимоги та зручність для клієнтів визначають актуальність розробки систем автоматизації на АЗС. Забезпечення швидкого, ефективного та зручного обслуговування, різних способів оплати, точного обліку, зручних функцій та безпеки даних створює сприятливі умови для залучення та утримання клієнтів, що робить дані системи важливими для успішної роботи АЗС.

Автоматизація дозволяє оптимізувати використання ресурсів на АЗС, таких як паливо, робоча сила та енергія. Це може призвести до зниження витрат та збільшення рентабельності.

Ефективність керування ресурсами – одна з ключових переваг систем

автоматизації на автомобільних заправках (АЗС). Цей аспект важливий з економічної та екологічної точки зору і може суттєво вплинути на прибутковість та стійкість роботи АЗС. Давайте докладніше розглянемо, як системи автоматизації допомагають ефективно управляти ресурсами:

- Управління запасами пального:

Системи автоматизації дозволяють АЗС точно відстежувати обсяги пального на станції. Це дозволяє вчасно робити замовлення на постачання пального, уникнути його надмірної запасу або нестачі. Оптимальний рівень запасу пального допомагає знизити витрати на зберігання і уникнути втрат через зміну якості пального внаслідок тривалого зберігання.

- Моніторинг та управління робочою силою:

Системи автоматизації допомагають керувати робочою силою більш ефективно. Це включає в себе планування графіків роботи, відстеження робочого часу та реєстрацію відомостей про відпустки та відпрацьовані години. Це допомагає уникнути переплати за працю та забезпечити належний рівень обслуговування клієнтів.

- Оптимізація витрат енергії:

Автоматизовані системи можуть допомагати контролювати та оптимізувати споживання енергії на станції. Наприклад, вони можуть автоматично вимикати освітлення та обладнання в нічний час або під час неактивності. Це допомагає знизити рахунки за комунальні послуги і зменшує вплив АЗС на навколишнє середовище.

- Оптимізація робочих процесів:

Системи автоматизації дозволяють автоматизувати багато рутинних операцій на АЗС, таких як облік продажів, розрахунок заправленого пального, обслуговування клієнтів тощо. Це допомагає покращити продуктивність персоналу і зменшити ризик помилок.

- Моніторинг витрат і прибутковості:

Системи автоматизації забезпечують детальний аналіз фінансових показників АЗС, що дозволяє власникам зрозуміти, як використовуються ресурси і де можливо

здійснити заходи щодо підвищення прибутковості.

### 1.3 Потреба автоматизації роботи АЗС для визначення актуальності розробки даної системи

У цьому підрозділі ми розглянемо необхідність автоматизації роботи автомобільних заправок (АЗС) та визначимо актуальність розробки систем автоматизації для даного сегмента галузі. Розглянемо фактори, що призводять до необхідності автоматизації, а також можливі переваги та проблеми, пов’язані із впровадженням даних систем.

У зв’язку із зростанням автопарку та збільшенням кількості автовласників попит на паливо на АЗС продовжує зростати. У той же час, конкуренція серед заправок зростає. Щоб залучити та утримати клієнтів, заправні станції повинні надавати швидке та ефективне обслуговування. Системи автоматизації допомагають зробити це, автоматизуючи багато процесів, що скорочує час обслуговування та підвищує задоволеність клієнтів.

Заправні станції повинні стежити за запасами палива. З точки зору бізнесу важливо мати оптимальний рівень запасів для задоволення попиту клієнтів, але не мати надмірних запасів, які можуть призвести до додаткових витрат на зберігання та зниження прибутковості. Системи автоматизації дозволяють точно відстежувати рівень запасів та автоматично замовляти паливо за потреби.

Крадіжка палива та інші види шахрайства можуть призвести до серйозних фінансових втрат для АЗС. До автоматизованих систем належать системи відеоспостереження, контролю доступу та моніторингу, які допомагають запобігати та виявляти несанкціоновані дії.

Ефективне керування персоналом є важливим аспектом ефективної роботи АЗС. Системи автоматизації допомагають оптимізувати графіки роботи, відстежувати робочий час, планувати відпустки та реагувати на надзвичайні ситуації. Це дозволяє знизити трудовитрати та підвищити продуктивність.

Зростання уваги до екологічних проблем та регулювання скорочення викидів вуглекислого газу призводить до необхідності впровадження більш екологічних

рішень на заправних станціях. Автоматизація може допомогти оптимізувати використання ресурсів та знизити вплив заправних станцій на довкілля.

**Висновок**

У сучасному світі, коли технічний прогрес розвивається неконтрольовано, системи автоматизації автомобільних заправок стають невід'ємною частиною їхньої роботи. У ході нашого дослідження ми ретельно розглянули всі аспекти автоматизації АЗС, включаючи історію її розвитку, переваги та недоліки, актуальність та споживчі вимоги до клієнтів. На закінчення хотілося б підкреслити ключові моменти, які виділяють автоматизацію АЗС як важливий та актуальний напрямок.

Почавши з історії розвитку систем автоматизації АЗС, бачимо, що ця ідея постійно розвивалася і адаптувалася до змін вимог ринку та технологічних можливостей. Від перших механічних пристроїв до сучасних інтегрованих систем автоматизація АЗС стала важливим чинником підвищення ефективності та конкурентоспроможності станцій.

Докладно вивчені переваги та недоліки систем автоматизації. З одного боку, ці системи допомагають знизити витрати, підвищити точність обліку, забезпечити клієнтам зручність та швидкість обслуговування. З іншого боку, вони можуть стати вразливими для кібератак та вимагати високих витрат на впровадження та обслуговування. Однак, на наш погляд, переваги набагато переважують недоліки, правильно інтегрувавши такі системи, АЗС можуть покращити якість обслуговування та стати більш конкурентоспроможними.

Актуальність розробки систем автоматизації автозаправних станцій також незаперечна. Постійне зростання попиту на пальне, ринкова конкуренція, необхідність управління ресурсами та вимоги споживачів до зручності клієнтів роблять автоматизацію необхідністю. Впровадження таких систем допомагає АЗС стати більш ефективними, знижує ризики та сприяє їхній стабільності та успіху.

Зрештою, в основі сучасної АЗС лежать споживчі вимоги та зручність для клієнтів. Швидкість обслуговування, різні способи оплати, точність обліку,

зручність для користувачів та безпека даних – усі ці фактори роблять обґрунтованим рішення щодо впровадження систем автоматизації на АЗС. Забезпечуючи зручність та задоволеність клієнтів, заправні станції можуть покращити свою репутацію та залучити більше клієнтів.

На закінчення відзначимо, що системи автоматизації на АЗС – важливий та актуальний напрямок розвитку. Вони допомагають заправним станціям підвищити ефективність, конкурентоспроможність та задоволеність клієнтів. Основними перевагами є підвищена точність та швидкість обслуговування, а також забезпечення безпеки та захисту даних. Ці системи стають важливим інструментом для заправних станцій у досягненні успіху та стійкості на паливному ринку.

# РОЗДІЛ 2
# ПРОЕКТУВАННЯ ІС ДЛЯ КОЛОНКИ САМООБСЛУГОВУВАННЯ НА АЗС

## 2.1 Існуючі рішення та недоліки для побудови інформаційної системи для терміналу колонки самообслуговування АЗС

На сьогоднішній день існує кілька інформаційних систем для терміналів колонок самообслуговування на автозаправних станціях (АЗС). Ці системи можуть включати в себе різноманітні функції, такі як автоматизація оплати пального, моніторинг запасів, взаємодія з клієнтами та багато іншого. Нижче подано кілька прикладів існуючих рішень:

1. Wayne iX Pay:

Опис:

Wayne iX Pay є рішенням для автозаправних станцій від компанії Wayne Fueling Systems, яке надає повний набір функцій для терміналів самообслуговування. Система включає в себе можливість оплати через різні канали, взаємодію з клієнтами та моніторинг операцій.

2. Gilbarco Passport EDGE:

Опис:

Passport EDGE від Gilbarco Veeder-Root є інтегрованою системою для АЗС, яка має модуль самообслуговування. Вона дозволяє клієнтам здійснювати оплату, вибирати тип пального та використовувати різні форми платежів.

3. NCR SelfServ Petrol:

Опис:

NCR SelfServ Petrol - це рішення для терміналів самообслуговування від NCR. Воно надає можливість клієнтам вибирати продукти, сплачувати за них та отримувати квитанції. Система також дозволяє відстежувати та аналізувати покупки.

4. VeriFone Commander Site Controller:

Опис:

Commander Site Controller від VeriFone - це інтегрована система управління АЗС, яка включає в себе модуль самообслуговування. Клієнти можуть вибирати пальне, здійснювати оплату та отримувати квитанції.

5. Dover Fueling Solutions (DFS) Tokheim Crypto VGA:

Опис:

DFS Tokheim Crypto VGA - це рішення для АЗС від Dover Fueling Solutions. Воно дозволяє автоматизувати оплату, моніторинг запасів та взаємодію з клієнтами через термінали самообслуговування.

## 2.2 Постановка задачі розробки

Вимоги до функціоналу інформаційної системи що будується:

- Обирати колонку;
- Обирати тип палива;
- Обирати кількість літрів;
- Обирати тип оплати;
- Формування чеку;
- Меню додаткових послуг;
- Можливість придбати паливо дистанційно;
- Простий та зрозумілий інтерфейс;
- Можливість перегляду локалізації АЗС;
- Формування звіту по замовленню для підтвердження;
- Формування QR-коду транзакції для миттєвої оплати;
- Видача решти на банківську картку або на рахунок користувача. Вимоги до надійності:

В процесі проектування повинні бути використані по можливості типові проектні рішення там. На стадії технічного проектування повинні прийматися рішення, що зменшують спростять експлуатації системи для користувача.

Умови експлуатації:

Розроблюване програмне забезпечення буде створено за допомогою мов програмування php, js та мов розмітки html, css.

## 2.3 Архітектура програмного застосунку

Концептуальна модель системи представлена діаграмами випадків за допомогою UML (Unified Modeling Language).

В UML діаграми випадків моделюють поведінку системи та допомагають визначити системні вимоги.

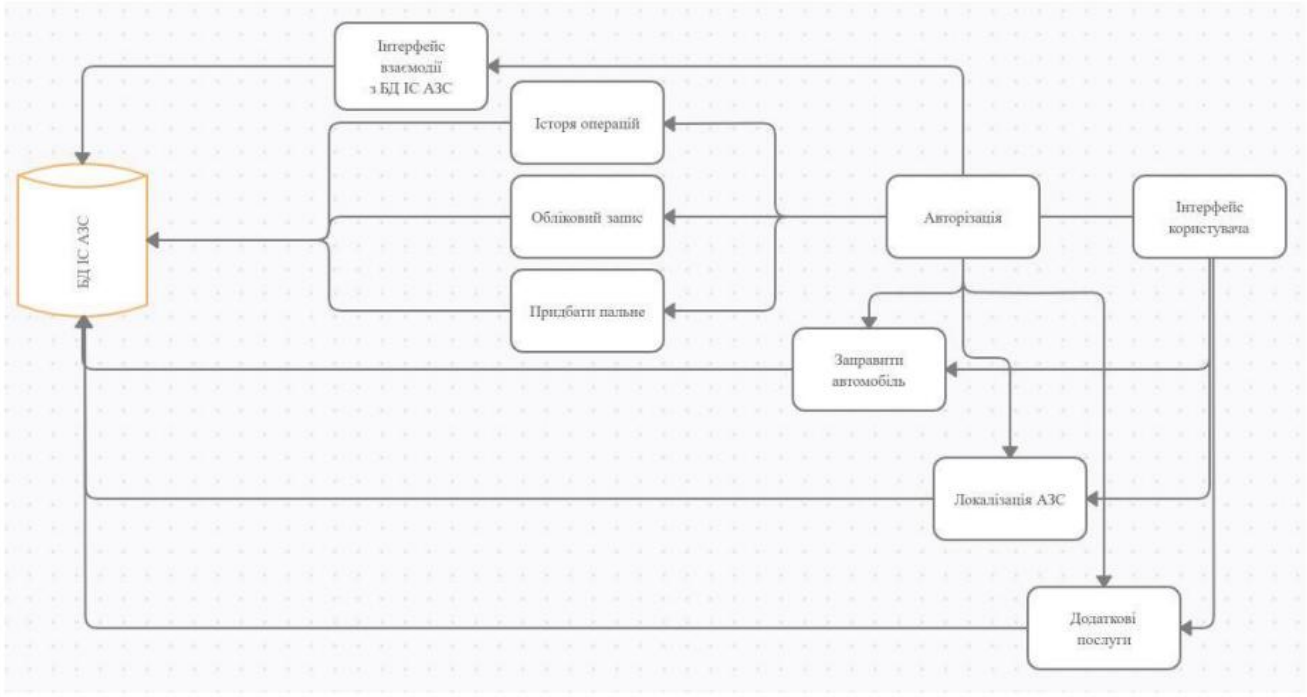


Рис. 2.1. Схема програмного застосунку

Діаграми випадків описують функціональні можливості високого рівня та обсяг системи. Ці діаграми також визначають взаємодію між системою та її акторами. Варіанти використання та актори на діаграмі варіантів використання описують, що робить система та як актори її використовують, але не те, як система працює всередині.

Діаграма випадку ілюструє та визначає контекст і вимоги всієї системи або значної частини системи. Ви можете використовувати діаграму одного випадку для моделювання складної системи або створити багато діаграм для моделювання компонентів системи. Діаграми прецедентів зазвичай розробляються на початку проекту та посилаються на них протягом усього процесу розробки.

Діаграми варіантів використання корисні, коли:

- перед початком проекту необхідно створити діаграму випадку для бізнес-моделювання, щоб усі учасники проекту мали розуміння співробітників, клієнтів і корпоративної діяльності;

- збираючи вимоги, ви можете створювати діаграми випадків, щоб відобразити системні вимоги та показати іншим, що повинна робити система;

- на етапах аналізу та проектування варіанти використання та актори на діаграмі випадку можуть бути використані для визначення класів, необхідних системі;

- під час фази тестування діаграми варіантів використання можуть бути використані для визначення системних тестів [13].

Учасники системи користувач мають можливість генерувати секретні ключі зв'язку, підключатися до своїх співрозмовників і обмінюватися повідомленнями в зашифрованому вигляді.

Діаграма варіантів використання (use case diagram) є одним із видів діаграм UML (Unified Modeling Language), який використовується для моделювання взаємодії між різними суб'єктами (користувачами або системами) та їх варіантами використання в конкретному контексті.

Основні елементи діаграми варіантів використання включають:

Актори: Це зовнішні сутності, такі як користувачі або інші системи, які взаємодіють з системою.

Варіанти використання (use cases): Це конкретні функціональні можливості або послуги, які система надає своїм акторам.

Відносини між акторами та варіантами використання: Вказують, як актори взаємодіють із системою та один з одним через варіанти використання.

Процес створення діаграми варіантів використання може бути наступним:

Визначення акторів: Визначте всі зовнішні сутності, які будуть взаємодіяти з системою.

Визначення варіантів використання: Ідентифікуйте всі можливі варіанти використання системи або певного компонента.

Створення діаграми: Розмістіть акторів та варіанти використання на діаграмі, позначте їх відносини.

Додавання асоціацій: Вкажіть, які актори мають доступ до конкретних варіантів використання.

Додавання розширень та включень: Якщо є варіанти використання, які можуть бути розширені або включені в інші, вказуйте ці відносини.

Аналіз та вдосконалення: Проаналізуйте діаграму, можливо, вносячи зміни або вдосконалюючи її.

Діаграма варіантів використання допомагає команді розробників та зацікавленим сторонам зрозуміти, як користувачі будуть взаємодіяти з системою та які функціональні можливості вона надає (рис. 2.2).

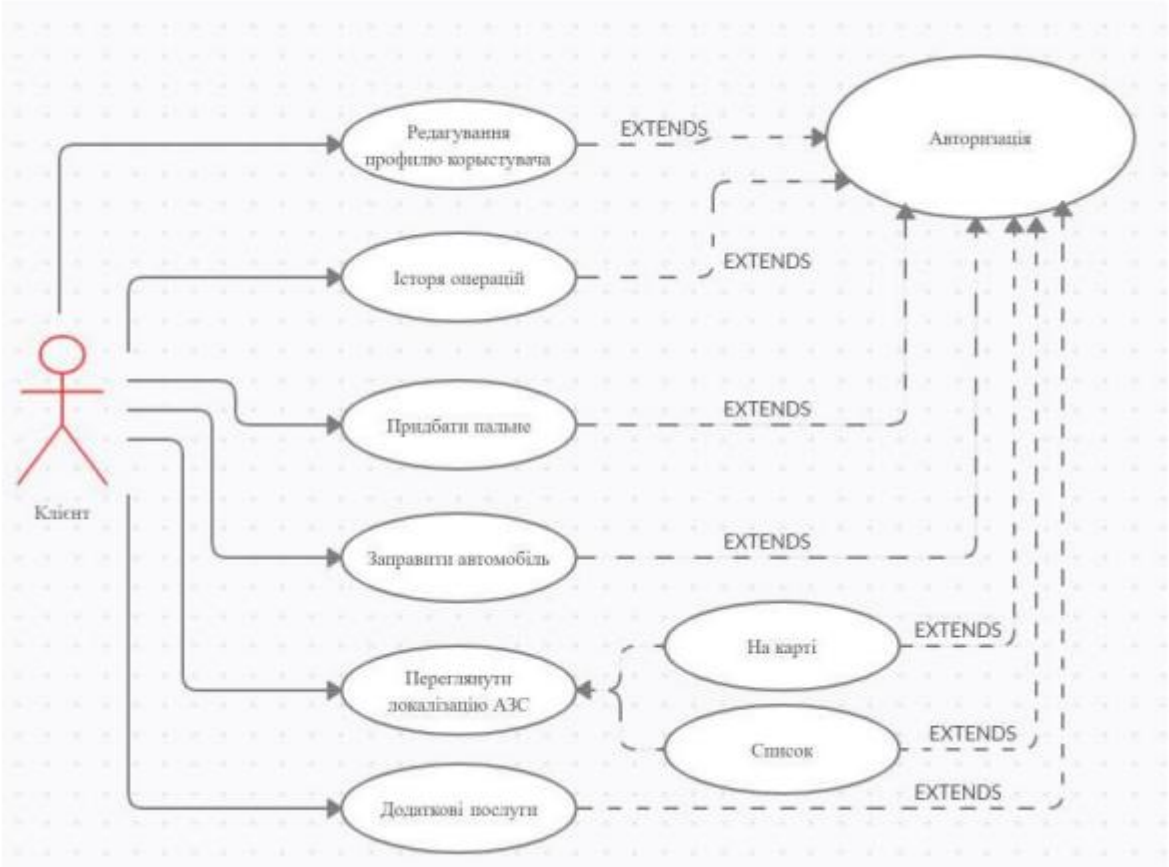


Рис. 2.2. Діаграма варіантів використання

Діаграма компонентів представлена на рис. 2.3.

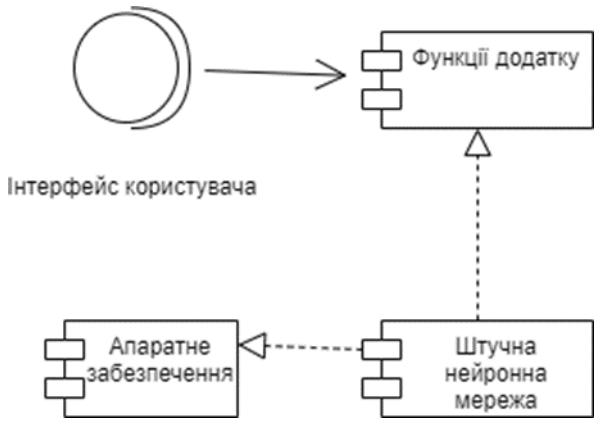


Рис. 2.3. Діаграма компонентів системи

Інтерфейс користувача потрібен для взаємодії між користувачем та ПЗ.

До функцій додатку відносяться процес шифрування, розшифрування та надсилання і отримання повідомлень.

Апаратне забезпечення складається з комп’ютерних комплектуючих системи користувача.

ШНМ являє собою головну функцію системи, а саме генерація ключа шифрування.

**Висновок**

В другому розділі було виконано проектування інформаційної системи для колонки самообслуговування, а саме розглянуті існуючи рішення на ринку програмного забезпечення та їх недоліки для виявлення вимог та побудови інформаційної системи для терміналу колонки самообслуговування АЗС. На основі отриманих даних була побудована схема концепції для створюваного програмного забезпечення, побудували діаграму використання та специфікацію варіантів використання провівши детальний опис кожного з варіантів. Кінцевою стадією проектування було будування концептуальної модель даних, яка дозволила описати концептуальні схеми за допомогою узагальнених конструкцій блоків і перейти до реалізації програмного забезпечення.

# РОЗДІЛ 3
# ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ АЗС

## 3.1 Вибір системи управління базами даних

Система управління базами даних (СУБД) є програмним забезпеченням, яке дозволяє зберігати, організовувати та управляти базами даних. Вона надає інтерфейс для взаємодії з базою даних, забезпечуючи можливість вставки, вилучення, оновлення та вибірки даних.

1. Oracle Database є однією з найбільш популярних та впливових комерційних систем управління базами даних (СУБД). Вона розроблена компанією Oracle Corporation і використовується в різноманітних індустріях та галузях для зберігання, управління та доступу до даних.

Основні характеристики Oracle Database включають:

Структура даних: Oracle підтримує реляційну модель даних, що дозволяє зберігати дані у вигляді таблиць з рядками та стовпцями. Також вона має можливості для розширення та використання об'єктно-орієнтованих підходів.

Мова запитів: Мова SQL (Structured Query Language) використовується для взаємодії з базою даних. Oracle також надає розширені можливості SQL, такі як аналітичні функції, об'єднання даних з різних джерел, та інші.

Масштабованість: Oracle Database підтримує великі обсяги даних і може бути масштабована від невеликих проектів до великих корпоративних систем.

Безпека: Oracle має різноманітні засоби для забезпечення безпеки даних, включаючи механізми автентифікації, авторизації, шифрування та аудиту.

Можливості резервного копіювання та відновлення: Система надає можливості для створення резервних копій бази даних та відновлення даних в разі втрати.

Транзакційна підтримка: Oracle дотримується принципів ACID (Atomicity, Consistency, Isolation, Durability) для забезпечення надійності та цілісності транзакцій.

Інтеграція та розширюваність: Oracle може інтегруватися з іншими системами та підтримує різноманітні інструменти для розширення функціональності.

Управління продуктивністю: Засоби для оптимізації та моніторингу продуктивності запитів та бази даних.

Oracle Database використовується в багатьох корпоративних середовищах, де вимагаються високі рівні надійності, продуктивності та безпеки. Вона також має різні версії та конфігурації для різних потреб, таких як Oracle Database Standard Edition, Enterprise Edition, та інші.

2. Microsoft SQL Server — це система управління базами даних (СУБД), розроблена компанією Microsoft. SQL Server надає ряд рішень для зберігання, управління та взаємодії з даними у середовищах Windows та інших операційних системах.

Основні характеристики Microsoft SQL Server включають:

Реляційна модель даних: SQL Server базується на реляційній моделі даних, де дані зберігаються у вигляді таблиць з рядками та стовпцями.

SQL мова запитів: SQL Server використовує мову запитів SQL для взаємодії з базою даних. Вона також має розширені можливості для роботи з процедурним SQL (T-SQL).

Інтеграція з іншими продуктами Microsoft: SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Microsoft Azure, Microsoft Excel, та інші.

Масштабованість: SQL Server може бути використаний в різних обсягах, від невеликих бізнес-проектів до великих корпоративних систем.

Механізми безпеки: Надає засоби для забезпечення безпеки даних, включаючи автентифікацію, авторизацію, та шифрування.

Оптимізація продуктивності: Має інструменти для оптимізації та моніторингу продуктивності запитів та бази даних.

Транзакційна підтримка: SQL Server дотримується принципів ACID для забезпечення цілісності та надійності транзакцій.

Аналітика та звітність: SQL Server має різні інструменти для аналізу даних, створення звітів, та візуалізації результатів.

Система резервного копіювання та відновлення: Надає можливості для створення резервних копій баз даних та відновлення даних.

SQL Server доступний в різних версіях, таких як SQL Server Express (безкоштовна версія для невеликих проектів), SQL Server Standard Edition та SQL Server Enterprise Edition для більших та корпоративних проектів.

Розроблена логічна модель бази даних, у середовищі програми SQL Server. Приведені сутності з атрибутами системи, розроблені на основі інформаційного забезпечення системи та з метою побудови логічної моделі.

Логічна модель бази даних системи представлена на рисунку 3.1.

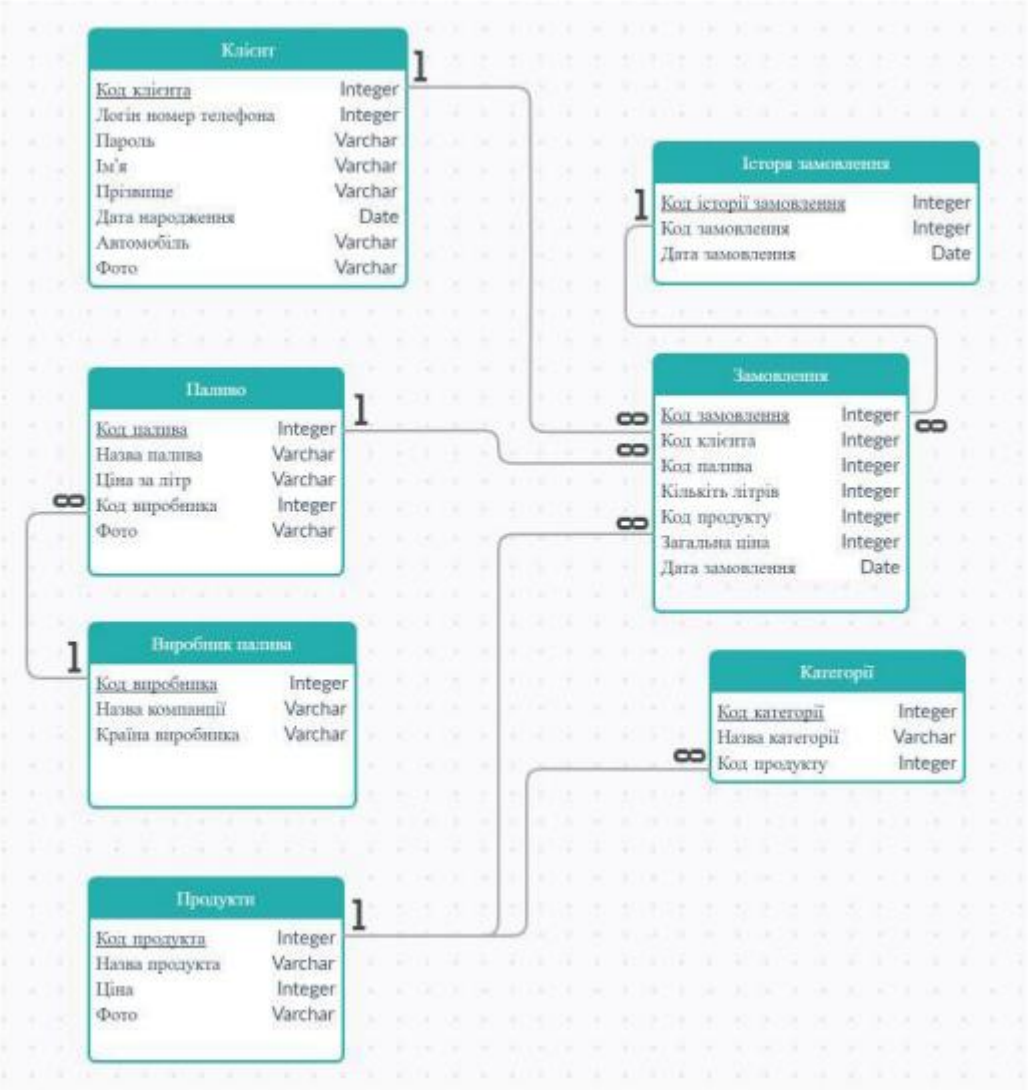


Рис. 3.1. Логічна модель бази даних

### 3.2 Рішення з програмного забезпечення

Для реалізації проекту дипломної роботи оптимально використовувати наступні мови програмування PHP та JavaScript мову розмітки HTML та мову опису зовнішнього вигляду документа, написаного з використанням мови розмітки CSS. Виходячи з цього середовищем програмування обрано Sublime Text. Сервером бази даних було обрано СУБД MySQL.

Опис використаних мов:

1) PHP (Hypertext Preprocessor) - це мова програмування, яка використовується для розробки веб-застосунків. Вона є широко використовуваною для створення динамічних веб-сайтів і забезпечує можливість взаємодії з базами даних, роботи з формами, обробки HTTP-запитів та іншими завданнями, пов'язаними із веб-розробкою.

2) HTML (Hypertext Markup Language) є стандартною мовою розмітки для створення та відображення веб-сторінок. HTML визначає структуру документа та елементи, які знаходяться на сторінці, такі як заголовки, абзаци, зображення, посилання і багато інших. HTML використовується для створення базової структури веб-сторінок та надання засобів для взаємодії з їхнім вмістом.

3) CSS (Cascading Style Sheets) є мовою стилів, яка використовується для оформлення та визначення зовнішнього вигляду веб-сторінок, написаних мовою розмітки, такою як HTML чи XML. CSS дозволяє визначати розміри, кольори, шрифти, розташування елементів та інші аспекти вигляду веб-сторінок. Основна ідея CSS полягає в тому, щоб відокремити структуру документа від його візуального представлення.

4) JavaScript є найбільш широко використовуваною мовою в області розробки в усьому світі. Вже більше 25 років JavaScript Frameworks керує Інтернетом. Але код JavaScript обережно оброблявся в фреймворках JavaScript принаймні протягом останнього десятиліття.

JavaScript є однією з найпопулярніших мов і в даний час є вибором як для фронтальної, так і для серверної веб-розробки. Крім того, у середині фреймворку JavaScript ви, ймовірно, виберете досить скоро.

JavaScript Frameworks: Frameworks надає базову платформу для створення програм JavaScript для розробників. Це заощаджує розробникам роботу, щоб почати з функціональної основи, щоб почати все з нуля. Ця база включає в себе набір бібліотек коду в екземплярі JavaScript. Бібліотеки компілюють код для конкретного типу програми, з якою ви можете працювати. Фреймворк по суті сам визначить структуру програми. У кожній структурі JavaScript використовується інша функція. Для веб-розробки JavaScript є надійним вибором, і багато його структур базуються на цьому підприємстві.

JavaScript — це проста об'єктно-керована мова програмування, яка використовується для написання сценаріїв веб-сторінок на кількох веб-сайтах. Це інтерпретація, всеосяжна мова програмування. При застосуванні до HTML-документа JavaScript забезпечує динамічну інтерактивність веб-сайту.

5) SQL – це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, обновлення, і вилучення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно.

### 3.3 Програмна реалізація інформаційної системи

Головна сторінка програмного забезпечення АЗС містить 4 основі модулі: «Авторизація», що є не обов’язковим кроком, «заправка автомобіля», «кафе» або ж додаткові послуги, «локалізація». Містить підменю на якому містяться переходи на підмодулі: «про нас», «Акції», «Оплатити пальне». Представлено на рисунку 3.2.

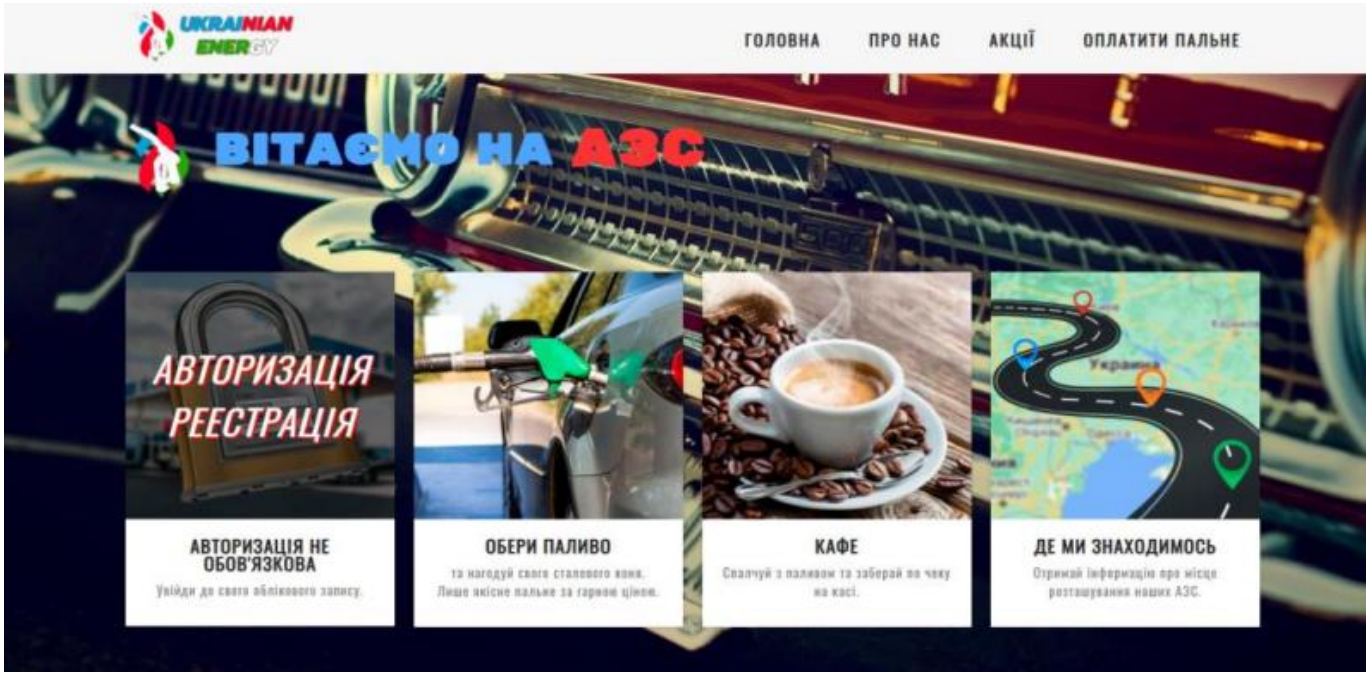


Рис. 3.2. Головна сторінка програмного забезпечення АЗС

Сторінка «Авторизація» містить на собі форму на якій містяться поля вводу логіна та пароля та 3 кнопки: «увійти», «реєстрація» та «повернутись» якщо користувач перейшов на авторизацію випадково. Представлено на рисунку 3.3.

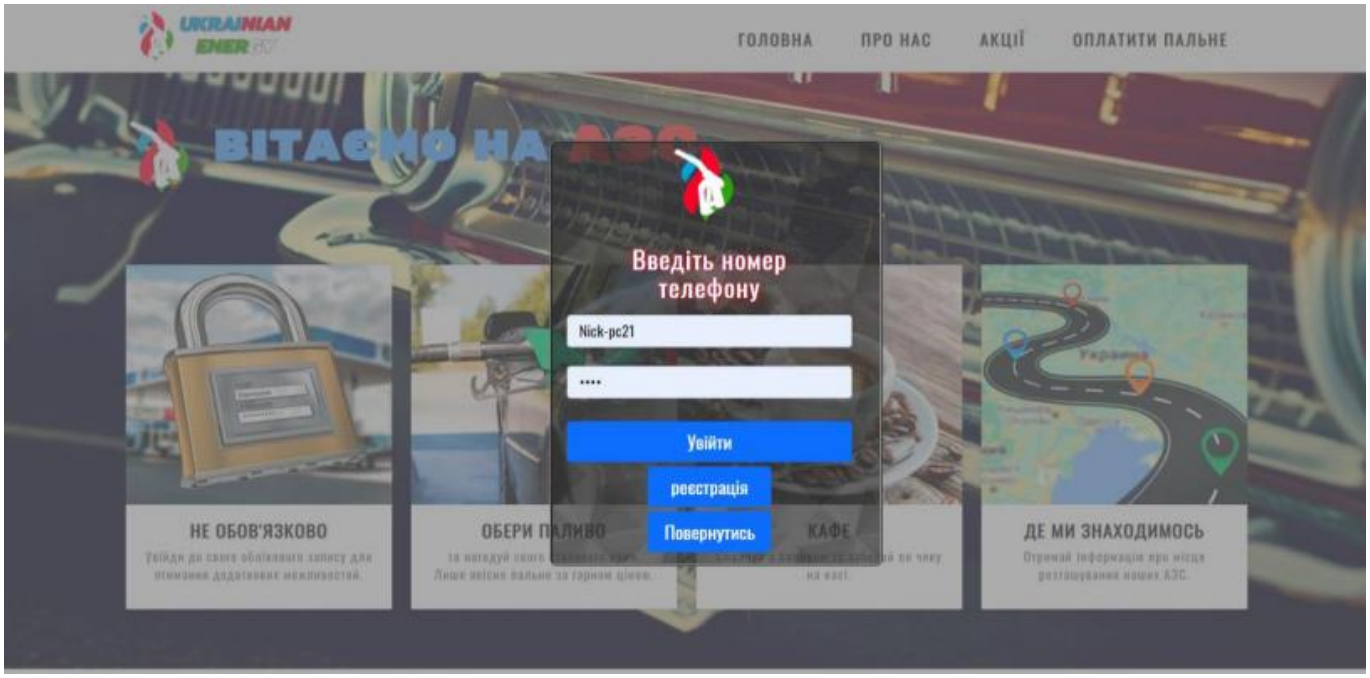


Рис. 3.3. Сторінка програмного забезпечення «Авторизації»

Якщо користувач захоче він має можливість створити обліковий запис для цього йому необхідно перейти на форму «Реєстрація» на якій містяться поля вводу для логіна (Номеру телефона), імені та пароля та 2 кнопки: «Реєстрація», «на головну». Представлено на рисунку рис. 3.4.

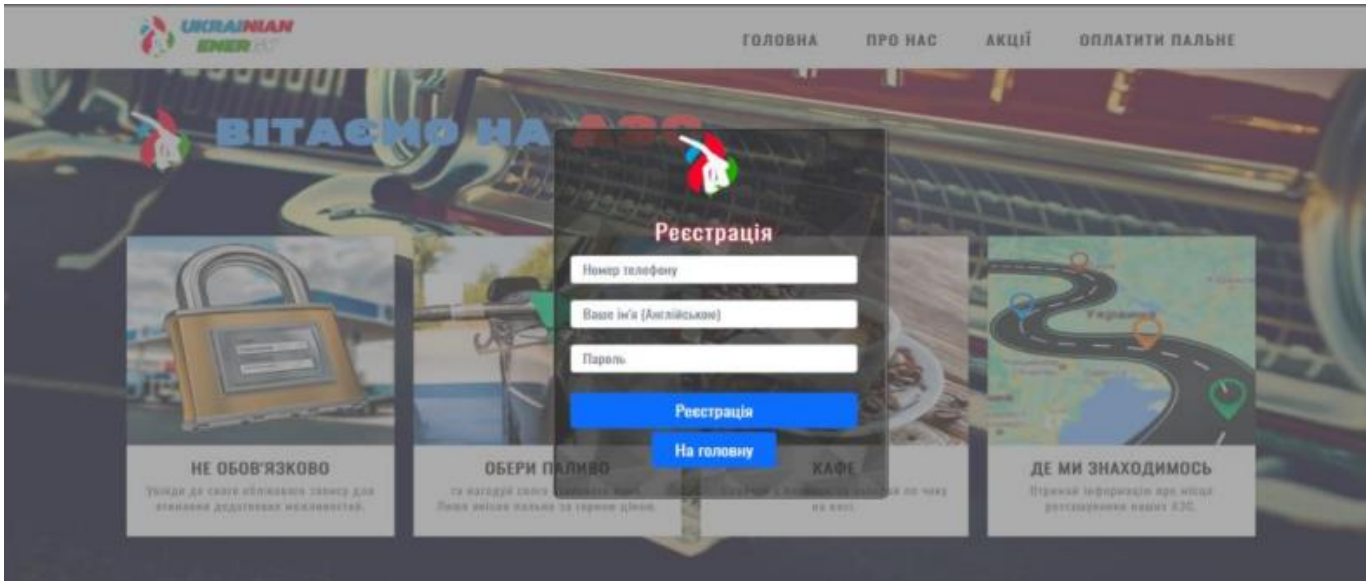


Рис. 3.4. Сторінка програмного забезпечення «Реєстрація»

Сторінка «Заправити автомобіль» дозволяє користувачеві виконати дії які сприяють початку заправки його автомобіля, він має вибрати тип бензину, що береться по конкретному id з серверу методом GET та вноситься в таблицю «замовлення» за допомогою методу POST. Представлено на рисунку 3.5.

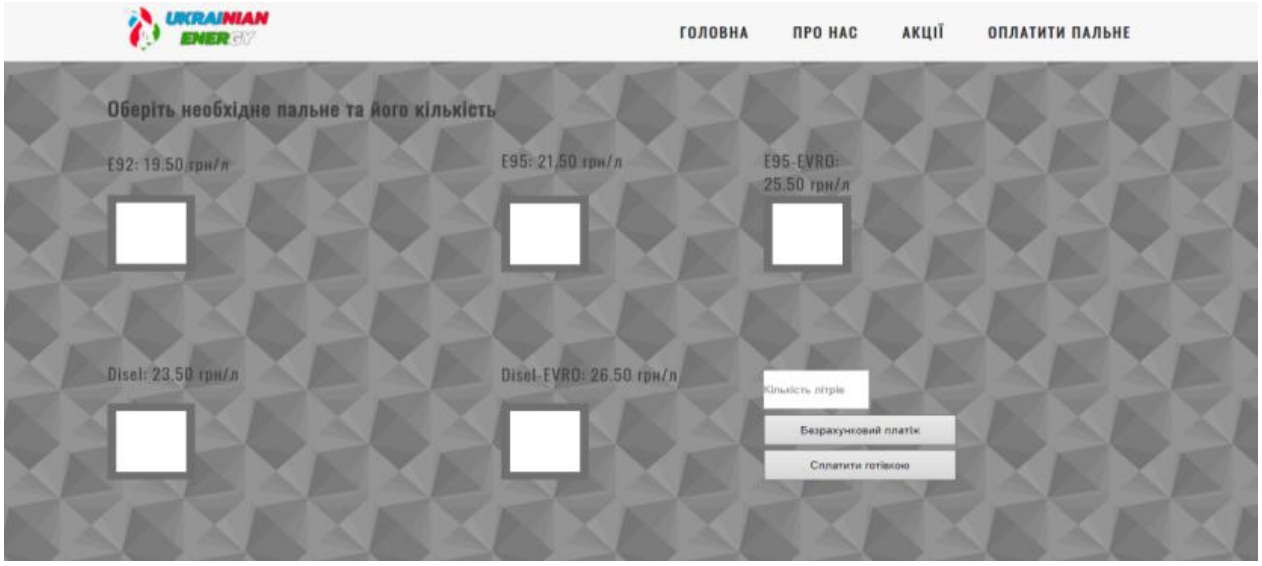


Рис. 3.5. Сторінка програмного забезпечення «Заправити автомобіль»

Так як це програмне забезпечення реалізовано в рамках магістерської дипломної роботи для прикладу було реалізовано «безрахунковий платіж» при натисненні на цю кнопку для клієнта генерується QR код транзакції, або ж клієнт має можливість виконати оплати через сервіс LiqPay зручним для себе банком. Представлено на рисунку 3.6.

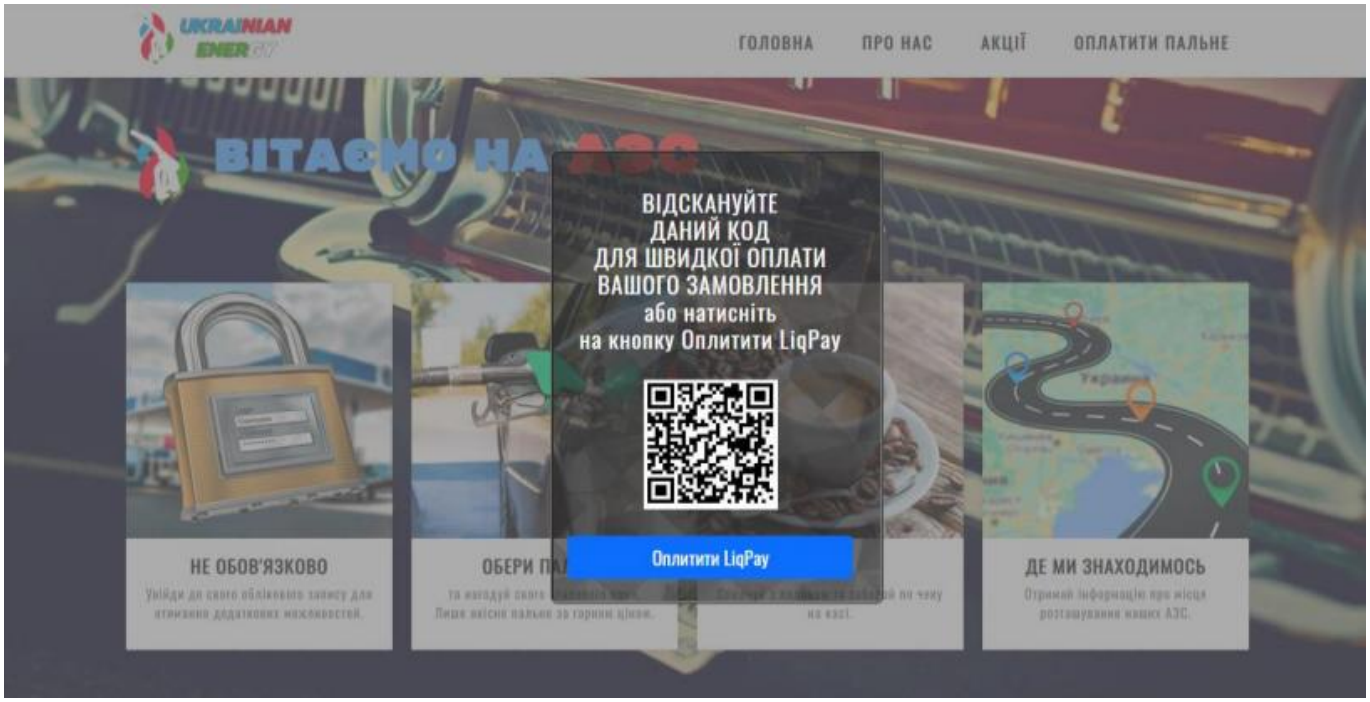


Рис. 3.6. Сторінка програмного забезпечення «Сплата замовлення»

**Висновок**

Реалізацію інформаційної системи для колонки самообслуговування було розпочато з побудови діаграми послідовності яка відображає взаємодію та порядок дій програмних модулів. Далі була побудована діаграма станів яка слугує для описання реакції системи на вплив ззовні та розглянута специфікація переходів станів. Була побудована логічна і фізична моделі бази даних, детальніше розглянута структура фізичної моделі. Були встановлені мінімальні системні вимоги для експлуатації інформаційної системи та прийняті рішення з програмного забезпечення для написання ІС. Кінцевим кроком було будування діаграми розгортання, що дозволило закінчити реалізацію ІС.

# РОЗДІЛ 4
# ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

− відповідність вимогам, якими керувалися проектувальники та розробники;

− правильна відповідь для усіх можливих вхідних даних;

− виконання функцій за прийнятний час;

− практичність;

− сумісність з програмним забезпеченням та операційними системами;

− відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення баґів (помилок або інших дефектів).

Тестування ПЗ може надавати об’єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників.

Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершено). Процес розробки зазвичай передбачає коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно [15].

Під час проведення тестування інформаційної системи для автоматизації АЗС в інтернеті виявлено, що застосунок відображається однаково в найбільш популярних браузерах, таких як Internet Explorer, Opera, Mozilla Firefox, Chrome.

Перевірка програми на юзабіліті показала, що програмний застосунок є зручним для користувача, має зручну навігаційну панель та інтуїтивно зрозумілу логічну структуру.

Під час перевірки не виявлено помилок коду.

**Висновки**

В даному розділі розглянута інструкція користувача. Також було проведено тестування інформаційної системи для автоматизації АЗС. В ході тестування жодних помилок не виявлено, програмний застосунок працює коректно.

# ВИСНОВКИ

У сучасному світі, коли технічний прогрес розвивається неконтрольовано, системи автоматизації автомобільних заправок стають невід'ємною частиною їхньої роботи. У ході нашого дослідження ми ретельно розглянули всі аспекти автоматизації АЗС, включаючи історію її розвитку, переваги та недоліки, актуальність та споживчі вимоги до клієнтів. На закінчення хотілося б підкреслити ключові моменти, які виділяють автоматизацію АЗС як важливий та актуальний напрямок.

Почавши з історії розвитку систем автоматизації АЗС, бачимо, що ця ідея постійно розвивалася і адаптувалася до змін вимог ринку та технологічних можливостей. Від перших механічних пристроїв до сучасних інтегрованих систем автоматизація АЗС стала важливим чинником підвищення ефективності та конкурентоспроможності станцій.

Докладно вивчені переваги та недоліки систем автоматизації. З одного боку, ці системи допомагають знизити витрати, підвищити точність обліку, забезпечити клієнтам зручність та швидкість обслуговування. З іншого боку, вони можуть стати вразливими для кібератак та вимагати високих витрат на впровадження та обслуговування. Однак, на наш погляд, переваги набагато переважують недоліки, правильно інтегрувавши такі системи, АЗС можуть покращити якість обслуговування та стати більш конкурентоспроможними.

Актуальність розробки систем автоматизації автозаправних станцій також незаперечна. Постійне зростання попиту на пальне, ринкова конкуренція, необхідність управління ресурсами та вимоги споживачів до зручності клієнтів роблять автоматизацію необхідністю. Впровадження таких систем допомагає АЗС стати більш ефективними, знижує ризики та сприяє їхній стабільності та успіху.

Зрештою, в основі сучасної АЗС лежать споживчі вимоги та зручність для клієнтів. Швидкість обслуговування, різні способи оплати, точність обліку, зручність для користувачів та безпека даних – усі ці фактори роблять обґрунтованим рішення щодо впровадження систем автоматизації на АЗС. Забезпечуючи зручність

та задоволеність клієнтів, заправні станції можуть покращити свою репутацію та залучити більше клієнтів.

На закінчення відзначимо, що системи автоматизації на АЗС – важливий та актуальний напрямок розвитку. Вони допомагають заправним станціям підвищити ефективність, конкурентоспроможність та задоволеність клієнтів. Основними перевагами є підвищена точність та швидкість обслуговування, а також забезпечення безпеки та захисту даних. Ці системи стають важливим інструментом для заправних станцій у досягненні успіху та стійкості на паливному ринку.

В другому розділі було виконано проектування інформаційної системи для колонки самообслуговування, а саме розглянуті існуючи рішення на ринку програмного забезпечення та їх недоліки для виявлення вимог та побудови інформаційної системи для терміналу колонки самообслуговування АЗС. На основі отриманих даних була побудована схема концепції для створюваного програмного забезпечення, побудували діаграму використання та специфікацію варіантів використання провівши детальний опис кожного з варіантів. Кінцевою стадією проектування було будування концептуальної модель даних, яка дозволила описати концептуальні схеми за допомогою узагальнених конструкцій блоків і перейти до реалізації програмного забезпечення.

Реалізацію інформаційної системи для колонки самообслуговування було розпочато з побудови діаграми послідовності яка відображає взаємодію та порядок дій програмних модулів. Далі була побудована діаграма станів яка слугує для описання реакції системи на вплив ззовні та розглянута специфікація переходів станів. Була побудована логічна і фізична моделі бази даних, детальніше розглянута структура фізичної моделі. Були встановлені мінімальні системні вимоги для експлуатації інформаційної системи та прийняті рішення з програмного забезпечення для написання ІС. Кінцевим кроком було будування діаграми розгортання, що дозволило закінчити реалізацію ІС.

В четвертому розділі розглянута інструкція користувача. Також було проведено тестування інформаційної системи для автоматизації АЗС. В ході тестування жодних помилок не виявлено, програмний застосунок працює коректно.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. C# Guide [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/.
2. Developer Survey. [Електронний ресурс] — Режим доступу: https://insights.stackoverflow.com/survey/2020#most-popular-technologies
3. Digital in 2020. – Електрон. дан. – Режим доступу: https://wearesocial.com/digital-2020
4. F. T. Barwell (2013). "Automation and Control in Transport." CRC Press.
5. Ionic - Cross-Platform Mobile App Development – [Електронний ресурс]. https://ionicframework.com/
6. JQuery API Documentacion [Електронний ресурс]. – Режим доступу: https://api.jqueryui.com/
7. Kukulska-Hulme A. Mobile language learning now and in the future. / A. Kukulska-Hulme– London: Swedish Net University, 2006. – P. 295-310.
8. Lupu, I., & Marinescu, C. (2019). "Sustainable Mobility: Analysis and Models for a European Transport System." Springer.
9. Marta Biega´nska, Beata Paliwoda and Justyna Górna. (2014). "Internet of Things (IoT) in Industry: Research Profiling, Application, Challenges and Opportunities."
10. Mehmood, Y., & Hu, J. (2016). "A survey of big data architectures and machine learning algorithms in healthcare." Journal of King Saud University-Computer and Information Sciences.
11. Mobile Technology. [Електронний ресурс] – Режим доступу: http://en.wikipedia.org/wiki/Mobile_technology
12. Mobile Technology. [Електронний ресурс] – Режим доступу: http://www.itbusinessedge.com/topics/show.aspx?t=738
13. MySQL [електронний ресурс] // Режим доступу: https://www.mysql.com/ - Назва з екрану
14. Web-технології та Web-дизайн" [Електронний ресурс] – 2015 – Режим доступу до ресурсу:https://dl.sumdu.edu.ua/textbooks/86975/413008/index.html

15. What is Mobile Technology? [Електронний ресурс] – Режим доступу: http://www.strategicgrowthconcepts.com/growth/mobile-technology-facts.html
16. What is Use Case Diagram?- [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-usecase-diagram/
17. Yogesh R. Python: Simple though an Important Programming language [Електронний ресурс] / R. Yogesh // International Research Journal of Engineering and Technology, 2019. — Vol. 06. — P. 1856-1858. — Режим доступу: https://www.irjet.net/archives/V6/i2/IRJET-V6I2367.pdf
18. Автоматизація АЗС і паливозаправників: https://www.umt.ua/solutions/pidkluchennya-pkr-ta-paluvozapravnukiv/
19. Введение в C# [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/1.1.php.
20. Гнатієнко Г.М., Снитюк В.Є. Експертні технології прийняття рішень: Монографія / К.: ТОВ «Маклаут», 2008. 444 с.
21. Ілляшенко С.М. Сучасні тенденції застосування інтернет-технологій. /С.М. Ілляшенко// [Електронний ресурс] – Режим доступу: http://www.nbuv.gov.ua/portal/Soc_Gum/Mimi/2011_4_2/2_1.pdf
22. Мережа АЗС WOG оптимізувала процеси за допомогою бізнес-рішень Apple: https://solutions.asbis.ua/news/petrol-station-network-wog-optimized-processes-with-apple-business-solutions
23. Начало работы. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/1.2.php.
24. Організаційна структура проекту (OBS) – [Електронний ресурс]. – Режим доступу: https://studopedia.com.ua/1_243503_organizatsiyna-struktura-proektuoBS.html
25. Попова Ю. В. Сутність і технічні інструменти інтернет-маркетингу. /Ю.В. Попова// [Електронний ресурс] – Режим доступу: http://www.nbuv.gov.ua/portal/soc_gum/Uproz/2011_6/u1106pop.pdf
26. Принципи побудови та етапи проектування баз даних − [Електронний ресурс], — Режим доступу: http://stud.com.ua/35671/informatika/kontseptsiya_baz_danih

27. Система керування базами даних [електронний ресурс] // Режим доступу: http://uk.wikipedia.org/wiki/Система_керування_базами_даних – Назва з екрану.
28. Типи мобільних додатків. [Електронний ресурс] — Режим доступу: https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types
29. Що таке JQuery? [Електронний ресурс]. – Режим доступу: https://phpacademy.kiev.ua/uk/blog/what-is-jquery

# ДОДАТКИ

## Додаток А. Частина коду програми

```
<?php

namespace App\Console;

use Illuminate\Console\Scheduling\Schedule;
use Illuminate\Foundation\Console\Kernel as ConsoleKernel;

class Kernel extends ConsoleKernel
{
    /**
     * The Artisan commands provided by your application.
     *
     * @var array
     */
    protected $commands = [
        //
    ];

    /**
     * Define the application's command schedule.
     *
     * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
     * @return void
     */
    protected function schedule(Schedule $schedule)
    {
        // $schedule->command('inspire')
        //          ->hourly();
    }

    /**
     * Register the commands for the application.
```

```
     *
     * @return void
     */
    protected function commands()
    {
        $this->load(__DIR__.'/Commands');

        require base_path('routes/console.php');
    }
}
<?php

namespace App\Exceptions;

use Exception;
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

class Handler extends ExceptionHandler
{
    /**
     * A list of the exception types that are not reported.
     *
     * @var array
     */
    protected $dontReport = [
        //
    ];

    /**
     * A list of the inputs that are never flashed for validation exceptions.
     *
     * @var array
     */
    protected $dontFlash = [
        'password',
```

```
        'password_confirmation',
    ];

    /**
     * Report or log an exception.
     *
     * @param  \Exception  $exception
     * @return void
     *
     * @throws \Exception
     */
    public function report(Exception $exception)
    {
        parent::report($exception);
    }

    /**
     * Render an exception into an HTTP response.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Exception  $exception
     * @return \Symfony\Component\HttpFoundation\Response
     *
     * @throws \Exception
     */
    public function render($request, Exception $exception)
    {
        return parent::render($request, $exception);
    }
}
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
```

```
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
<?php
```

```
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('auth.login');
});


Route::get('/register', function () {
    return view('auth.register');
});

Route::get('/relatorios', function () {
    return view('postos.relatorio');
});


Auth::routes();

Route::resource('/postos','PostosController')
->middleware('auth');

Route::resource('/cidades','CidadesController')
->middleware('auth');

Route::resource('/combustiveis','CombustiveisController')
```

```
->middleware('auth');

Route::get('/home', 'HomeController@index')->name('home');

Route::get('/logout', '\App\Http\Controllers\Auth\LoginController@logout');
<?php

/*
|--------------------------------------------------------------------------
| Create The Application
|--------------------------------------------------------------------------
|
| The first thing we will do is create a new Laravel application instance
| which serves as the "glue" for all the components of Laravel, and is
| the IoC container for the system binding all of the various parts.
|
*/

$app = new Illuminate\Foundation\Application(
    $_ENV['APP_BASE_PATH'] ?? dirname(__DIR__)
);

/*
|--------------------------------------------------------------------------
| Bind Important Interfaces
|--------------------------------------------------------------------------
|
| Next, we need to bind some important interfaces into the container so
| we will be able to resolve them when needed. The kernels serve the
| incoming requests to this application from both the web and CLI.
|
*/

$app->singleton(
    Illuminate\Contracts\Http\Kernel::class,
```

```
    App\Http\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Console\Kernel::class,
    App\Console\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Debug\ExceptionHandler::class,
    App\Exceptions\Handler::class
);

/*
|--------------------------------------------------------------------------
| Return The Application
|--------------------------------------------------------------------------
|
| This script returns the application instance. The instance is given to
| the calling script so we can separate the building of the instances
| from the actual running of the application and sending responses.
|
*/

return $app;
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\User;
use Faker\Generator as Faker;
use Illuminate\Support\Str;

/*
|--------------------------------------------------------------------------
```

```
| Model Factories
|--------------------------------------------------------------------------
|
| This directory should contain each of the model factory definitions for
| your application. Factories provide a convenient way to generate new
| model instances for testing / seeding your application's database.
|
*/

$factory->define(User::class, function (Faker $faker) {
    return [
        'name' => $faker->name,
        'email' => $faker->unique()->safeEmail,
        'email_verified_at' => now(),
        'password'  =>  '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi',  //
password
        'remember_token' => Str::random(10),
    ];
});
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
```

```
            $table->bigIncrements('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePasswordResetsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
```

```
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
}
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateFailedJobsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->text('connection');
```

```
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
}
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class MigrationCidades extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('cidades', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nome');
```

```
            $table->string('uf');
            $table->string('cep');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('cidades');
    }
}
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class MigrationPostos extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('postos', function (Blueprint $table) {
            $table->increments('id');
            $table->string('cnpj');
            $table->string('razao_social');
```

```
            $table->string('nome_fantasia');
            $table->string('bandeira');
            $table->string('endereco');
            $table->string('bairro');
            $table->unsignedInteger('id_cidade');
            $table->foreign('id_cidade')->references('id')->on('cidades');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('postos');
    }
}
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class MigrationCombustiveis extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
```

```
        Schema::create('combustiveis', function (Blueprint $table) {
            $table->increments('id');
            $table->string('tipo');
            $table->date('data_coleta');
            $table->decimal('preco_venda', 8, 3);
            $table->unsignedInteger('id_posto');
            $table->foreign('id_posto')->references('id')->on('postos');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('combustiveis');
    }
}
```