**РЕФЕРАТ**

Пояснювальна записка складається з 8 розділів:

* вступ – визначає актуальність роботи, її цілі і головні задачі. Складається з 2 сторінок;
* призначення, постановка задачі, огляд аналогів і літератури – містить опис призначення роботи та методики вирішення поставленої задачі, приведені аналоги та огляд літератури. Складається з 17 сторінок;
* зовнішнє та внутрішнє проектування – містить інженерно-технологічну постановку задачі, процес вибору парадигми проектування, та архітектури програмування. Складається з 12 сторінок;
* розробка програми – містить проектування інтерфейсу користувача, проектування архітектури системи, вибір мови програмування та технологій. Складається з 35 сторінок;
* відлагодження та тестування – містить опис обраних методів тестування та процесу відлагодження. Складається з 14 сторінок;
* охорона праці та безпека в надзвичайних ситуаціях – містить аналіз шкідливих та небезпечних факторів при роботі з ЕОМ, та заходи щодо їх мінімізації. Складається з 10 сторінок;
* висновки – складається з 1 сторінки;
* література – список використаних літературних джерел. Складається з 2 сторінок.

Технічне завдання та робочий проект містяться в додатках.

Кількість таблиць: 28.

Кількість рисунків: 26.

Діаграма, графік, клієнт, сервер, діагностична система, вузли локомотиву, база даних, веб-інтерфейс, файл, статистичні дані, порівняння, авторизація.

**ЗМІСТ**

[ВСТУП 7](#_Toc390765157)

[1 ПРИЗНАЧЕННЯ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД АНАЛОГІВ І ЛІТЕРАТУРИ 9](#_Toc390765158)

[1.1 Призначення та область застосування 9](#_Toc390765159)

[1.2 Постановка задачі 9](#_Toc390765160)

[1.3 Огляд літератури 10](#_Toc390765161)

[1.4 Аналіз аналогів 18](#_Toc390765162)

[1.4.1 Характеристика системи ruDi. 18](#_Toc390765163)

[1.4.2 Характеристика системи "Магістраль" ДЕ1м. 24](#_Toc390765164)

[1.5 Аналіз розглянутих аналогів 24](#_Toc390765165)

[2 ЗОВНІШНЄ ТА ВНУТРІШНЕ ПРОЕКТУВАННЯ 26](#_Toc390765166)

[2.1 Зовнішнє проектування 26](#_Toc390765167)

[2.1.1 Опис функціональних характеристик 26](#_Toc390765168)

[2.1.2 Формалізація задач 27](#_Toc390765169)

[2.2 Внутрішне проектування 30](#_Toc390765170)

[2.2.1 Вибір парадигми програмування 30](#_Toc390765171)

[2.2.2 Вибір архітектури програмування 34](#_Toc390765172)

[3 РОЗРОБКА ПРОГРАМИ 38](#_Toc390765173)

[3.1 Проектування інтерфейсу користувача 38](#_Toc390765174)

[3.1.1 Класифікація веб-сайту 39](#_Toc390765175)

[3.1.2 Класифікація веб-дизайну 44](#_Toc390765176)

[3.1.3 Проектування макету сторінок 45](#_Toc390765177)

[3.1.4 Верстання сторінок 52](#_Toc390765178)

[3.2 Проектування архітектури системи 55](#_Toc390765179)

[3.3 Використані технології 62](#_Toc390765180)

[3.3.1 Вибір мови програмування 62](#_Toc390765181)

[3.3.2 Загальна характеристика Spring Framework. 63](#_Toc390765182)

[3.3.3 Загальна характеристика Spring Security. 64](#_Toc390765183)

[3.3.4 Загальна характеристика Apache Maven. 66](#_Toc390765184)

[3.3.5 Загальна характеристика Highcharts. 66](#_Toc390765185)

[3.3.6 Загальна характеристика Twitter Bootstrap. 68](#_Toc390765186)

[3.3.7 Загальна характеристика jQuery. 68](#_Toc390765187)

[3.3.8 Загальна характеристика Thymeleaf. 70](#_Toc390765188)

[3.3.9 Загальна характеристика Apache Tomcat. 71](#_Toc390765189)

[3.3.10 Загальна характеристика IntelliJ IDEA. 71](#_Toc390765190)

[4 ВІДЛАГОДЖЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМИ 73](#_Toc390765191)

[4.1 Функціональне тестування 73](#_Toc390765192)

[4.2 Відлагодження програми 85](#_Toc390765193)

[5 ОХОРОНА ПРАЦІ 87](#_Toc390765194)

[5.1. Аналіз шкідливих та небезпечних виробничих факторів 87](#_Toc390765195)

[5.2. Проектні заходи 88](#_Toc390765196)

[5.3. Безпека проці при виконанні робіт на ПЕОМ 92](#_Toc390765197)

[5.3.1. Вимоги безпеки праці перед початком роботи на ПЕОМ 92](#_Toc390765198)

[5.3.2. Вимоги безпеки праці під час роботи на ПЕОМ 93](#_Toc390765199)

[5.3.3. Вимоги безпеки праці після закінчення роботи на ПЕОМ 96](#_Toc390765200)

[5.3.4. Вимоги безпеки праці в аварійних ситуаціях 96](#_Toc390765201)

[ВИСНОВКИ 97](#_Toc390765202)

[ЛІТЕРАТУРА 98](#_Toc390765203)

# ВСТУП

На Українській залізниці планується поступовий перехід від планово попереджувальної системи ремонту рухомого складу, до системи ремонту рухомого складу з урахуванням його фактичного технічного стану. Такий перехід можливий тільки в тому разі, коли є найбільш достовірна інформація про стан кожної окремої одиниці рухомого складу. Таку інформацію можуть дати і тестові діагностичні системи, але, для економії часу та коштів, потрібно раціональніше впроваджувати функціональні засоби діагностування. До таких засобів належать бортові системи діагностування локомотивів.

На сьогоднішній день ремонт та технічне обслуговування локомотивів виконується згідно з фіксованим графіком, коли необхідність та вид чергового ремонту, як правило, визначаються пробігом локомотива. Такий підхід не враховує фактичний стан вузлів локомотивів і реальну необхідність в ремонті даного виду, що призводить до значних додаткових витрат. Крім того, відсутність інформації про обсяг і перелік ремонтних робіт для конкретних локомотивів ускладнює планування ремонту в депо та суттєво збільшує час його виконання.

Одним з варіантів вирішення цієї проблеми є удосконалення методів діагностування та прогнозування зміни технічного стану вузлів локомотивів, розробка методів визначення періодичності та обсягів виконання ремонтів на основі діагностичних даних та «історії» експлуатації локомотива.

Досвід розробки та впровадження систем управління локомотивним парком з використанням результатів діагностування за кордоном підтверджує ефективність такого підходу. Провідні світові компанії виробники локомотивів приділяють значну увагу впровадженню засобів діагностування та організації системи сервісного обслуговування локомотивів на основі результатів діагностування. [1-4].

Системи збору діагностичних даних встановлюються на багатьох локомотивах, під час проведення діагностування система формує запис показників всіх датчиків локомотива. Сформований запис зберігається в журналі показників датчиків локомотива, на кожному локомотиві існує свій автономний журнал (база даних) показників датчиків, який формується з багатьох записів.

Для аналізу діагностичної інформації в умовах депо Нижньодніпровськ Вузол та Красний Лиман використовується програмний комплекс Магістраль Центр розроблений НВО Квант-Радіоелектроніка. [5]. За допомогою цього комплексу інформація з бортового модуля пам’яті кожного електровоза переноситься до центральної діагностичної бази даних депо.

В рамках дипломного проекту буде розроблено програмний комплекс «Менеджер статистики локомотивів» призначений для перегляду статистики систем діагностування локомотивів.

«Менеджер статистики локомотивів» буде створено у вигляді веб-сервісу, для спрощення його обслуговування та модернізації з використанням інтерактивних діаграм, що дозволить універсальний віддалений доступ до даних кінцевому користувачу.

Такий шлях вирішення проблеми є актуальним та концептуально новим, через те, що проектоване рішення є орієнтованим на розширення функціональності за допомогою зовнішніх модулів.

Результатом проектування та розробки буде програмний комплекс аналіз роботи якого дозволить уточнювати перелік контрольованих параметрів, удосконалювати алгоритми контролю, періодичність опитування датчиків контролюючих параметри вузлів і агрегатів локомотивів, а також використовувати інформацію при розробці й проектуванні вузлів локомотивів для забезпечення їх придатності до контролю.

# 1 ПРИЗНАЧЕННЯ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД АНАЛОГІВ І ЛІТЕРАТУРИ

## 1.1 Призначення та область застосування

Програмний комплекс, що розробляється, призначений для перегляду статистики систем діагностування локомотивів.

Функціональне призначення продукту полягає у наданні сервісу що зможе адмініструвати систему, виконувати пошук діагностичних даних локомотивів за період часу та відображати їх значення на діаграмі та таблицях. Експлуатаційне призначення – спрощення доступу до діагностичних даних локомотивів та надання можливості побудови порівняльних діаграм з характеристиками вузлів локомотивів.

Програма може бути розповсюджена та використана у локомотивному господарстві Укрзалізниці, де для збору діагностичних даних локомотивів використовується програмний комплекс Магістраль.

## 1.2 Постановка задачі

Необхідно розробити програмний комплекс, що надасть змогу видалено переглядати статистичну інформацію систем діагностування локомотивів.

Результатом розробки повинен стати сервіс, що надає програмні інтерфейси для перегляду статистичної інформації локомотивів. Кінцевому користувачу повинні бути надані можливості для побудови порівняльних діаграми з характеристиками вузлів локомотивів та можливість друку та збереження діаграм в графічному форматі. Порівняльна діаграма повинна відображати данні у вигляді графіка, забезпечувати масштабування та відображати дані за декількома локомотивами та декількома вузлами на одній площині. Програма повинна надати можливість завантаження файлу з статистичною інформацією про вузли локомотивів. Продукт повинен забезпечувати можливість клієнт-серверної взаємодії, без обмеження мовою програмування.

Продукт має відповідати наступним вимогам: гнучкі критерії для пошуку діагностичної інформації в базі даних, відмово стійкість серверу, забезпечення гнучкої системи контролю доступу до ресурсів.

## 1.3 Огляд літератури

Під час виконання аналізу літератури розглядалися наступні питання: загальна характеристика бортових систем діагностування локомотивів, дослідження технічного стану локомотивів з використанням інформації бортових систем діагностування.

Посилення конкуренції між різними видами транспорту вимагає від локомотивного господарства підвищення експлуатаційної надійності локомотивного парку, скорочення часу простою локомотивів в ремонті і витрат на проведення ремонту. Підвищення експлуатаційної надійності повинно досягатись з мінімальними витратами, з точки зору технічного обслуговування це означає максимальне скорочення часу простою локомотивів і зменшення імовірності виходу локомотивів з ладу під час експлуатації. Одним з шляхів вирішення цієї задачі є впровадження на локомотивах засобів технічного діагностування.

Аналіз впровадження і використання систем діагностування на залізничному транспорті за кордоном, впровадження систем оптимізації технічного обслуговування транспортних засобів в інших видах транспорту дозволяє зробити висновок про високу ефективність використання систем діагностування при переході до системи ремонту локомотивів з урахуванням їх експлуатаційної надійності. Сучасний рівень розвитку технічних засобів дозволяє організувати спільну роботу систем управління, діагностування, системи контролю місця знаходження локомотива, систем бездротової передачі інформації на пункти технічного обслуговування локомотивів.

Найбільш перспективним напрямком розвитку діагностичних засобів локомотивів є бортові (вбудовані) системи діагностування локомотивів. Такі системи забезпечують найбільш повну реалізацію ресурсу вузлів і агрегатів локомотивів, попереджають аварійні відмови, знижують експлуатаційні витрати на утримання локомотива. Впровадження вбудованих систем діагностування здійснюється, у першу чергу, для тих вузлів і агрегатів, технічний стан яких впливає на безпеку руху поїздів.

Бортові системи мікропроцесорного діагностування, згідно [6], уперше були застосовані в середині 1980-х рр. на тепловозах серії 60 відділення Е1 есtrо моdel (ЕМD) корпорації General Моtоrs і на тепловозах корпорації General Еlесtriс (GЕ). Пізніше фірма Rockwell Іnternational випустила систему аналізу й реєстрації технічного стану локомотивів (LARS), яку залізниця Burlington Northern (США) застосувала на 100 тепловозах SD40-2 і на 50 тепловозах ОР50 фірми ЕМD. Система LARS вимірює до 40 параметрів роботи (або відмов у роботі) локомотива й передає інформацію в пункт технічного обслуговування й ремонту тепловозів. Розглянуті системи діагностування автоматично передають по радіоканалах інформацію з локомотива в наступний пункт технічного обслуговування й ремонту, для того щоб його персонал міг підготуватися до виконання ремонту.

Перевагою бортових систем діагностування є контроль стану локомотива безпосередньо в процесі руху, що дозволяє виявляти причини виникнення й попереджати раптові відмови, виявляти сховані несправності, які можуть проявлятися тільки при деяких режимах роботи локомотива. Бортові системи діагностування підвищують безпека руху завдяки безперервному контролю найбільш відповідальних вузлів локомотива.

Бортові системи діагностування відносяться до нижнього рівня автоматизованих систем. Вони представляють інформацію для систем верхнього рівня (системи автоведения, системи автоматичного керування тяговим приводом, автоматизовані системи безпеки).

Недоліком бортових систем діагностування є їх висока вартість, що обумовлено складними умовами експлуатації рухомого складу, а також забезпеченням придатності вузлів і агрегатів до контролю. Використовувані мікропроцесорні пристрої повинні надійно працювати в умовах значних перепадів температур, вібрації, впливу електромагнітних полів, для чого необхідне використання обладнання спеціального виконання.

Новий рухомий склад українських залізниць обладнаний бортовими системами діагностування. Уперше на вітчизняних локомотивах такі системи були встановлені на електровозах серії ДЕ1. Система діагностування контролює параметри основних вузлів електровоза, таких як: тягові двигуни, допоміжні машини, пуско-гальмові резистори, акумуляторна батарея, перетворювач, букси. Також система контролює стан електричних апаратів силовому ланцюга й ланцюгів керування.

Система працює в чотирьох режимах, які дозволяють безупинно контролювати вузли, переглядати поточні значення параметрів і накопичену діагностичну інформацію. Вікно системи діагностування показане на рисунку 1.1, структура системи показана на рисунку 1.2.

Технічна характеристика мікропроцесорної системи "МАГИСТРАЛЬ-ДЕ1М":

* кількість контрольованих параметрів: 400;
* частота опиту датчиків 1 Гц;
* напруга живлення 50 В;
* споживана потужність по ланцюгу 50 В, 126 Вт;
* пусковий струм по ланцюгу 50 В, 15 А;
* маса, 150 кг;
* напрацювання на відмову, 120 тис./ год;
* час підготовки системи до роботи, 60 с.



Рисунок 1.1 - Вікно бортової системи діагностування електровоза ДЕ1.

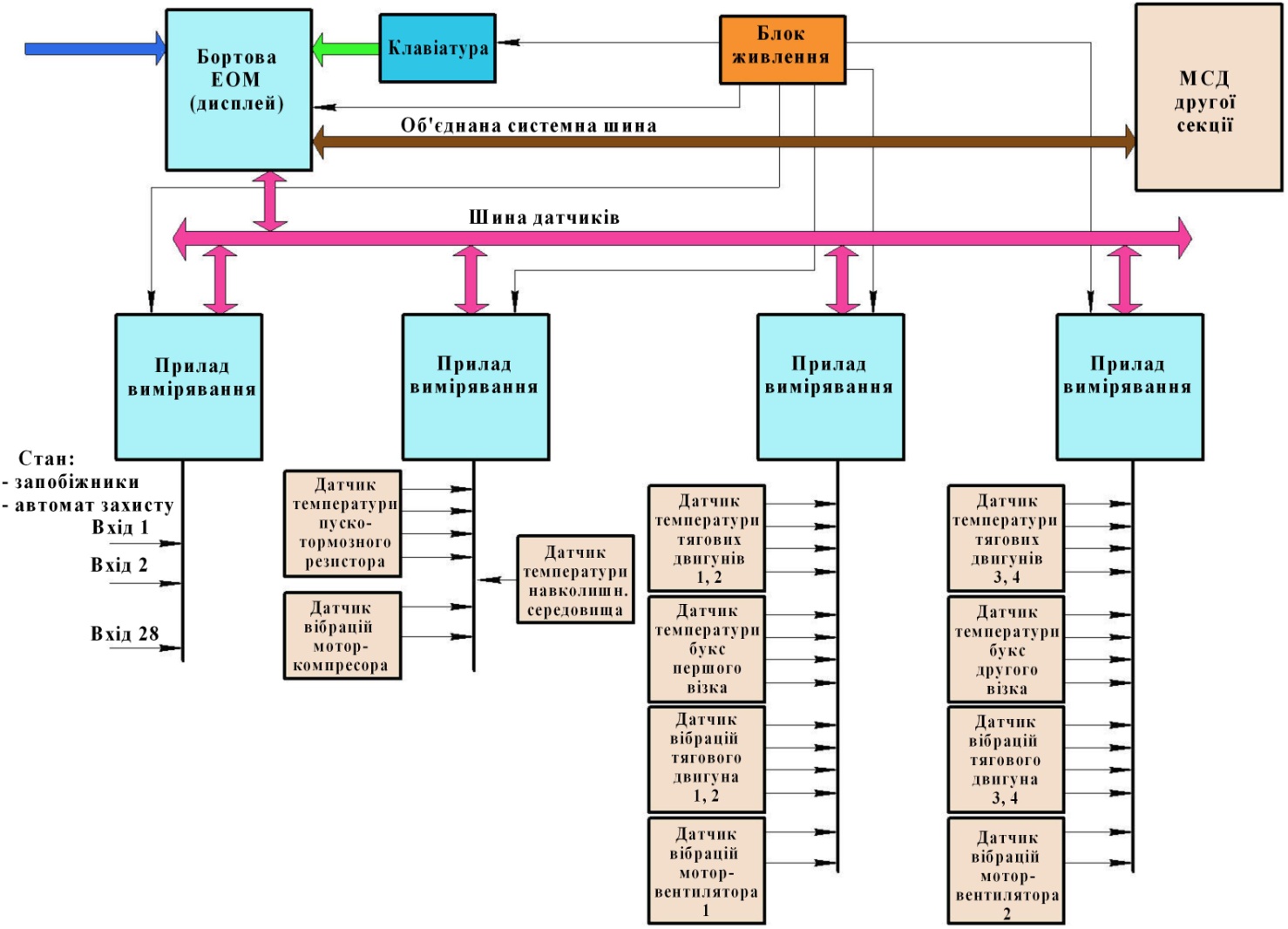


Рисунок 1.2 – Структурна схема МСД "МАГИСТРАЛЬ-ДЕ1М"

Система забезпечує вимірювання аналогових і дискретних параметрів зокрема:

– температури БПТР (8 каналів);

– температури на вході і виході тягових двигунів (16 каналів);

– температури букс (16 каналів);

– температура навколишнього середовища (2 канали);

– віброприскорення на тягових двигунах в двох площинах (16 каналів);

**–** віброприскорення на моторвентиляторах і моторкомпрессорах;

– положення запобіжників і автоматів захисту в ланцюгах живлення.

Також прийом даних з пристрою управління тяговим електроприводом (УУТЕП):

– струми і напруга тягових двигунів;

– струми і напруга акумуляторних батарей;

– струми і напруга перетворювачів;

– струми вентиляторів і компресорів;

– напруга контактної мережі;

– стан контакторів, перемикачів, вимикачів, реле та іншого устаткування.

Виявлення відхилень в роботі устаткування по критерію перевищення допустимих і критичних значень для аналогових параметрів.

Візуальне і звукове сповіщення про досягнення параметрами критичних і гранично допустимих значень, а також про появу заборонених комбінацій дискретних сигналів).

Відображення поточного стану контрольованих параметрів на відео моніторі.

Реєстрацію інформації про відхилення від норми в роботі устаткування в незалежному модулі пам’яті.

Розшифровка даних, записаних в модулі пам'яті, і видача підсумкового зведення про кількість і характер зареєстрованих відхилень в роботі устаткування електровоза.

Прийом і відображення даних з швидкостеміра (швидкість, тиск, обмеження швидкості).

Розрахунок тягових і гальмівних зусиль і потужності споживаною тяговим і допоміжним устаткуванням і відображення результатів на відеомоніторі.

Програмні і апаратні засоби системи забезпечують функціонування систем діагностики двох секцій електровоза в єдиному комплексі, в якому обробляється, відображається на відеомоніторі і реєструється в модулі пам’яті вимірювальна інформація всього електровоза.

Система побудована за блоково-модульним принципом і складається з окремих конструктивно і функціонально закінчених елементів різних рівнів складності: приладів, блоків, вузлів і датчиків, що розміщуються в кузові електровоза і на візках. Всі компоненти системи з'єднуються між собою за допомогою джгутів монтажного комплекту.

Програмне забезпечення системи (ПО) розміщується в твердотільних пристроях (ЗУ) бортової ЕОМ (зберігаючих пристроях). Збір вимірювальної інформації на рівні системи організований за магістральним принципом. У кожній секції збір здійснюється по послідовній шині датчиків (ШД).

Як базові технічні засоби вимірювального поста використовуються уніфіковані, системно сумісні прилади, до складу яких входять один або два програмно керованих модуля збору інформації з сімейства IDEM-7000. Модулі збору інформації вирішують завдання комутації вимірювальних каналів, кодування вимірювальної інформації і передачі її в прилад (ЗУ) по послідовному каналу шини датчиків.

До складу системи кожної секції входять два блоки (основний і резервний) вторинного електроживлення.

Для накопичення і аналізу діагностичної інформації безпосередньо на електровозі в системі діагностування передбачений режим «Статистика». Додатково для розшифрування й накопичення інформації в умовах депо використовується програмне забезпечення для розшифрування й аналізу інформації.

Система діагностування електровозів серії ДС3 за своїми функціями аналогічна системі діагностування електровоза ДЕ1. Відмінною рисою цієї системи є більш зручний графічний інтерфейс при відображенні інформації. Системою також передбачене накопичення діагностичної інформації, розшифрування якої виконується представниками фірми Siemens.

Системи діагностування дизель-поїздів ДЕЛ02 і тепловозів ТЕП150 також виконують контроль дизеля і його систем, вібрації й електричних параметрів тягових електродвигунів, параметрів акумуляторної батареї, а також контроль роботи електричних апаратів, температури букс, тиску в гальмових циліндрах. Структура системи діагностування дизель поїзда СПРАВ-02 наведена на рисунку 1.3. Вікно системи діагностування показане на рисунку 1.4.

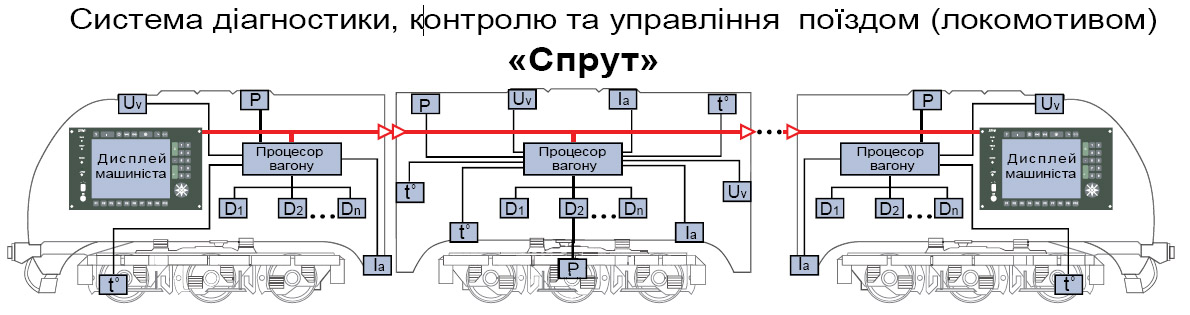


Рисунок 1.3 – Структура системи діагностування дизель поїзда ДЕЛ02

Системи діагностування електровозів ДЕ1 і ДС3 виконують функції контролю й накопичення діагностичної інформації про основні вузли. На інших типах рухомого состава (електровози ВЛ11М6, ЕПЛ2Т, 2С5К и ін.) також використовуються бортові системи діагностування, структура яких аналогічна розглянутим системам. У цілому кількість діагностичних параметрів у бортових системах діагностування нових і існуючих локомотивів більшою мірою залежить від технічної можливості вимірювати той або інший параметр існуючими засобами діагностування, а також від кількості коштів, виділених розроблювачам на систему діагностування. Ефективність системи залежить не тільки від кількості діагностичних параметрів, але в значній мірі від організації роботи системи діагностування й використання отриманої інформації ремонтними службами.

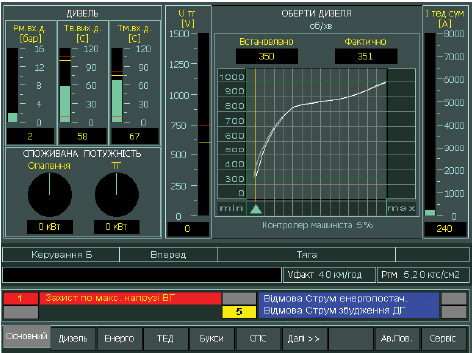


Рисунок 1.4 – Вікно системи діагностування дизель поїзда ДЕЛ-02

На підставі аналізу існуючих бортових систем діагностування локомотивів можна зробити висновок, що їх впровадження на сьогоднішній день дозволяє:

* підвищити безпеку руху (попереджати аварійні ситуації під час руху, контролювати дії локомотивної бригади);
* скоротити час пошуку несправності в електричних схемах (контролювати спрацьовування електричних апаратів);
* виявляти (а частіше — підтверджувати) факт настання відмови контрольованих вузлів.

Фактично системи діагностування виконують функцію моніторингу (безперервного спостереження) зміни технічного стану вузлів локомотивів.

З позиції технічної діагностики [7] основними завданнями будь-якої системи діагностування є:

* визначення працездатності об'єкта діагностування (справний/несправний);
* контроль функціонування (контроль того, щоб у процесі експлуатації не виникали відмови, що порушують працездатність);
* пошук несправності (вказівка місця виникнення несправності й способу її усунення);
* прогнозування зміни технічного стану об'єкта діагностування надалі;
* визначення попереднього стану об'єкта діагностування.

Існуючі системи діагностування тією чи іншою мірою виконують перші три завдання, але їх розв'язок не дозволяє чітко визначити обсяги ремонту й планувати його періодичність, що необхідно для переходу на ремонт рухомого состава з обліком його фактичного технічного стану. Це різко знижує ефективність використання систем діагностування й формує негативне відношення до самих систем діагностування з боку ремонтних служб.

Для підвищення ефективності систем діагностування необхідно налагодити зворотний зв'язок між діагностичними системами й ремонтними службами депо. Для цього слід розробити технологію запису, зберігання й обробки інформації від систем діагностування локомотива в депо. Із цією метою вхідна інформація повинна фіксуватися, накопичуватися, систематизуватися й використовуватися для розрахунків системи утримання кожного локомотива.

Тільки на підставі спільного аналізу діагностичної інформації й даних про відмови локомотивів у процесі експлуатації може бути розрахована раціональна система утримання та отриманий ефект від впровадження систем діагностування.

Крім того, подальший аналіз роботи систем діагностування дозволить уточнювати перелік контрольованих параметрів, удосконалювати алгоритми контролю, періодичність опитування датчиків контролюючих параметри вузлів і агрегатів локомотивів, а також використовувати інформацію при розробці й проектуванні вузлів локомотивів для забезпечення їх придатності до контролю.

## 1.4 Аналіз аналогів

На сьогоднішній день, аналогів які призначені для тих самих цілей що проект та надають подібний функціонал не знайдено. Це пов’язане з тим, що зазначений програмний комплекс спроектовано для збору діагностичних даних локомотивів у локомотивному господарстві де використовується програмний комплекс Магістраль. Іншою причиною є те, що існуючі рішення призначені для внутрішнього використання, та не надаються для придбання.

Саме тому, в цьому розділі будуть розглянуті також закордонні рішення бортових систем діагностування локомотивів.

### 1.4.1 Характеристика системи ruDi.

У Германії розроблена система ruDi для контролю і управління парком тягового рухомого складу. Вона забезпечує реєстрацію всіх експлуатаційних даних, технічну діагностику рухомого складу в оперативному режимі і визначає його місцезнаходження в межах європейської мережі. На основі даних, що поступають безперервно, виробляється оцінка міри завантаження, окремо для кожного типа рухомого складу і визначаються терміни його технічного обслуговування. Система дозволяє не лише визначати місцезнаходження рухомого складу в будь який час, але і сприяє оптимізації процесу управління парком рухомого складу і підвищенню якості організації його технічного обслуговування. У зв'язку з цим помітно підвищується надійність і експлуатаційна готовність рухомого складу.

Завдяки модульній компоновці система ruDi має можливість розширення незалежно від виробників рухомого складу, її можна встановлювати на рухомий склад як додаткове устаткування. Експлуатаційним компаніям система ruDi забезпечує оперативний контроль парку рухомого складу, поліпшення планерування поточного вмісту, точну документальну реєстрацію всіх виробничих робіт, а також підтримку при розрахунках експлуатаційних витрат протягом всього терміну життєвого циклу (LCC).

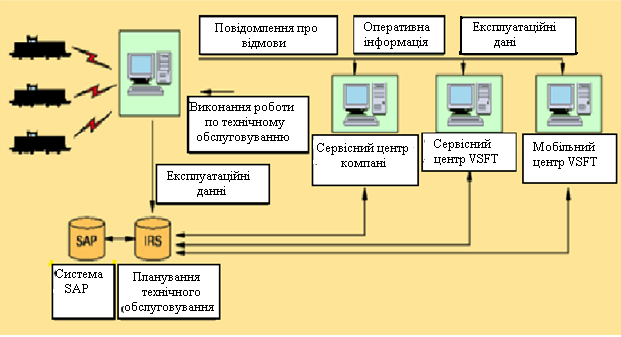


Рисунок 1.5 – Структура інтегрованої системи ruDi/VSMS

VSMS об'єднує в собі систему планування, організації і документальної реєстрації процесів технічного обслуговування тягового рухомого складу, а також системи реєстрації витрат і експлуатаційні показники. На всіх стадіях виготовлення, експлуатації і поточного обслуговування рухомого складу безперервно і в реальному масштабі часу реєструються експлуатаційні показники (години роботи, пробіг, витрата палива і т. д.). Крім того, проводиться аналіз відмов з класифікацією їх по серіях рухомого складу, типах вузлів і компаніях, що виробляють вузли та агрегати.

На основі отриманих даних ведеться автоматизоване планування заходів, щодо технічного обслуговування з визначенням потрібних ресурсів. Ці заходи документально реєструються системою, на основі чого формується «історія» поточного утримання рухомого складу і його вузлів з врахуванням інструкцій, що діють. В результаті доступності документації фахівці сервісних служб в будь-якій точці світу отримують інформацію в оперативному режимі.

З іншого боку, фахівцями сервісної служби виробляється безпосередня і точна реєстрація без використання паперових носіїв всіх проведених заходів щодо технічного обслуговування, а також пов'язаних з ними витрат .

Постійний контроль за витратами на обслуговуючий персонал і матеріальними ресурсами забезпечує їх облік протягом всього терміну служби (витрат LCC). Крім того, зібрані дані дозволяють проводити аналіз показників RAMS (надійність, експлуатаційна готовність, ремонтопридатність, безпека).

Постійна документальна реєстрація відповідних параметрів дозволяє систематично фіксувати всі виникаючі відхилення від нормального стану і аналізувати частоту і причини виникнення несправностей. Аналіз показників LCC і RAMS для різних цілей може проводитися з класифікацією за серією рухомого складу, типові вузла, а також по виробникові або компанії, що експлуатує рухомий склад. Таким чином, всі зібрані дані служать основою як для розрахунку показників LCC і RAMS, так і для розробки системи профілактичного і ремонтного обслуговування рухомого складу, що базується на чіткій інформації.

Важливою складовою частиною планування робіт по поточному утриманню тягового рухомого складу, управління і документування є реєстрація і обробка в реальному часі експлуатаційних характеристик (наприклад, частоти обертання дизелів, температури, тиску). Цю важливу задачу виконує система ruDi у складі VSMS, розробленою компанією Vossloh.

Система управління парком залізничного рухомого складу ruDi забезпечує постійну реєстрацію в реальному масштабі часу відповідних експлуатаційних показників і дозволяє проводити в оперативному режимі технічне діагностування і визначати місцезнаходження рухомого складу на будь – якій мережі.

Вона працює незалежно від встановлених на рухомому складі систем управління і тому не робить впливу на допуск рухомого складу до експлуатації. Проте за допомогою стандартних інтерфейсів, таких, наприклад, як інформаційна шина CAN, дані можуть бути отримані з систем управління і оброблені для здійснення функцій контролю.

За рахунок модульної структури система ruDi володіє високою гнучкістю вживання і може встановлюватися як на новому рухомому складі, так і тому, що експлуатується будь – яких компаніях – виробниках. Ще однією перевагою системи менеджменту рухомого складу ruDi є використання радіозв'язку для передачі даних (таких, наприклад, як сигнали від супутникових систем GSM або GPRS), завдяки чому система є мобільною.

Обробка даних може проводитися за бажанням або самим замовником, або сторонньою фірмою, що надає такі послуги. Якщо експлуатуюче підприємство користується послугами спеціалізованої фірми з обробки даних, то через певні проміжки часу воно отримує актуальну інформацію про свій рухомий склад або може само отримати таку інформацію через Інтернет.

Система ruDi має, наступні стандартні функції:

* дистанційна технічна діагностика;
* реєстрація експлуатаційних показників;
* визначення інтервалів технічного обслуговування;
* документальна реєстрація всіх виконаних робіт по технічному утриманню;
* перевірка технічного стану для сертифікації;
* аналіз витрат;
* визначення місцезнаходження рухомого складу;
* захист від розкрадань;
* визначення коефіцієнта використання рухомого складу.

Система ruDi складається з двох основних частин: мобільного устаткування рухомого складу і стаціонарної центральної інформаційної станції .

Основною складовою частиною устаткування рухомого складу є бортовий комп'ютер з радіо модемом (стандарт GSM/GPRS) і під'єднаним блоком супутникового позиціювання GPS для визначення місцезнаходження рухомого складу. При необхідності устаткування рухомого складу може доповнюватися індивідуальними датчиками.

Бортовий комп'ютер був спеціально розроблений для цих цілей. Завдяки дуже міцній конструкції він щонайкраще проявив себе в умовах залізничної експлуатації. У нього є вісім аналогових і вісім цифрових входів, а також достатній об'єм пам'яті для автономної експлуатації локомотиву без обміну інформацією з диспетчерським пунктом.

Концепція оцінки витрат життєвого циклу (LCC) набула широкого поширення у всіх галузях економіки, у тому числі і на залізничному транспорті. Головною перевагою цієї концепції для користувачів є поліпшення ухвалення рішень про інвестиції і можливість робити вплив на загальні витрати з використанням порівняльного аналізу витрат на придбання нової техніки і додаткових витрат, необхідних для ремонту старої. В даний час у зв'язку з постійним скороченням життєвих циклів продукції концепція LCC стає усе більш актуальною.

Основним завданням розрахунків LCC є завчасне планування, регулювання і контроль успішності життєвого циклу продукції. За допомогою методу LCC витрати і доходи можна аналізувати і прогнозувати вже на стадії розробки з метою регулювання подальших експлуатаційних витрат, ще до початку виробництва. Таким чином, розрахунок по методу LCC надає істотну допомогу при планування і ухваленні рішень, а також може використовуватися для обґрунтування або оцінки вже прийнятих рішень.

Метод LCC є інструментом системи менеджменту витрат, який служить для аналізу і регулювання аспектів економічної ефективності протягом всього життєвого циклу об'єкту від його розробки, реалізації і використання аж до утилізації. Крім того, можуть розроблятися різні моделі LCC з врахуванням особливостей підприємства і специфіки завдань.

На відміну від традиційних методів розрахунку, які відображають витрати і доходи, що виникають лише в період експлуатації об'єкту, в методі LCC враховуються, крім того, витрати передування і подальшого періодів, а також інші, пов'язані з об'єктом, такі, наприклад, як витрати в результаті простою. Таким чином, при складанні балансу враховуються не лише витрати, пов'язані з придбанням і виникаючі при використанні об'єкту. Розробка систем менеджменту витрат, таких, як LCC, не може замінити традиційних методів розрахунку витрат, а швидше служить для розширення їх можливостей.

Період, передуючий часу знаходження об'єкту на ринку, охоплює всі види діяльності, необхідні для його виготовлення. Сюди відносяться, наприклад, витрати на дослідження і розробку, а також витрати, пов'язані з впровадженням виробу (наприклад, презентація на виставці) і адаптацією до умов ринку.

Період, наступний за часом знаходження об'єкту на ринку, включає всі витрати, що виникають після виготовлення і реалізації об'єкту. До них відносяться, наприклад, витрати на гарантійне обслуговування, можливе повернення продукції, а також витрати на поточний вміст.

Так, витрати протягом життєвого циклу залізничного рухомого складу складаються з наступних витрат:

* інвестиційних;
* монтаж і введення в експлуатацію;
* споживання енергоносіїв;
* експлуатаційні та ремонтні витрати;
* компенсаційні в результаті простоїв і відмов;
* екологічні;
* на вивід з експлуатації і утилізацію.

Модель розрахунку витрат по методу LCC, призначена для виробника, детально враховує всі попередні витрати на такі роботи, як планування, дослідження і розробка, конструювання.

Система управління парком рухомого складу ruDi є основною складовою частиною сучасних моделей LCC, призначеною для планування і ухвалення рішень, а також для обґрунтування вже прийнятих рішень. Дані, отримані в системі ruDi, дозволяють аналізувати і регулювати витрати по поточному вмісту рухомого складу протягом всього життєвого циклу. Таким чином, система менеджменту рухомого складу ruDi є основою для розрахунку експлуатаційних витрат і витрат на поточне утримання.

### 1.4.2 Характеристика системи "Магістраль" ДЕ1м.

Проаналізувавши систему "Магістраль" ДЕ1м були виділені наступні характеристики:

* статистичні данні зчитуються і зберігаються на стаціонарному комп’ютері;
* перегляд даних у зручному форматі;
* статистика відмов вузлів за заданий період часу;
* друк даних у зручному форматі;
* данні по кожному локомотиву зберігаються в окремій базі;
* база даних має не реляційну структуру та представляє собою бінарний файл.

## 1.5 Аналіз розглянутих аналогів

Серед розглянутих аналогів, найбільш функціональним є система ruDi розроблена в Германії. Проте, зважаючи на те, що у локомотивному господарстві Укрзалізниці використовується програмний комплекс Магістраль, закордонний аналог не може буди використаний для досягнення основних цілей задачі. Це зумовлено тим, що система ruDi не має засобів інтеграції з іншими подібними програмними комплексами, насам перед с комплексом Магістраль. В свою чегру Магістраль не забезпечує потрібний функціонал для доступу до діагностичних даних локомотивів. Тому розроблювальний програмний продукт є унікальним і впершу чергу призначений на обробку даних від програмного комплексу Магістраль.

# 2 ЗОВНІШНЄ ТА ВНУТРІШНЕ ПРОЕКТУВАННЯ

## 2.1 Зовнішнє проектування

### 2.1.1 Опис функціональних характеристик

Дана робота призначена для надання сервісу що дозволить перегляд статистики систем діагностування локомотивів. Виходячи з постановки задачі, приведена характеристика функціональних вимог до програми, що розробляється.

Програмний продукт повинен надавати такі можливості:

* можливість побудовапорівняльної діаграми з характеристиками вузлів локомотивів за певний період часу;
* можливість друку та збереження діаграми в декількох різнихграфічних форматах;
* можливість перегляду у вигляді таблиці даних побудови діаграми;
* можливість ведення обліку локомотивів (додавання/видалення з бази даних);
* можливість перегляду інформації про оновлення даних (дата та час оновлень, кількість завантажених записів);
* можливість перегляду інформації про стан бази даних (кількість записів в таблицях даних, фізичний розмір базі).

Вимоги до порівняльної діаграми з характеристиками вузлів локомотивів:

* діаграма повинна відображати данні у вигляді графіка;
* забезпечувати масштабування;
* відображати дані за декількома локомотивами та декількома вузлами на одній площині;
* надавати можливість приховувати графічне відображення показників вузлів.

Збереження діаграми як графічного файлу має бути в наступних форматах:

* PNG;
* JPG;
* PDF;
* SVG.

Таблиця даних побудови діаграми повинна містити такі поля:

* номер запису;
* назва або номер локомотиву;
* дата та час вимірювання;
* назва вузла;
* значення виміру;
* одиниці вимірювання.

Інформація про оновлення даних складається з:

* номеру локомотива;
* дати оновлення;
* кількості нових завантажених записів.

Інформація про стан бази даних має містити наступне:

* фізичний об’єм бази на диску;
* кількість записів у таблицях з даними про характеристику вузлів локомотивів.

Адміністратору повинні бути надані такі можливості:

* видалення та додаванняінформації про локомотиви (номер, назва);
* завантаження до бази нових даних бортових систем діагностування локомотивів;
* додання та видалення користувачів в системі;
  + встановлення входження користувачів до груп.

### 2.1.2 Формалізація задач

Формалізація задачі на рівні зовнішнього проектування представлена у вигляді діаграми варіантів використання (Use Case Diagram) [8].

Система, що проектується представлена у вигляді декількох діаграм. Таке рішення дозволило відокремити функції та можливості користувачів із різним рівнем привілеїв. Діаграми варіантів використання системи були розділені на діаграму використання для звичайного користувача системи та користувача із правами адміністратора. На діаграмах користувач представлений у вигляді одного актора (Actor), що взаємодіє з системою за допомогою варіантів використання (Use Case). У якості актора виступає користувач системи. На діаграмі використані наступні типи відношень між варіантами використання та актором:

* відношення асоціації – відображає зв’язок між актором та варіантом використання. Відображається лінією зі стрілкою між актором і варіантом використання;
* відношення розширення – варіант використання розширює базову послідовність дій і додає власну, позначається пунктирною стрілкою з поміткою «extend», тобто такий варіант використання є опціональним, що може розширити базовий варіант;
* відношення включення – показує, що варіант використання включається в базову послідовність, позначається пунктирною стрілкою з поміткою «include», тобто є обов’язковою частиною варіанту використання.

Звичайному користувачу доступні такі варіанти використання (рис: 2.1):

* побудова порівняльної діаграми з характеристиками вузлів локомотивів за певний період часу;
* перегляду інформації про оновлення даних;
* перегляду інформації про стан бази даних;
* друку та збереження діаграм;
* зміна паролю для доступу до системи.

Для адміністратора системи доступні такі варіанти використання (рис. 2.2):

* керування користувачами;
  + реєстрація;
  + видалення;
  + встановлення входження до групи;
* керування інформацією про локомотиви;
  + реєстрація нового локомотиву в системі;
  + видалення даних про локомотив з системи;
* завантаження до бази нових даних бортових систем діагностування локомотивів.



Рисунок 2.1 – Діаграма варіантів використання для користувача системи



Рисунок 2.2 – Діаграма варіантів використання для адміністратора систему

## 2.2 Внутрішне проектування

### 2.2.1 Вибір парадигми програмування

Парадигма програмування в одних випадках «природно» визначається самою задачею, наприклад, об'єктно-орієнтований підхід при створенні пакетів графічних програм, в інших – її вибір може бути не таким очевидним і потребує значної роботи щодо обґрунтування.

Функційне програмування — парадигма програмування, яка розглядає програму як обчислення математичних функцій та уникає станів та змінних даних. Функційне програмування наголошує на застосуванні функцій, на відміну від імперативного програмування, яке наголошує на змінах в стані та виконанні послідовностей команд.

Основною особливістю функціонального програмування, яка визначає як переваги, так і недоліки даної парадигми, є те, що в ній реалізується модель обчислень без станів. Якщо імперативна програма на будь-якому етапі виконання має стан, тобто сукупність значень всіх змінних, і виробляє побічні ефекти, то чисто функціональна програма ні цілком, ні частинами стану не має і побічних ефектів не виробляє. Те, що в імперативних мовах робиться шляхом присвоювання значень змінним, в функціональних досягається шляхом передачі виразів в параметри функцій. Безпосереднім наслідком стає те, що чисто функціональна програма не може змінювати вже наявні в ній дані, а може лише породжувати нові, шляхом копіювання та / або розширення старих. Наслідком того ж є відмова від циклів на користь рекурсії.

Сильні сторони:

* підвищення надійності коду. Будь-яка функція працює тільки з локальними даними і працює з ними завжди однаково, незалежно від того, де, як і за яких обставин вона викликається. Неможливість мутації даних при користуванні ними в різних місцях програми виключає появу помилок, які тяжко виявити (таких, наприклад, як випадкове присвоювання невірного значення глобальній змінній в імперативній програмі);
* зручність організації модульного тестування. Єдиним ефектом від обчислення функції є повернений їй результат, і єдиний чинник, який впливає на результат — це значення аргументів. Таким чином, є можливість протестувати кожну функцію в програмі, просто обчисливши її від різних наборів значень аргументів. При цьому можна не турбуватися ні про виклик функцій в правильному порядку, ні про правильне формуванні зовнішнього стану;
* можливості оптимізації при компіляції. Традиційно згадуваною позитивною особливістю функціонального програмування є те, що воно дозволяє описувати програму в так званому «декларативному» вигляді, коли жорстка послідовність виконання багатьох операцій, необхідних для обчислення результату, в явному вигляді не задається, а формується автоматично в процесі обчислення функцій;
* можливості паралелізму. Оскільки відсутність побічних ефектів гарантовано, в будь-якому виклику функції завжди припустиме паралельне обчислення двох різних параметрів — порядок їх обчислення не може вплинути на результат виклику.

До недоліків відносять відсутність присвоювання і заміна їх на породження нових даних призводять до необхідності постійного виділення та автоматичного звільнення пам'яті, тому в системі виконання функціональної програми обов'язковим компонентом стає високоефективний збирач сміття. Нестрога модель обчислень призводить до непередбачуваного порядку виклику функцій, що створює проблеми при введенні-виведенні, де порядок виконання операцій є важливим. Крім того, очевидно, функції введення в своєму природному вигляді (наприклад, getchar із стандартної бібліотеки мови C) не є чистими, оскільки здатні повертати різні значення для одних і тих же аргументів, і для усунення цього потрібні певні хитрощі. [9].

Об'єктно-орієнтоване програмування - це метод програмування, заснований на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. Програмісти спочатку пишуть клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів.

Об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:

* все є об'єктами;
* всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, при якій один об'єкт потребує, щоб інший об'єкт виконав деяку дію. Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення — це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії;
* кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів;
* кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів;
* у класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії;
* класи організовані у єдину деревоподібну структуру з загальним корінням, яка називається ієрархією успадкування. Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Таким чином, програма являє собою набір об'єктів, що мають стан та поведінку. Об'єкти взаємодіють використовуючи повідомлення. Будується ієрархія об'єктів: програма в цілому — це об'єкт, для виконання своїх функцій вона звертається до об'єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об'єктів програми. Звісно, щоб уникнути безкінечної рекурсії у зверненнях, на якомусь етапі об'єкт трансформує запит у повідомлення до стандартних системних об'єктів, що даються мовою та середовищем програмування. Стійкість та керованість системи забезпечуються за рахунок чіткого розподілення відповідальності об'єктів (за кожну дію відповідає певний об'єкт), однозначного означення інтерфейсів міжоб'єктної взаємодії та повної ізольованості внутрішньої структури об'єкта від зовнішнього середовища (інкапсуляції). [10].

Задача побудови порівняльних діаграм що ставиться в рамках дипломного проекту відноситься до задач що добре вкладається в концепцію об’єктно-орієнтованого програмування.

Основні переваги ООП для практичного програмування наступні:

* наявність добре узгодженої мови проектування;
* наявність якісного середовища проектування;
* можливість керування якістю об‘єктно-орієнтованого проекту;
* можливість розробляти великі програмні проекти, розбиваючи великі задачі на невеликі, незалежні і легко доступні до огляду частини, що дозволяє залучати до розробки велику кількість виконавців;
* широкі можливості з повторного використання коду (більш спеціалізовані функції можуть бути написані за рахунок додавання частин, що роблять їх унікальними; загальна частина успадковується).

Також ООП надає можливість використовувати компонентно-орієнтований підхід до архітектури додатку, що дозволяє розділити програму на повністю незалежні модулі, комунікація між якими виконується лише через інтерфейси, для зменшення зв’язності кодової бази та простоти внесення змін.

### 2.2.2 Вибір архітектури програмування

Архітектура «Клієнт-сервер» (рис. 2.3).

Є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

* набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
* набір клієнтів, які використовують сервіси, що надаються серверами;
* мережа, яка забезпечує взаємодію між клієнтами та серверами.



Рисунок 2.3 - Архітектура «Клієнт-сервер»

Трирівнева клієнт-серверна архітектура передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка [застосунку](http://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA). Програми проміжного рівня можуть функціювати під управлінням спеціальних серверів [застосунків](http://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення — звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає). Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше [веб-сервер](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) і серверні [модулі](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C) проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі. [11].

Архітектура «Файл-сервер» (рис. 2.4).

Тут функціонують два компоненти: це файл-сервер і робоча станція. Файл-сервер і станція працюють на різних комп'ютерах, які з'єднані між собою мережею. Запити до сервера формуються на рівні доступу до файлів.



Рисунок 2.4 - Архітектура «Файл-сервер»

Недоліком даної системи є те, що дані обробляються на станціях, і зберігаються на сервері. Це тягне за собою велике навантаження на мережу і зниження продуктивності всієї системи. Крім того, проблеми одночасного доступу до даних, підтримки цілісності та узгодженості даних вирішуються копіями додатків на робочих станціях, тобто децентралізовано, що загрожує конфліктами і труднощами в їх вирішенні.

Для реалізації «Менеджеру статистики локомотивів» було обрано архітектуру «Клієнт-сервер» із-за наступних переваг:

* робить можливим, в більшості випадків, розподіл функцій обчислювальної системи між декількома незалежними комп'ютерами в мережі. Це дозволяє спростити обслуговування обчислювальної системи. Зокрема, заміна, ремонт, модернізація або переміщення сервера не зачіпають клієнтів;
* всі дані зберігаються на сервері, який, як правило, захищений набагато краще ніж у більшості клієнтів. На сервері простіше забезпечити контроль повноважень, щоб вирішувати доступ до даних тільки клієнтам з відповідними правами доступу.

# 3 РОЗРОБКА ПРОГРАМИ

## 3.1 Проектування інтерфейсу користувача

Веб-інтерфейс - це сукупність засобів, за допомогою яких користувач взаємодіє з веб-сайтом або будь-яким іншим додатком через браузер.

Класичним і найбільш популярним методом створення веб-інтерфейсів є використання HTML із застосуванням CSS і JavaScript. Однак різна реалізація HTML, CSS, DOM і інших специфікацій в браузерах викликає проблеми при розробці веб-додатків і їх подальшої підтримки. Крім того, можливість користувача налаштовувати багато параметрів браузера (наприклад, розмір шрифту, кольору, відключення підтримки сценаріїв) може перешкоджати коректній роботі інтерфейсу.

З розвитком DHTML та JavaScript набув популярності підхід до розробки інтерфейсної частини веб-застосунків, названий AJAX. Серцем технології є здатність веб-сторінки зініціювати запит до веб-сервера і отримати потрібні дані, так щоб інтерфейс не перезавантажував сторінку цілком, а лише довантажують необхідні дані і змінив потрібні частини сторінки, що робить їх більш інтерактивними і продуктивними.

Веб-інтерфейси зручні тим, що дають можливість вести спільну роботу співробітникам, які не перебувають в одному офісі (наприклад, веб-інтерфейси часто використовуються для заповнення різних баз даних або публікації матеріалів в інтернет-ЗМІ.

Веб-інтерфейс дає можливість універсального віддаленого доступу до служб та пристроїв, у цьому технології практично нема альтернатив. Але водночас, оскільки такий інтерфейс доступний усім, постають серйозні питання забезпечення безпеки, зокрема автентифікація та авторизація користувачів, шифрування переданих даних від сторонніх очей, модерація вмісту тощо. [12].

### 3.1.1 Класифікація веб-сайту

Веб-сайт — це сукупність веб-сторінок, які об'єднані між собою за змістом та навігаційно. Всі сайти, до яких доступ вільний, утворюють Всесвітню павутину — найбільше сховище інформації. Веб-сайтів є дуже багато, і класифікувати їх можна по-різному та за різними критеріями. Статичні та динамічні сайти За однією з класифікацій веб-сайти поділяють на статичні та динамічні (рис. 3.1) відповідно до способу наповнення сторінок матеріалом.

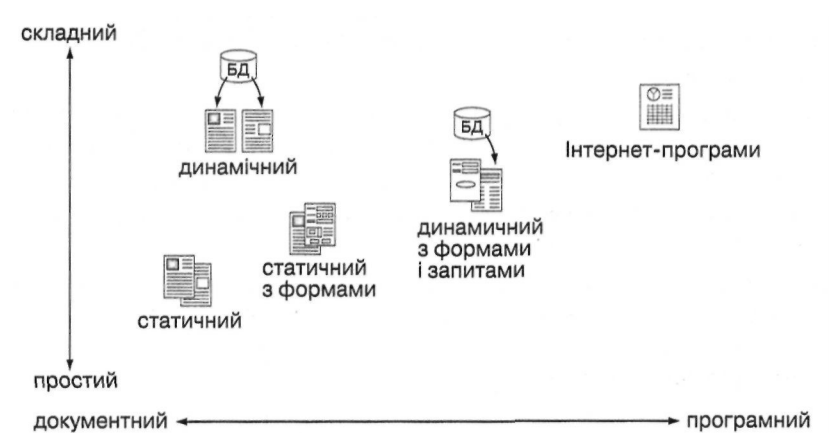


Рисунок 3.1 – Статичні та динамічні сайти

Статичний сайт — це сайт, вміст якого сталий і не може змінюватися під час відвідування сайту. Він не розрахований на часті оновлення, його легко підтримувати й одній людині, особливо за допомогою програми-редактора. Статичні сайти ще називають сайтами-брошурами.

Динамічний сайт — сайт, що може змінюватися у процесі взаємодії з відвідувачем. Зазвичай це досягається завдяки використанню програмних засобів та наявності бази даних. Отримавши запит на отримання сторінки такого сайту, програмне забезпечення веб-сервера виконує дії відповідно до цього запиту і повертає сторінку клієнту.

Тематична класифікація:

* блог (мережний журнал або щоденник) — сайт, ризначений для публічного ведення онлайнових щоденників (блогів), може також містити форум. Приклади: www.livejournal.com та www.dnevnik.org.ua;
* бізнес-сайт — містить інформацію про компанію чи послуги, які вона надає. Таких сайтів дуже багато: кожна компанія, навіть маленька, намагається подати відомості про себе у мережі Інтернет. Приклад: www.bhv.kiev.ua;
* сайт електронної комерції — забезпечує функцію електронної торгівлі для купівлі та продажу різних товарів. Наприклад, www.amazon.com;
* громадський сайт — слугує потребам деякої громади, групи людей зі спільними інтересами, які хочуть спілкуватися між собою. Наприклад, сайт www.myspace.com, де міститься інформація, за якою людину можуть знайти однокласники чи друзі через багато років після закінчення школи;
* база даних — призначений для пошуку та відображення записів бази даних, наприклад, www.imdb.com чи www.rada.gov.ua (сайт Верховної Ради України, на якому розміщено всі матеріали щодо законодавства нашої держави);
* каталог — сайт, де розміщено посилання, розподілені за категоріями та підкатегоріями; до таких сайтів належать www.yahoo.com та meta.ua;
* сайт матеріалів — призначений для завантаження файлів, таких як програми, ігри, музика, зображення в електронному вигляді тощо; Користуючись цими сайтами, необхідно пам'ятати про авторське право та про особливості його застосування в Інтернеті;
* сайт працевлаштування — дає змогу працедавцям розміщувати інформацію щодо вакансій. Потенційний працівник оже її знайти і заповнити анкету чи надіслати резюме, яке відповідає вибраній посаді. Приклад: www.job.ua;
* ігровий сайт — пропонує майданчик для гри. Прикладами можуть бути www.buka.ru, game.km.ru;
* гумористичний сайт — містить анекдоти, жарти, веселі історії чи будь-які інші гумористичні матеріали для розваги (www.kulichki.ru, www.exler.ru);
* інформаційний сайт — призначений для інформування відвідувачів, але необов'язково з комерційною метою. Більшість урядових, освітніх та неприбуткових організацій мають інформаційні сайти, наприклад. Львівська міська рада (www.city-adm.lviv.ua);
* сайт новин — схожий на інформаційний, однак призначений для поширення новин та коментарів до них (www.bbc.co.uk, www.korespondent.net);
* сайт відгуків — на ньому відвідувачі можуть залишити відгуки про різноманітні продукти чи послуги (www.ayda.ru — відгуки про відпочинок, www.mrqe.com — рецензії на фільми);
* пошуковий сайт — дає змогу виконувати пошук інформації чи посилань в Інтернаті, наприклад www.google.com;
* особистий сайт — підтримується однією людиною або групою людей (наприклад, сім'єю); містить різну інформацію, яку автори вважають за потрібне розмістити;
* веб-портал — універсальний сайт, через який можна вийти на інші ресурси Інтернету (www.bigmir.net);
* вікі-сайт — сайт, який редагують кілька чи багато людей. Наприклад, вільна енциклопедія uk.wikipedia.org. Кожен відвідувач може зареєструватись у ній і внести інформацію про певний термін, людину, географічний об'єкт, подію тощо.

Деякі сайти можна віднести до кількох категорій одночасно. Наприклад, на сайті електронної комерції може рекламуватися певна продукція, а також він може містити інформаційні документи, звіти тощо.

Візуальна класифікація:

* текстові сайти містять переважно текстову інформацію та відносно мало зображень (рис. 3.2). Вони порівняно невеликі за розміром і мають простий дизайн. Дуже часто сайти інших типів крім стандартної мають ще й текстову версію — для тих відвідувачів, які з різних причин не хочуть завантажувати повні варіанти веб-сторінок;

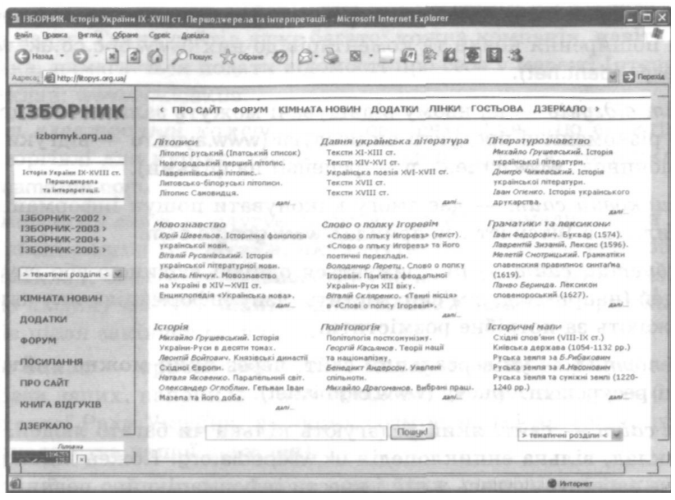


Рисунок 3.2 – Представлення сайту електронної бібліотеки давньої української літератури litopys.org.ua

* графічні сайти окрім певних текстових матеріалів містять багато графіки: спливаючі меню, піктограми, анімаційні зображення тощо  
  (рис. 3.3);
* імітаційні сайти запозичують ідею оформлення з реального світу. Так, наприклад, сайт про автомобілі може містити програму, що відображає всі деталі автомобіля на екрані та дає огляд з позиції водія у процесі руху (рис. 3.4).

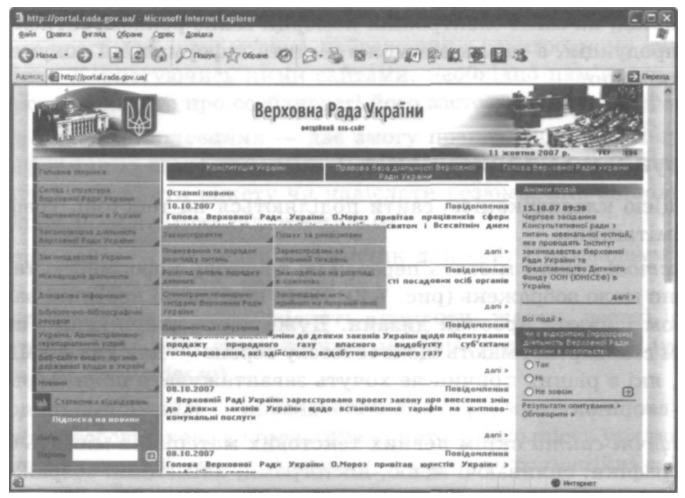


Рисунок 3.3 – Представлення сайту Верховної Ради України [www.racla.gov.ua](http://www.racla.gov.ua)

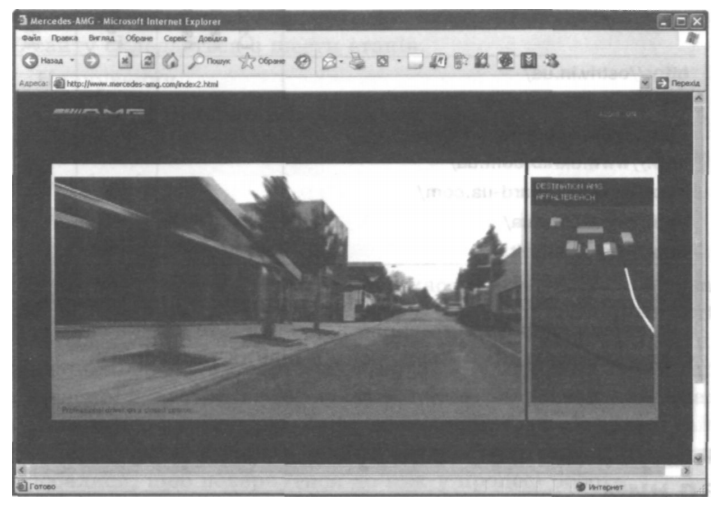


Рисунок 3.4 – Представлення сайту компанії Mersedes: mersedes-amg.com

Класифікація сайтів непроста та неоднозначна. Передусім можна по-різному визначати критерії, за якими класифікують сайти. Крім того, іноді складно визначити, до якої категорії належить той чи інший сайт. [13].

Веб-інтерфейс розроблювального програмного продукту можна віднести за тематичною класифікацією до сайтів баз даних тому, що він призначений для пошуку статистичної інформації, а за візуальною класифікацією більше відповідає графічному сайту так, як містить у своєму контенті досить багато графіків та піктограм.

### 3.1.2 Класифікація веб-дизайну

Перед початком розробки дизайну необхідно знати, який вид дизайну можна застосувати для того, щоб сайт був стильним. Стиль - це система візуальних елементів, покликана забезпечити цілісність сприйняття сторіночки або всього сайту. Виходячи з розробленої концепції і контенту (текстового матеріалу, готових ілюстрацій і малюнків), підбираються шрифти, колір фону, спосіб обробки картинок, колажів, іконок та інших елементів.

Виділяються наступні елементи при створенні стилю :

* шрифт. В межах публікації шрифт повинен мати однакові характеристики ( гарнітура, кегль, колір);
* абзац. В межах публікації переважний єдиний вид вирівнювання;
* колірна схема. Єдине оформлення створить у відвідувача відчуття зв'язності сайту;
* графічне оформлення має укладатися в загальну колірну схему. Використовуючи в оформленні фотографії в якості ілюстрацій, необхідно зробити грамотну тонову і колірну корекцію, кадрування, знайти спосіб обробки країв;
* логотип.

На даний момент в залежності від контексту можна виділити наступні типи веб-дизайну:

* текстовий дизайн. Майже повна відсутність зображень, що значно прискорює завантаження сторінки;
* поліграфічний дизайн. Веб-сторінка імітує друковане видання. Подібний дизайн поширений на корпоративних і рекламних сайтах, тобто там, де необхідно образно-емоційне наповнення змісту. Як правило, основне враження користувач отримує за рахунок піксельної графіки;
* інтерфейсний дизайн. Мінімізація коду і гранична оптимізація зображень, ретельно продумана навігація і загальна схема сайту;
* динамічний дизайн. Активне використання динамічних об'єктів, зокрема, анімації;
* адаптивний дизайн. Дизайн веб-сторінок, що забезпечує відображення сайту на різних пристроях, підключених до інтернету.
* змішані типи - комбінація представлених типів дизайну.

Для реалізації «Менеджеру статистики локомотивів» було обрано адаптивний дизайн. Метою цього є універсальність веб-сайту для різних пристроїв. Для того, щоб веб-сайт було зручно переглядати з пристроїв різних роздільних здатностей та форматів, за технологією адаптивного веб-дизайну не потрібно створювати окремі версії веб-сайту для окремих видів пристроїв. Один сайт може працювати на смартфоні, планшеті, ноутбуці та телевізорі з виходом в інтернет, тобто на всьому спектрі пристроїв.

### 3.1.3 Проектування макету сторінок

Перш ніж створювати сторінки в HTML-редакторі, потрібно зробити макет майбутньої сторінки. Макет - це основа верстки, той каркас, на якому збираються елементи сторінки та інформаційне наповнення. Його можна попередньо створити на листі паперу або відразу в графічному редакторі, це залежить від звички та досвіду дизайнера.

Макет дозволяє створити цілісну картину елементів сторінки, єдність та відчуття неподільного образного ряду. Макет впорядковує структуру сторінки, робить її зручною та зрозумілою сприйняття. Іноді, макетом сайту називають готовий зверстаний шаблон сторінки сайту.

Макет веб-сайту (каркас) не відображує деталі естетичного оформлення, а має відношення тільки лише до інформації основних сторінок, показуючи сиру навігацію, положення текстів, графіки, ключові заголовки і будь-які інші елементи, які повинні знаходитися на сторінці. Макети відображають деяку ієрархію інформації, але не диктують точну форму і місце її подання.

Перш ніж почати створення макету, необхідно визначити кількість принципово різних сторінок у майбутньому сайті.

Якщо перша сторінка сайту зовні дещо відрізняється від решти сторінок, тоді створення макету для сайту зводитися до планування однієї сторінки і єдиного шаблону, за яким будуть створено всі внутрішні сторінки. Всі сторінки майбутнього сайту повинні бути виконані в єдиному стилі, хіба що компоновка текстової області на головній сторінці може бути дещо складнішою, ніж у внутрішніх. Для такого сайту створюють один-два макети.

Якщо перша сторінка сайту суттєво відрізняється від внутрішніх, або сторінки розділів також різняться за наявності різних типів матеріалів. Тоді, звісно потрібно створення кількох макетів для одного сайту, причому стилістично схожих між собою.

Веб-сторінка фактично розглядається як набір прямокутних блоків, які складаються в певному порядку. Завдання макетування полягає у гармонійному розташуванні різноманітних текстових та графічних блоків майбутньої сторінки.

За сталими традиціями інформація розподіляється наступним чином:

В верхній частині сторінки (шапка, хедер, header) містяться:

* + логотип.
  + заголовок.
  + слоган.
  + телефон.
  + вибір мовної версії.
  + меню навігації (по сайту).

В середній (основній) частині сторінки:

* + меню навігації по тематичних розділах сайту.
  + основна інформація.
  + зображення, банери.

В нижній частині сторінки (підвал, футер, footer):

* + копірайти.
  + адреси, телефони.
  + лічильники і банери.
  + додаткове меню навігації (по сайту, по розділах сайту).

Макети сторінок програмного продукту представлені на рисунках 3.5 – 3.10.

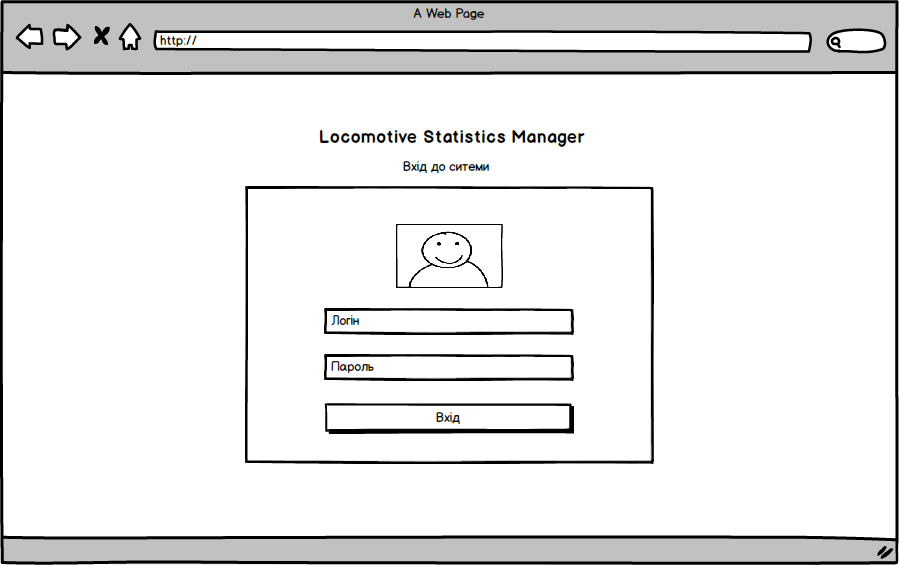


Рисунок 3.5 – Макет форми входу

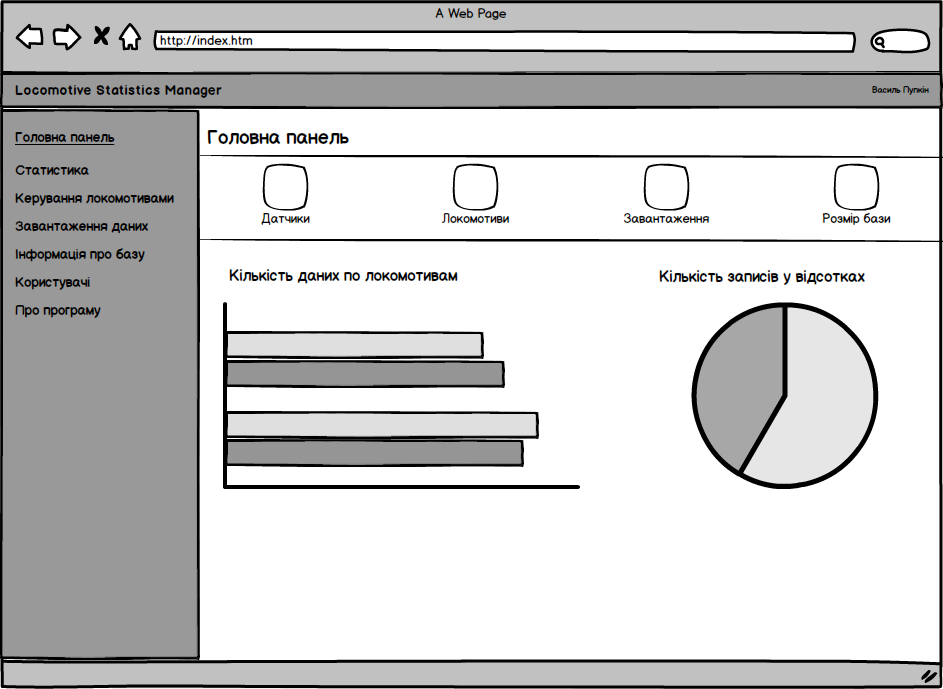


Рисунок 3.6 – Макет головної сторінки

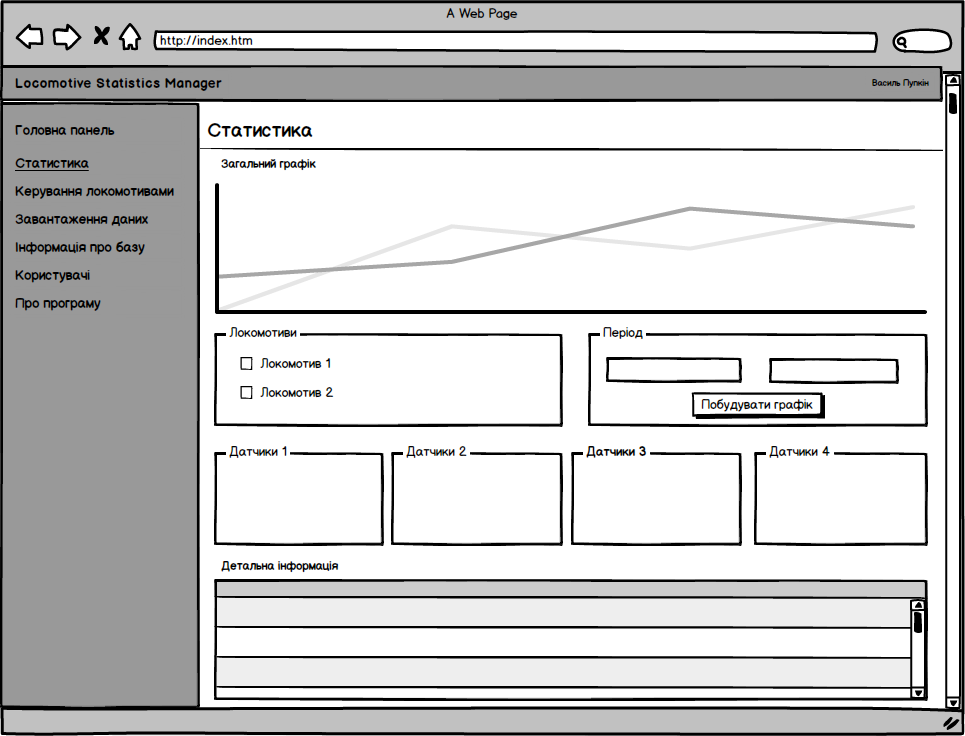


Рисунок 3.7 – Макет сторінки «Статистика»

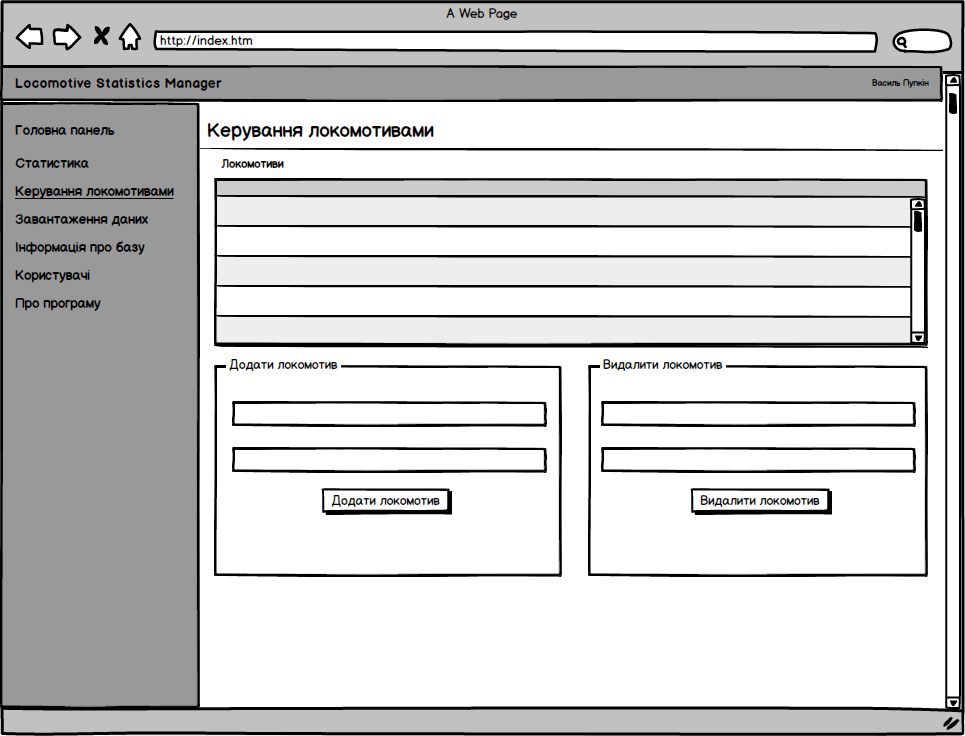


Рисунок 3.7 – Макет сторінки «Керування локомотивами»

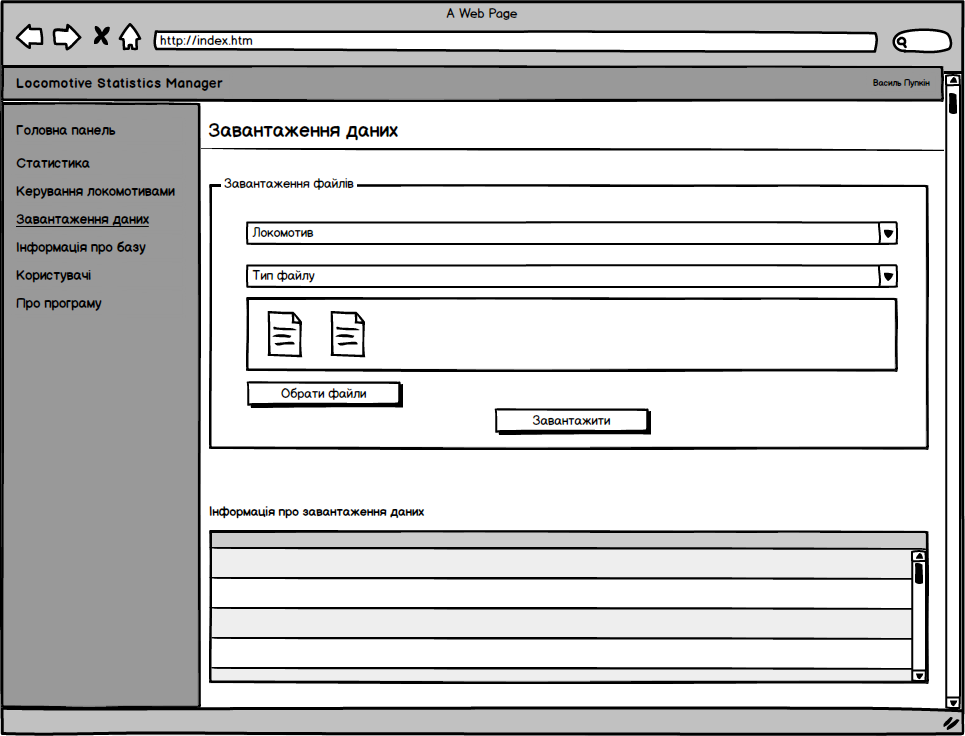


Рисунок 3.7 – Макет сторінки «Завантаження даних»

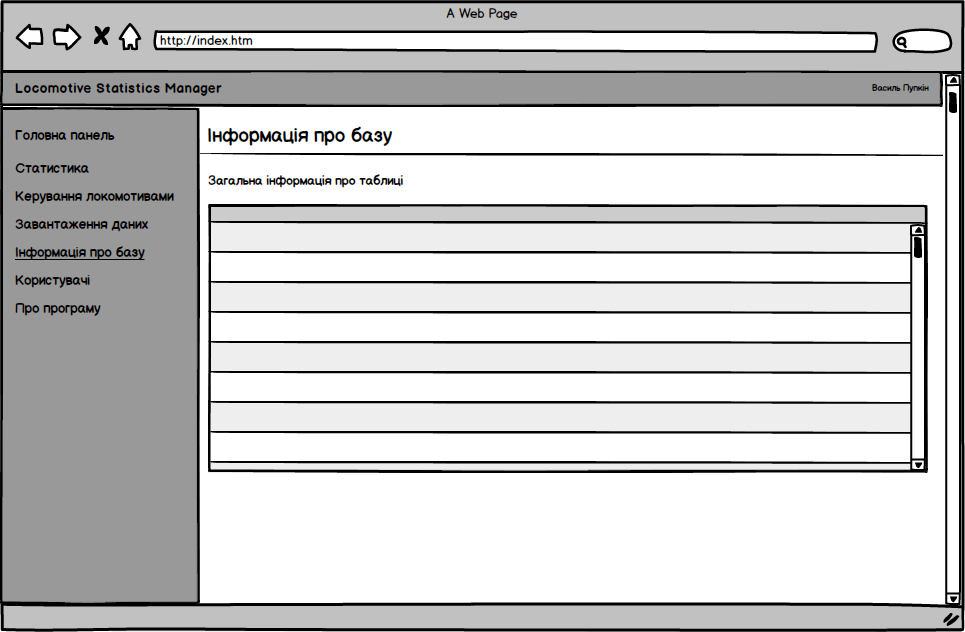


Рисунок 3.8 – Макет сторінки «Інформація про базу»

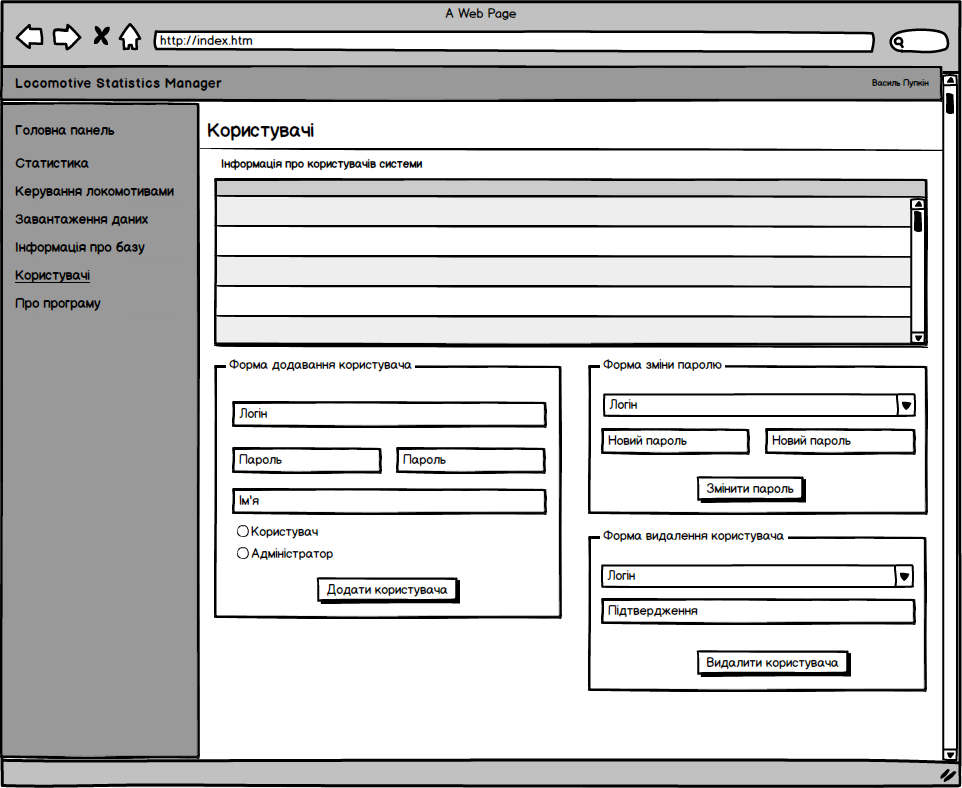


Рисунок 3.9 – Макет сторінки «Користувачі»

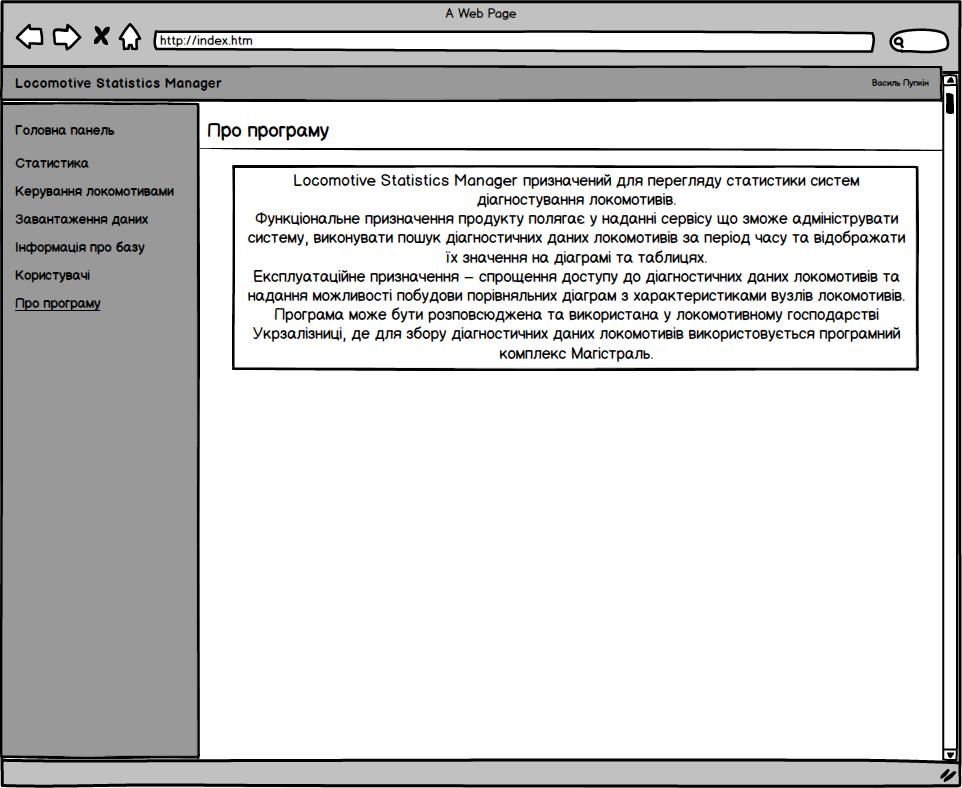


Рисунок 3.9 – Макет сторінки «Про програму»

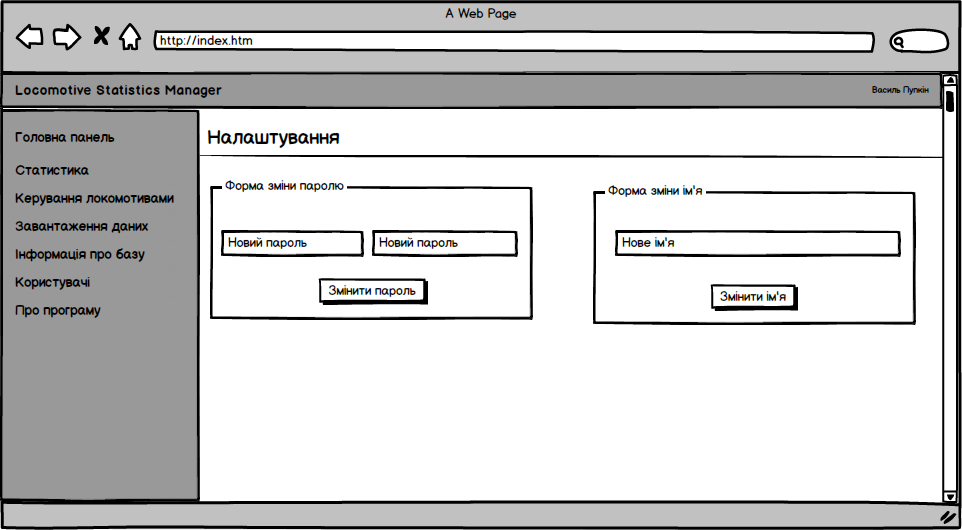


Рисунок 3.10 – Макет сторінки «Налаштування»

### 3.1.4 Верстання сторінок

По закінченні роботи зі створення графічного макету дизайну і схвалення його іншими учасниками проекту чи замовником, приступають до створення HTML-шаблону сторінки.

Верстання сторінок – це процес написання HTML чи XHTML коду сторінки, при якому сторінка набуває вигляд, подібний до дизайну макету. Безумовно, існують і інші технології розмітки тексту, які підтримуються браузерами, проте, такі речі не є поки що популярними. Найпоширенішим варіантом є верстання за допомогою мови HTML та стилів CSS.

До HTML верстки можна застосувати два підходи щодо розподілення елементів сегментів у різних місцях.

Таблична верстка — умовна назва методу верстання HTML-документів, при якому за структурну основу для розташування текстових чи графічних елементів документа використовуються таблиці (HTML-тег <table>).

Метод широко застосовувався до появи стандарту CSS, оскільки на той момент не було іншої можливості точно розташувати елементи на сторінці. Таблиці є спроможними автоматично змінювати свої розміри відповідно до вмісту, і навпаки, тримати точні розміри певних комірок, дозволяють швидко і зручно розставити по комірках різнорідну текстову чи графічну інформацію.

Таблиці в HTML можуть бути вкладеними, що дозволяє створювати цілі ієрархії таблиць при верстанні складних сторінок, окремі елементи яких мають зберігати своє розташування і розмір на екрані незалежно від розміру вікна браузера, тоді як інші елементи, навпаки, мають змінюватися в розмірах або змінювати своє розташування щодо решти об'єктів документа.

Таблична верстка залишається найбільш зручною та простою в засвоєнні, і багато розробників продовжують нею користуватися.

Переваги табличної верстки:

* + - табличні теги з'явилися в стандартах HTML значно раніше за блокові теги, і тому практично всі браузери відображають їх коректно і що є важливим — практично однаково. Цей метод верстки зарекомендував себе, як найбільш стабільний і надійний;
    - при розумному підході таблиці спроможні чітко структурувати інформацію;
    - у випадку, якщо елементи макету є яскраво вираженою таблицею, застосування блокової верстки буде недоцільним.

Недоліки табличної верстки:

* + - у випадку, якщо макет містить багато дрібних деталей, HTML-код виходить занадто громіздким, оскільки він складається в основному з вкладених таблиць;
    - таблична верстка погано поєднується з концепцією позиціонування;
    - таблиці не можуть перекривати одна одну;
    - в табличні теги не можна прописати програмний код скрипта або певний сценарій інтерактивності елементу.

На відміну від табличного способу розташування даних блокова верстка не потребує чіткої прив'язки кожного логічного блоку до певної комірки. Спосіб блокової верстки базується на абсолютно інших принципах розташування і взаємодії за допомогою блокових елементів <div> і <span>. Кожний логічний елемент (текст, картинка, таблиця) представляються як окремі блоки, які також можуть бути вкладеними. Таке розміщення є більш природним потоком.

Характерним моментом для блокової верстки є почергове розташування блоків - один за іншим і чітке розділення між собою. Розташування блоків в одному рядку можливим, але воно використовується лише у випадках, коли виникає потреба добитися особливого ефекту.

Суттю блокової верстки є використання всіх можливостей CSS замість таблиць і скеровуючих зображень.

Переваги блокової верстки:

* + як правило, об’єм коду сторінки, що зверстана блоками є відчутно меншим, ніж об’єм коду сторінки, де застосовано табличну структуру;
  + блокова верстка дуже «дружньо» відноситься до позиціонування;
  + блоки можуть перекривати один одного, причому розробник вказує, який з блоків буде зверху;
  + блокова верстка дозволяє розмістити блок, що відображає будь-яку частину сторінки на початку html-коду. Це буде доречним при застосуванні певних технологій просування сайтів;

Недоліки блокової верстки:

* + блокова верстка є досить складною і потребує професійних навичок верстання;
  + блокову верстку неможливо застосовувати для певних структур сторінок;
  + за низької роздільної здатності монітору блоки можуть зсуватися, перекривати один одного, або закривати доступ до інтерфейсних елементів сайту;
  + блокова верстка має проблеми з сумісністю в різних браузерах;
  + не всі типи браузерів якісно обробляють блоки, особливо старі версії браузерів, що не підтримують стандарти CSS 2.0.

Кажучи про блокову і табличну верстки не можна не згадати про те, що деякі верстальники декларують свою повну байдужість до суперечки між прихильниками різних методів. На їх думку, різниці між таблицями і блоками практично немає, а всі проблеми з вибором певного типу верстки пов'язані виключно з вузькою кваліфікацією виконавців.

Для розробки сторінок програмного продукту була обрана блокова верстка так, як саме вона є оптимальною для виконання цієї задачі бо здатна забезпечити:

* + максимально ідентичне відображення сторінки в різних браузерах;
  + код сторінки має найменший розмір;
  + зрозуміле відображення сторінки в різних роздільних здатностях екрану.

## 3.2 Проектування архітектури системи

Для первинної класифікації був обраний метод оснований на неформальному описі програми – метод CRC-карток. CRC-карта - метод мозкового штурму, призначений для проектування об'єктно-орієнтованого програмного забезпечення [14].

CRC-картки містять наступну інформацію:

* базовий клас;
* похідні класи;
* відповідальність класу;
* зв’язки з іншими класами.

У якості зв’язків показані класи, які використовуються описаним класом.

Були розроблені CRC-картки для класів програми (табл. 3.1 – 3.14).

Таблиця 3.1 – CRC-картка для класу AboutController

|  |  |
| --- | --- |
| AboutController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Про програму» |  |

Таблиця 3.2 – CRC-картка для класу DashboardController

|  |  |
| --- | --- |
| DashboardController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Статистика» |  |

Таблиця 3.3 – CRC-картка для класу DataManagmentController

|  |  |
| --- | --- |
| DataManagmentController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку що до ajax-запитів про отримання статистичних даних локомотивів та керування інформацією про локомотиви в системі. |  |

Таблиця 3.4 – CRC-картка для класу DdInfoController

|  |  |
| --- | --- |
| DdInfoController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Інформація про базу» |  |

Таблиця 3.5 – CRC-картка для класу DownloadDataController

|  |  |
| --- | --- |
| DownloadDataController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Завантаження даних» |  |

Таблиця 3.6 – CRC-картка для класу IndexController

|  |  |
| --- | --- |
| IndexController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Головна панель» |  |

Таблиця 3.7 – CRC-картка для класу IndexController

|  |  |
| --- | --- |
| IndexController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Головна панель» |  |

Таблиця 3.8 – CRC-картка для класу LocomotiveController

|  |  |
| --- | --- |
| LocomotiveController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Керування локомотивами» |  |

Таблиця 3.9 – CRC-картка для класу UsersController

|  |  |
| --- | --- |
| UsersController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Користувачі» |  |

Таблиця 3.10 – CRC-картка для класу LoginController

|  |  |
| --- | --- |
| LoginController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку форми входу |  |

Таблиця 3.11 – CRC-картка для класу UserManagerController

|  |  |
| --- | --- |
| UserManagerController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку що до ajax-запитів на видалення, додавання та зміни користувачів у системі. |  |

Таблиця 3.12 – CRC-картка для класу UserSettingsController

|  |  |
| --- | --- |
| UserSettingsController | |
| Базовий класс | Похідні класи |
| - | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку сторінки «Налаштування» |  |

Таблиця 3.13 – CRC-картка для класу UserAuthManagerImpl

|  |  |
| --- | --- |
| UserAuthManagerImpl | |
| Базовий класс | Похідні класи |
| UserAuthManager | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку видалення, додавання та зміни користувачів. |  |

Таблиця 3.14 – CRC-картка для класу CustomAuthenticationProvider

|  |  |
| --- | --- |
| CustomAuthenticationProvider | |
| Базовий класс | Похідні класи |
| AuthenticationProvider | - |
| Відповідальність | Зв’язки |
| Реалізує бізнес-логіку автентифікації користувачів у системі. |  |

За первинною класифікацією розробляється діаграма класів (Class Diagram).

Діаграма класів — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. [15].

При проектуванні діаграми класів програми (рис. 3.12 – 3.13) було використано шаблон проектування MVC на базі фреймворку Spring MVC   
(рис. 3.11). [16].



Рисунок 3.11 – Реалізація Model-View-Controller у Spring MVC.

Model-View-Controller (MVC) — схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, користувальницький інтерфейс і взаємодія з користувачем розділені на три окремих компонента таким чином, щоб модифікація одного з компонентів надавала мінімальний вплив на інші. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.



Рисунок 3.12 – Діаграма класів для реалізації автентифікації та для керування інформацією про користувачів в системі



Рисунок 3.13 – Діаграма класів для реалізації прошарку контролерів

## 3.3 Використані технології

При розробці системи були використані наступні технології:

* мова програмування Java;
* spring framework;
* spring security
* інструмент для управління Java проектами maven;
* highcharts;
* twitter bootstrap;
* jQuery;
* thymeleaf;
* контейнер сервлетів Apache Tomcat;
* середовище розробки IntelliJ IDEA.

### 3.3.1 Вибір мови програмування

Виходячи з вимог поставлених до програмного комплексу було прийняте рішення використовувати мову Java. Це зумовлене тим, що середовище Java існує для більшості операційних систем, що не обмежує користувача використанням продукту від одного поставника. Вибір Java також зумовлений тим, що це стандартом де-факто для розробки промислових додатків через офіційні гарантії зворотної сумісності, сталий синтаксис та високо продуктивність при наявності необхідних ресурсів.

Для розробки було обрано JDK 7 (Java Development Kit, комплект розробки Java). JDK7, на даний момент, є стабільною версією що підтримується розробником, корпорацією Oracle. Важливим також є підтримка зі сторони середовища останньої версії GC (Garbage Collector, збирач сміття – підсистема середовища, що відповідає за автоматичне звільнення пам’яті) що може працювати майже без зупинки додатку, та нової підсистеми вводу-виводу Java NIO (Java New Input/Output, новий ввід/вивід Java – набір бібліотек для неблокуючого вводу виводу), що дозволяє підвищити продуктивність роботи системи при роботі з великими файлами, що є актуальним для розроблюваного продукту.

Роботу серверу було вирішено базувати на технології Java Servlet API, що гарантує можливість виконання додатку в будь якому сумісному з JDK7 контейнері сервлетів.

Комплексну підтримку архітектурних рішень в додатку покладено на Spring Framework 3.3, що забезпечує наявність механізмів для зворотнього контролю залежностей (Dependency Injection) та декларативного керування транзакціями (Declarative Transactions). Spring Framework є стандартом для розробки серверних промислових додатків Java.

Для реалізації веб-інтерфейсу на стороні клієнта обрано HTML, CSS, JavaScript. Ця зв’язка дозволяє зробити сторінки сайту більш динамічними.

JavaScript зазвичай використовується, як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах, як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси:

* динамічна типізація;
* слабка типізація;
* автоматичне керування пам'яттю;
* прототипна програмування;
* функції як об'єкти першого класу.

Для розробки додатку було обрано середовище IntelliJ IDEA 13.0. Це зумовлено наявністю якісної підтримки мови Java, HTML, CSS, JavaScript, та супутніх технологій, засобів відлагодження, тестування та профілювання.

### 3.3.2 Загальна характеристика Spring Framework.

Spring framework - це фреймворк з відкритим вихідним кодом, він надає набір легковагих інструментів які полегшують рішення завдань у додатках корпоративного масштабу. Власне усунення складності розробки корпоративних додатків і можливість використання простих компонентів JavaBean є метою створення даного фреймворка. Варто зазначити що область застосування даного фреймворка не обмежується розробкою програмних компонентів, що виконуються на серверній стороні додатків. Будь-який додаток може використовувати Spring для усунення сильної пов'язаності, і полегшення тестування.

У додатках на основі Spring використовувані об'єкти зберігаються в контейнері. Контейнер (знаходиться в ядрі Spring Spring Framework) створює об'єкти і пов'язує їх один з одним, конфігурує і управляє їх життєвим циклом (від створення за допомогою оператора new до методу finalize ()). Для управління компонентами контейнер використовує впровадження залежностей (Dependency Injection).

Spring framework має не один контейнер. До його складу входить кілька реалізацій контейнерів:

* + - фабрика компонентів (bean factories);
    - контекст додатків (application contexts).

В основному використовуються контексти додатків, фабрики компонентів є занадто низькорівневим інструментом (прив'язка на код). Контексти додатків прив'язують на конфігураційні файли (ApplicationContext.xml, SecurityContext.xml).

Крім перерахованих вище можливостей можна сказати що для фреймворка існує ціла екосистема проектів, які розширюють можливості застосування Spring на такі області як веб-служби, OSGi, Flash і. NET.

### 3.3.3 Загальна характеристика Spring Security.

Spring Security це Java / JavaEE framework, що надає механізми побудови систем аутентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework. Проект був розпочатий Беном Алексом (Ben Alex) наприкінці 2003 року під ім'ям «Acegi Security», перший реліз вийшов у 2004 році. Згодом проект був поглинений Spring'ом і став його офіційним дочірнім проектом. Вперше публічно представлений під новим ім'ям Spring Security 2.0.0 в квітні 2008 року.

Ключові об'єкти контексту Spring Security:

* SecurityContextHolder, в ньому міститься інформація про поточний контексті безпеки програми, який включає в себе детальну інформацію про користувача (Principal) працюючому в даний час з додатком. За замовчуванням SecurityContextHolder використовує ThreadLocal для зберігання такої інформації, що означає, що контекст безпеки завжди доступний для методів виконуються в тому ж самому потоці. Для того що б змінити стратегію зберігання цієї інформації можна скористатися статичним методом класу SecurityContextHolder.setStrategyName (String strategy).
* SecurityContext, містить об'єкт Authentication і в разі необхідності інформацію системи безпеки, пов'язану із запитом від користувача.
* Authentication представляє користувача (Principal) з точки зору Spring Security.
* GrantedAuthority відображає дозволу видані користувачеві в масштабі всього програми, такі дозволи (як правило називаються «ролі»), наприклад ROLE\_ANONYMOUS, ROLE\_USER, ROLE\_ADMIN.
* UserDetails надає необхідну інформацію для побудови об'єкта Authentication з DAO об'єктів програми або інших джерел даних системи безпеки. Об'єкт UserDetailsсодержіт ім'я користувача, пароль, прапори: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled і Collection - прав (ролей) користувача.
* UserDetailsService, Дозволяє отримати з джерела даних об'єкт користувача і сформувати з нього об'єкт UserDetails який буде використовуватися контекстом Spring Security.

Аутентифікація:

* користувачеві буде запропоновано увійти в систему надавши ім'я (логін або email) і пароль. Ім'я користувача і пароль об'єднуються в екземпляр класу UsernamePasswordAuthenticationToken (екземпляр інтерфейсу Authentication) після чого він передається екземпляру AuthenticationManager для перевірки.
* у разі якщо пароль не відповідає імені користувача буде викинуто виключення BadCredentialsException з повідомленням "Bad Credentials".
* якщо аутентифікація пройшла успішно повертає повністю заповнений екземпляр Authentication.
* для користувача встановлюється контекст безпеки шляхом виклику методу SecurityContextHolder.getContext(). SetAuthentication (...), куди передається об'єкт який повернув AuthenticationManager.

### 3.3.4 Загальна характеристика Apache Maven.

Apache Maven - фреймворк для автоматизації збирання проектів, специфікований на XML-мові POM (Project Object Model). Активно використовується у великих проектах і допомагає здійснити процеси компіляції, створення jar, створення дистрибутива програми, генерації документації.

Для платформи Java існують два основні інструменти для збирання: Ant і Maven. На відміну від Apache Ant, Maven забезпечує декларативну, а не імперативну збірку проекту. Тобто, в файлах проекту pom.xml, які використовує для своєї роботи Maven, міститься його декларативне опис, а не окремі команди. Усі завдання з обробки файлів проекту Maven виконує через плагіни.

Головна перевага Maven - це управління залежностями. Рідко які проекти пишуться без використання сторонніх бібліотек. Ці сторонні бібліотеки часто теж в свою чергу використовують бібліотеки різних версій. Maven дозволяє керувати такими складними залежностями, що дозволяє вирішувати конфлікти версій і в разі потреби легко переходити на нові версії бібліотек.

### 3.3.5 Загальна характеристика Highcharts.

Highcharts – бібліотека для створення діаграм написана на JavaScript, дозволяє легко додавати інтерактивні, анімовані графіки на сайт або у веб-додаток. На даний момент Highcharts підтримує велику кількість діаграм лінійних, кругових, блочних і багатьох інших типів.

Highcharts працює з усіма популярними браузерами, включаючи Safari на iPhone. Мінімальна версія для IE становить 6 +. Також браузери підтримують Canvas елемент, і в деяких випадках SVG для графічного рендеринга.

Підтримуються наступні типи виведення графіків:

* line;
* spline;
* area;
* areaspline;
* column;
* bar;
* pie;
* scatter.

Вони можуть бути комбіновані між один одним при виведенні кінцевому користувачеві, і володіють можливістю відключення будь-якого з них в режимі реального часу безпосередньо користувачем для зручності розбірки інформації.

Дані для побудови графіків можуть бути взяті як із самого JS, так з локального файлу, бази даних, або з віддаленого сервера.

Переваги використання Highcharts:

* працює на чистому JS і не потребує будь-яких плагінів або Flash;
* вивід графіків досить простий, і вони можуть бути використані навіть новачками в веб-розробці;
* є збільшення окремих областей;
* підтримка скінів / тем оформлень;
* підтримка tooltip з виведенням інформації;
* у більшості типів чартів є підтримка date-time X-осі;
* розмір бібліотеки ~ 18кб.

### 3.3.6 Загальна характеристика Twitter Bootstrap.

Twitter Bootstrap — це набір інструментів від Twitter (відноситься до класу інструментів: CSS-фреймворк), створений для полегшення розробки web застосунків та сайтів. Він включає CSS та HTML для типографії, форм, кнопок, таблиць, сіток, навігації тощо, а також додаткові розширення JavaScript.

Репозиторій з фреймворком є одним з найбільш популярних на GitHub і, серед інших, його використовують NASA і MSNBC.

Bootstrap використовує найсучасніші напрацювання в області CSS та HTML, тому необхідно бути уважним при підтримці старих браузерів.

Основні інструменти Bootstrap:

* сітки. Наперед задані розміри колонок, які можна відразу ж використовувати, наприклад ширина колонки 90px відноситься до класу .span2, який ми можемо використовувати в CSS описі документа;
* шаблони. Фіксований або ґумовий шаблон документа;
* типографіка. Опис шрифтів, визначення деяких класів для шрифтів таких як код, цитати тощо;
* медіа. Представляє певне управління зображеннями та відео;
* таблиці. Засоби оформлення таблиць, дозволяє додавати функціональність сортування;
* форми. Класи для оформлення не тільки форм але і деяких подій;
* навігація. Класи оформлення для табів, вкладок, сторінок, меню і тулбара;
* алеут. Оформлення діалогових вікон, підказок і спливаючих вікон.

3.3.7 Загальна характеристика jQuery.

jQuery — популярна JavaScript-бібліотека з відкритим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

Основне завдання jQuery – це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів.

Роботу з jQuery можна розділити на 2 типи:

* отримання jQuery-об'єкта за допомогою функції $(). Наприклад, передавши в неї CSS-селектор, можна отримати jQuery-об'єкт всіх елементів HTML, що потрапляють під критерій і далі працювати з ними за допомогою різних методів jQuery-об'єкта. У разі, якщо метод не повинен повертати якого-небудь значення, він повертає посилання на jQuery об'єкт, що дозволяє вести ланцюжок викликів методів згідно концепції текучого інтерфейсу.
* виклик глобальних методів у об'єкту $, наприклад, зручних ітераторів по масиву.

Типовий приклад маніпуляції відразу декількома вузлами DOM полягає у виклику $ функції з рядком селектора CSS, що повертає об'єкт jQuery, що містить деяку кількість елементів HTML-сторінки. Ці елементи потім обробляються методами jQuery.

3.3.8 Загальна характеристика Thymeleaf.

Thymeleaf є Java XML/XHTML/HTML5 шаблонним двигуном, який може працювати як в веб-середовищі (Servlet основі) так і в не веб-середовищах. Найкраще підходить для обслуговування XHTML/HTML5, представляє собою набір веб-додатків на MVC основі, може обробляти будь які XML файли, навіть в оффлайн середовищі. Він забезпечує повну інтеграцію зі Spring Framework.

У веб-додатках Thymeleaf прагне бути повноцінною заміною JSP і реалізує концепцію Natural Templates: файли шаблонів, які можуть бути безпосередньо відкриті в браузерах та відображатися правильно, як веб-сторінки.

Приклад використання Thymeleaf, вивід таблиці з інформацією про локомотиви:

<table class="table table-bordered table-hover table-striped tablesorter">

<thead>

<tr>

<th class="header headerSortDown">№ <i class="fa fa-sort"></i></th>

<th class="header">Номер локомотива <i class="fa fa-sort"></i></th>

<th class="header">Додаткова інформація <i class="fa fa-sort"></i></th>

</tr>

</thead>

<tbody>

<tr th:each="item,iterStat : ${locomotives}">

<td th:text="${iterStat.count}">1</td>

<td th:text="${item.idLoco}">Номер локомотиву</td>

<td th:text="${item.titleLoco}">Описання</td>

</tr>

</tbody>

</table>

3.3.9 Загальна характеристика Apache Tomcat.

Apache Tomcat – контейнер сервлетів з відкритим вихідним кодом, що розробляється Apache Software Foundation. Реалізує специфікацію сервлетів і специфікацію JavaServer Pages (JSP) і JavaServer Faces (JSF). Написаний на мові Java.

Tomcat дозволяє запускати веб-додатки, містить ряд програм для самоконфігурірованія.

Tomcat використовується в якості самостійного веб-сервера, як сервер контенту в поєднанні з веб-сервером Apache HTTP Server, а також в якості контейнера сервлетів в серверах додатків JBoss і GlassFish.

Компоненти:

– сatalina - контейнер сервлетів Tomcat'а. Catalina реалізує специфікацію сервлетів і JavaServer Pages (JSP).

– сoyote - компонент стека HTTP Tomcat'а, який підтримує протокол HTTP 1.1 для веб серверів або контейнера додатків. Coyote прослуховує вхідні з'єднання на певному TCP порту сервера і пересилає запити в механізм Tomcat для обробки запитів і відправляє відповідь назад запитуючій клієнту.

– jasper - механізм JSP Tomcat'а. Tomcat 5.x використовує Jasper 2, який є реалізацією специфікації JavaServer Pages 2.0 Sun Microsystems. Jasper аналізує JSP-файли, щоб компілювати їх в Java код, як сервлети (які можуть бути оброблені за допомогою Catalina). Під час виконання, Jasper може автоматично виявляти зміни JSP-файлу і перекомпілювати.

3.3.10 Загальна характеристика IntelliJ IDEA.

IntelliJ IDEA — комерційне інтегроване середовище розробки для Java від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії "Community Edition" і повнофункціональної комерційної версії "Ultimate Edition", для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію.

Перша версія IntelliJ IDEA з'явилася у січні 2001 року й швидко здобула популярність, як перша Java IDE із широким набором інтегрованих інструментів для рефакторингу, що дозволяла програмістам швидко реорганізовувати сирцевий код програм. Дизайн середовища орієнтовано на продуктивність праці програмістів, дозволяючи їм сконцентруватися на розробці функціональності, тоді як IntelliJ IDEA бере на себе виконання рутинних операцій.

Community версія середовища IntelliJ IDEA підтримує інструменти для проведення тестування TestNG і JUnit, системи контролю версій CVS, Subversion, Mercurial і Git, засоби складання Maven і Ant, мови програмування Java, Java ME, Scala, Clojure і Groovy. Підтримується розробка застосунків для мобільної платформи Android. До складу входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Комерційна версія "Ultimate Edition" відрізняється наявністю підтримки додаткових мов програмування (наприклад, PHP, Ruby, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, UML-діаграм, підрахунок покриття коду, можливістю роботи з фреймворками (Rails, Grails, Google Web Toolkit, Spring, Play Java Web framework і Hibernate), засобами інтеграції з Perforce, Microsoft Team Foundation Server і Rational ClearCase.

# 4 ВІДЛАГОДЖЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

## 4.1 Функціональне тестування

Цей вид тестування перевіряє відповідність реалізованих функцій вимогам, ТЗ, специфікаціям, різним іншим проектним документам і просто очікуванням користувача. Перевіряється кожна з функцій програми і всі вони в комплексі. Досліджуються всі сценарії використання. Перевіряється адекватність збережених і вихідних даних, методи їх обробки, обробка даних, що вводяться, методи зберігання даних, методи імпорту та експорту даних і т.д. залежно від специфіки додатку.

Основний перелік функціоналу, який необхідно протестувати:

* авторизуватися;
* перегляд головної сторінки;
* перегляд сторінки зі статистикою;
* побудова порівняльної діаграми;
* перегляд сторінки з інформацією про локомотиви;
* додавання інформації про новий локомотив;
* видалення інформації про локомотив із системи;
* перегляд сторінки з інформацією про завантаження даних;
* завантажити дані;
* перегляд сторінки з інформацією про базу даних;
* перегляд сторінки з інформацією про користувачів;
* додати нового користувача;
* видалили користувача;
* змінити пароль користувачу;
* перегляд сторінки з інформацією про програму;
* перегляд сторінки з налаштуваннями користувача;
* вихід з системи.

Для кожної функціональної частини було розроблено тест кейси, які представлені у вигляді таблиць із наступними колонками:

* дія;
* очікуваний результат;
* результат тесту.

Основні етапи тестування кожної функціональної частини:

* передумова;
* кроки тестування;
* післяумова.

Тест кейси для кожної функціональної частини відображаються в таблицях (табл. 4.1 – 4.14).

Таблиця 4.1 – Тест кейс для авторизації

|  |  |  |
| --- | --- | --- |
| Назва: | Тест авторизації | |
| Функція: | Авторизація | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com/login.htm | Відкрилася форма входу на сайт | Тест пройдено |
| Кроки тестування: |  |  |
| Заповнити форму авторизації:  Логін: admin;  Пароль: pass. | Данні успішно введені | Тест пройдено |
| Післяумова |  | | |
| Натиснути кнопку «Вхід» | Перехід на головну сторінку | Тест пройдено | |

Таблиця 4.2 – Тест кейс для перегляду головної сторінки

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду головної сторінки | |
| Функція: | Перегляд головної сторінки | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Кроки тестування: |  |  |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту. | Тест пройдено |

Таблиця 4.3 – Тест кейс для перегляду сторінки зі статистикою

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки зі статистикою | |
| Функція: | Перегляд сторінки зі статистикою | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Кроки тестування: |  |  |
| Перейти на сторінку «Статистика» | Сторінка «Статистика» відкрилася та доступна | Тест пройдено |

Таблиця 4.4 – Тест кейс для побудови порівняльної діаграми

|  |  |  |
| --- | --- | --- |
| Назва: | Тест побудови порівняльної діаграми | |
| Функція: | Побудова порівняльної діаграми | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Перейти на сторінку «Статистика» | Сторінка «Статистика» відкрилася та доступна | Тест пройдено |
| Кроки тестування: |  |  |
| Вибрати будь-які два локомотиви зі списку. | Данні введені вірно | Тест пройдено |
| Вибрати період від 06.05.2013 19:26 до 15.06.2014 19:26 | Данні введені вірно | Тест пройдено |
| Вибрати будь-які два датчики. | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Оновити графік» | Графік успішно побудовано | Тест пройдено |
| Післяумова |  |  |
| Побудувати порівняльний графік | Успішно побудовано таблицю, що містить детальну інформацію точок, за якими побудовано графік | Тест пройдено |

Таблиця 4.5 – Тест кейс для перегляду сторінки з інформацією про локомотиви

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки з інформацією про локомотиви | |
| Функція: | Перегляд сторінки з інформацією про локомотиви | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Кроки тестування: |  |  |
| Перейти на сторінку «Керування локомотивами» | Сторінка «Керування локомотивами» відкрилася та доступна | Тест пройдено |

Таблиця 4.6 – Тест кейс для додавання та видалення інформації про локомотиви з системи

|  |  |  |
| --- | --- | --- |
| Назва: | Тест додавання та видалення інформації про локомотиви з системи | |
| Функція: | Додавання та видалення інформації про локомотиви з системи | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |

Продовження таблиці 4.6 – Тест кейс для додавання та видалення інформації про локомотиви з системи

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Перейти на сторінку «Керування локомотивами» | Сторінка «Керування локомотивами» відкрилася та доступна | Тест пройдено |
| Кроки тестування: |  |  |
| Ввести інформацію про локомотив:  Номер локомотиву: 12345  Додаткова інформація: тест | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Додати локомотив» | Інформація про новий локомотив успішно додано | Тест пройдено |
| Вибрати локомотив з номером 12345, ввести для підтвердження «ВИДАЛИТИ» | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Видалити локомотив» | Інформація про локомотив успішно видалена з системи | Тест пройдено |

Таблиця 4.7 – Тест кейс для перегляду сторінки з інформацією про завантаження даних

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки з інформацією про завантаження даних | |
| Функція: | Перегляд сторінки з інформацією про завантаження даних | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Кроки тестування: |  |  |
| Перейти на сторінку «Завантаження даних» | Сторінка «Завантаження даних» відкрилася та доступна | Тест пройдено |

Таблиця 4.8 – Тест кейс для завантаження даних

|  |  |  |
| --- | --- | --- |
| Назва: | Тест для завантаження даних | |
| Функція: | Завантаження даних | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin. Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |

Продовження таблиці 4.8 – Тест кейс для завантаження даних

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Перейти на сторінку «Завантаження даних» | Сторінка «Завантаження даних» відкрилася та доступна | Тест пройдено |
| Кроки тестування: |  |  |
| Обрати перший локомотив зі списку.  Вибрати тип файлу «VL11\_NEW». | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Обрати файли» | Відкрилося діалогове вікно вибору файлів. | Тест пройдено |
| Обрати файл з даними | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Завантажити» | Завантаження даних успішно виконано | Тест пройдено |

Таблиця 4.9 – Тест кейс для перегляду сторінки з інформацією про базу даних

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки з інформацією про базу даних | |
| Функція: | Перегляд сторінки з інформацією про базу даних | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |

Продовження таблиці 4.9 – Тест кейс для перегляду сторінки з інформацією про базу даних

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Кроки тестування: |  |  |
| Перейти на сторінку «Інформація про базу» | Сторінка «Інформація про базу» відкрилася та доступна | Тест пройдено |

Таблиця 4.10 – Тест кейс для перегляду сторінки з інформацією про користувачів

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки з інформацією про користувачів | |
| Функція: | Перегляд сторінки з інформацією про користувачів | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |

Продовження таблиці 4.10 – Тест кейс для перегляду сторінки з інформацією про користувачів

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тесту |
| Кроки тестування: |  |  |
| Перейти на сторінку «Користувачі» | Сторінка «Користувачі» відкрилася та доступна. | Тест пройдено |

Таблиця 4.11 – Тест кейс для керування користувачами в системі

|  |  |  |
| --- | --- | --- |
| Назва: | Тест керування користувачами в системі | |
| Функція: | Керування користувачами в системі | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Перейти на сторінку «Користувачі» | Сторінка «Користувачі» відкрилася та доступна | Тест пройдено |
| Кроки тестування: |  |  |
| Ввести дані нового користувача:  Логін: test\_user.  Пароль: test.  Ім’я: тест.  Вибрати групу: «Користувач». | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Додати користувача» | Новий користувач успішно доданий | Тест пройдено |

Продовження таблиці 4.11 – Тест кейс для керування користувачами в системі

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тесту |
| Кроки тестування: |  |  |
| На формі зміни паролю вибрати користувача test\_user.  Ввести новий пароль: 1234 | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Змінити пароль» | Пароль для користувача test\_user успішно змінено | Тест пройдено |
| На формі видалення користувача вибрати користувача з логіном test\_user.  Ввести для підтвердження: «ВИДАЛИТИ». | Данні введені вірно | Тест пройдено |
| Натиснути кнопку «Видалити користувача» | Користувач успішно видалений з системи | Тест пройдено |

Таблиця 4.12 – Тест кейс для перегляду сторінки з інформацією про програму

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки з інформацією про програму | |
| Функція: | Перегляд сторінки з інформацією про програму | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |

Продовження таблиці 4.12 – Тест кейс для перегляду сторінки з інформацією про програму

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Відкрити сторінку «Про програму» | Сторінка «Про програму» відкрилася та доступна | Тест пройдено |

Таблиця 4.13 – Тест кейс для перегляду сторінки з налаштуваннями користувача

|  |  |  |
| --- | --- | --- |
| Назва: | Тест перегляду сторінки з налаштуваннями користувача | |
| Функція: | Перегляд сторінки з налаштуваннями користувача | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Кроки тестування: |  |  |
| Перейти на сторінку «Налаштування» | Сторінка «Налаштування» відкрилася та доступна | Тест пройдено |

Таблиця 4.14 – Тест кейс для виходу із системи

|  |  |  |
| --- | --- | --- |
| Назва: | Тест виходу із системи | |
| Функція: | Вихід із системи | |
| Дія | Очікуваний результат | Результат тесту |
| Передумова: |  |  |
| Відкрити сайт Locomotive Statistics Manager за адресою lsm.com | Відкрилася форма авторизації | Тест пройдено |
| Пройти авторизацію:  Логін: admin;  Пароль: pass. | Авторизація успішно пройдена, відкрилася головна сторінка сайту | Тест пройдено |
| Кроки тестування: |  |  |
| Натиснути кнопку «Вихід» | Вихід із системи успішно виконано | Тест пройдено |

## 4.2 Відлагодження програми

Якщо тестування – діяльність, спрямована на виявлення помилок, то відлагодження спрямоване на встановлення точної природи відомої помилки, а потім – на виправлення цієї помилки. Ці два види діяльності зв'язані – результати тестування є вихідними даними для налагодження.

Відлагодження програми виконувалося з використанням методу індукції та методу просування від місця виникнення помилки до місця помилки. Таким чином, визначивши збійну операцію, можна локалізувати безпосереднє місце помилки.

Налагодження відбувалося шляхом завантаження js-файлів у режимі відладки за допомогою браузера Google Chrome, використовуючи наступні інструменти:

* Call Stack – список кадрів стеку викликів, за якими програма потрапила в поточний стан;
* Watches – інтерфейс що дозволяє виконувати моніторинг значень змінних в поточному контексті відладчика;
* Variables – інтерфейс, що відображає поточні значення змінних, наявних в поточному кадрі стеку;
* Threads – вікно, що відображає список ниток(thread), що активні в віртуальній машині Java. Дозволяє переключитися на стек викликів довільної нитки.

До місця виникнення ймовірної помилки ставиться точка зупину Breakpoint (рис. 4.1), на якій програма зупиниться і чекатиме покрокового виконання. Це дозволяє ініціювати виконання операцій, робота яких призводить до збою, та розпочати відлагодження з місця проблеми.

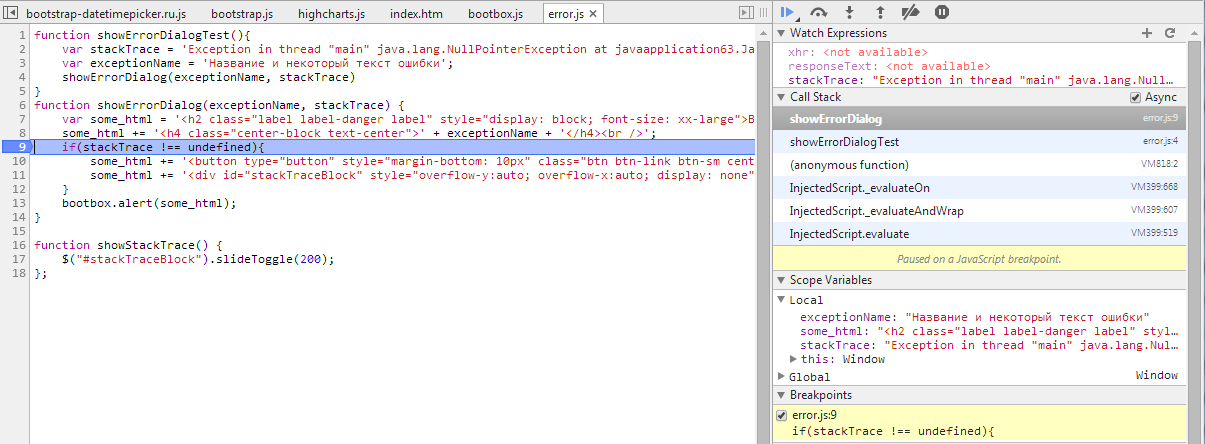


Рисунок 4.1 – Приклад використання точки зупину програми

# 5 ОХОРОНА ПРАЦІ

## 5.1. Аналіз шкідливих та небезпечних виробничих факторів

Відповідно до ДСТУ 2293-99 «Охорона праці. Терміни та визначення основних понять» шкідливі фактори – виробничі фактори, тривалий вплив яких на працюючого у визначених умовах людини приведе до захворювання, зниження працездатності. У залежності від рівня і тривалості впливу шкідливі фактори можуть класифікуватися і як небезпечні.

Умови праці співробітника відділу, який працює з ПЕОМ, повинні відповідати I або II класу відповідно до Гігієнічної класифікації праці показників шкідливості і небезпеки факторів виробничого середовища, ваги і напруженості трудового процесу відповідно до ДСанПІН 3.3.2-007-1998 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Робота на ПЕОМ пов'язана з наступними шкідливими факторами:

* випромінення монітору;
* відблиски на екрані монітора;
* електромагнітне поле.

Випромінення монітору призводить до швидкої стомленості очей, що у свою чергу, призводить до зниження продуктивності праці і може спричинити хронічні захворювання очей.

Відблиски на екрані монітора, що виникають при неправильному освітленні, приводить до погіршення зору, а у випадку тривалого впливу даного небезпечного фактору, може привести до повної втрати зору. З метою зниження рівня впливу на працівника даного шкідливого фактора, варто дотримуватись вимог нормативних документів.

Електромагнітне поле призводить до підвищеної втомлюваності людини, може викликати головний біль. Тривалий вплив електромагнітного поля призводить до погіршення стану здоров’я людини, та може викликати хронічні захворювання.

Відповідно до ДСТУ 2293-99 «Охорона праці. Терміни та визначення основних понять» небезпечні фактори – це виробничі фактори, вплив яких на працюючого у визначених умовах людини, приведе до травми, різкого погіршення здоров'я. До різкого погіршення здоров'я можна віднести отруєння, опромінення, удар електрострумом, тепловий удар та ін.

До небезпечних факторів відносяться:

* електронебезпека;
* пожежонебезпека.

При роботі з ПЕОМ найчастіше трапляються нещасні випадки, пов’язані з ураженням електричним струмом, які викликані дотиком до оголених місць струмоведучих частин устаткування, або частин, що знаходяться під напругою.

Вплив струму на людину залежить від його сили: струм до 0,6мА не відчувається людиною. Струм силою 0,6мА приводить до скорочення м'язів тієї частини, тіла, що піддалася його впливу. Цей струм називається «не відпускає». Значення струму, що перевищує 0,6мА, здатні викликати утрату свідомості і зупинку подиху, а при досягненні струмом порогу 100мА – смерть. При впливі на тіло людини струму в 3-4А виникає обвуглювання ділянок тіла.

Пожежі становлять особливу небезпеку для життя людини, і можуть призвести до великих матеріальних утрат. Під час пожежі людина може отримати опіки різного ступеня тяжкості, а також отруїтися чадним газом. Джерелами загоряння можуть виявитися електронні схеми ПЕОМ, що перегрілися.

## 5.2. Проектні заходи

Для якісної і зручної роботи співробітника відділу з ПЕОМ, необхідне проведення проектних заходів: відповідна облаштованість, належне дотримання ергономічних характеристик основних елементів робочого місця, санітарно-гігієнічних вимог і т.п.

Рівні електромагнітного випромінювання і магнітних полів на робочому місці повинні відповідати вимогам нормативних документів України.

За способами захисту людини від ураження електричним струмом відео-термінали, ПЕОМ, периферійні пристосування ПЕОМ і оснащення для обслуговування, ремонту і налагодження ПЕОМ повинні відповідати I класу захисту згідно нормативних документів з охорони праці чи повинні бути заземлені відповідно до НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів».

Лінія електромережі для живлення ПЕОМ, периферійних пристроїв ПЕОМ і устаткування для обслуговування, ремонту і налагодження ПЕОМ виконується як окрема групова трьох провідна мережа, шляхом прокладки фазового, нульового робочого і нульового захисного провідників.

Підключення на розподільному щиті до одного контактного затиску нульового робочого і нульового захисного провідників заборонено.

Площа перетину нульового робочого і нульового захисного провідника в груповій трьох провідній мережі повинна бути не менш площі перетину фазового провідника.

У приміщенні, де одночасно експлуатується чи обслуговується більше п'яти персональних ПЕОМ, на видному доступному місці встановлюється аварійний вимикач, за допомогою якого можливо зробити знеструмлення приміщення (за винятком освітлення).

При роботі неприпустимо:

* експлуатація кабелів і проводів з ушкодженою чи утративши захисні властивості за час експлуатації ізоляцією;
* використання ушкоджених розеток, розгалужених і сполучних коробок, вимикачів і інших електроприладів, а також ламп, скло яких має сліди чи затьмарення опуклості;
* підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною й іншими займистими матеріалами, експлуатація їх із знятими ковпаками.

Для всіх споруджень та приміщень, у яких експлуатуються ВДТ і ПЕОМ, повинна бути визначена категорія по вибухонебезпечній та пожежній безпеці і клас зони згідно ПУЕ. Відповідні позначення повинні бути нанесені на вхідні двері приміщення.

Будинки та ті їхні частини, у яких розміщуються ПЕОМ, повинні мати не нижче II ступеня вогнестійкості відповідно до НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою». Приміщення для обслуговування, ремонту і налагодження ПЕОМ повинні відноситися:

* по пожежонебезпеки до категорії В – пожежонебезпечні приміщення, де розташовуються тверді горючі речовини (ТГР);
* по класу приміщення до категорії ІІа по ПУЕ.

Неприпустимим є розташування приміщень категорії А и Б, а також виробництв із мокрими технологічними процесами поруч із приміщеннями, де розташовуються ПЕОМ, виконується їхнє обслуговування, налагодження і ремонт, а також над такими приміщеннями і під ними.

Приміщення з ПЕОМ повинні бути оснащені системою автоматичної пожежної сигналізації відповідно до вимог переліку однотипних за значенням об'єктів, що підлягають устаткуванню автоматичними установками пожежогасіння і пожежної сигналізації, затверджених нормативними документами України, з димовими пожежними оповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 м2 площі приміщення з обліком гранично допустимої концентрації вогнегасної рідини відповідно до вимог Правил пожежної безпеки в Україні, вказаних у документі НАПБ А.01.001-2004 «Правила пожежної безпеки в Україні».

Стіни приміщень з ПЕОМ виготовляються з негорючих матеріалів. Підходи до засобів пожежогасіння повинні бути вільними.

По інструкції "По охороні праці для операторів" співробітник відділу планування інструктується перед початком роботи первинним інструктажем.

Первинний інструктаж завжди проводиться на робочому місці з безпосереднім показом робіт (стажування 1 місяць). Потім, через кожні 6 місяців проводиться повторний інструктаж,

Результати інструктажу заносяться в "Журнал реєстрації інструктажів з питань охорони праці". У журналі після проходження інструктажу повинен бути підпис людини, яка інструктувала та співробітника відділу прогнозування.

Площа, виділена для одного робочого місця з ВДТ чи ПЕОМ, повинна складати не менш 6 м2 , а об’єм – не менш 20 м3.

Робочі місця з ВДТ щодо вікон світлових прорізів повинні розташовуватися так, щоб природне світло падало збоку, переважно ліворуч.

При розташуванні робочих місць із ВДТ необхідно дотримувати наступних вимог:

* робочі місця з ВДТ розташовуються на відстані не менш 1 м від стін із вікнами;
* відстань між бічними поверхнями ВДТ повинна бути не менше 1,2 м;
* відстань між тильною поверхнею ВДТ і екраном іншого ВДТ повинна бути не менш 2,5 м;

прохід між рядами робочих місць повинний бути не менше 1 м.

Конструкція робочого місця співробітника при роботі з ВДТ (при роботі сидячи) повинна забезпечувати підтримку оптимальної робочої пози з наступними ергономічними характеристиками: стопи ніг - чи на підлозі, чи на підставці для ніг; стегно - у горизонтальній площині; передпліччя - вертикально; лікті під кутом 70°-90° до вертикальної площини; зап'ястя - зігнуті під кутом не більш 20о щодо горизонтальної площини; нахил голови 15°-20° відносно вертикальної площини,

Висота робочої поверхні столу для ВДТ повинна бути в межах 680-800 мм, а ширина - забезпечувати виконання операцій у зоні досяжності моторного поля.

Розміри столу, які рекомендуються: висота -725 мм, ширина ~ 600-1400 мм, глибина - 800-1000 мм.

Робочий стіл для ВДТ повинний мати простір для ніг висотою не менш 600 мм, шириною не менш 500 мм, глибиною на рівні колін не менш 450 мм, на рівні витягнутої ноги ~ не менш 650 мм.

Можливість обертання екрана ВДТ навколо горизонтальної і вертикальної осі.

Клавіатуру слід розміщувати на поверхні столу або на спеціальній, регульований за висотою, робочій поверхні окремо від столу на відстані 100 - 300 мм від краю, ближчого до працівника. Кут нахилу клавіатури має бути в межах 5 - 15 град.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрану відео термінала, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля: по висоті 900 - 1300 мм, по глибині 400 - 500 мм.

При потребі високої концентрації уваги під час виконання робіт з високим рівнем напруженості суміжні робочі місця з відео терміналами та персональними ЕОМ необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м.

Параметри мікроклімату, іонного складу повітря, вміст шкідливих речовин на робочому місці, оснащеного ВДТ, повинні відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Для підтримки припустимих значень мікроклімату і концентрації позитивних і негативних іонів необхідно передбачити або установити прилади зволоження і (або) штучної іонізації, кондиціювання повітря.

## 5.3. Безпека проці при виконанні робіт на ПЕОМ

### 5.3.1. Вимоги безпеки праці перед початком роботи на ПЕОМ

Підготовка робочого місця повинна виконується відповідно до нижче приведених пунктів:

* + увімкнути систему кондиціювання повітря в приміщенні;
  + оглянути робоче місце і привести його в порядок: переконатися, що на ньому немає сторонніх предметів; все устаткування і блоки ПЕОМ з'єднані із системним блоком за допомогою сполучних шнурів;
  + перевірити надійність установки апаратури на робочому столі. Монітор повинен стояти не на краю столу. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) – та ледь зверху вниз, при цьому екран повинен бути ледь нахилений – нижній його край ближче до співробітника відділу;
  + перевірити загальний стан апаратури, або перевірити чи справні електропроводка, сполучні шнури, штепсельні вилки, розетки, перевірити заземлення захисного екрана;
  + відрегулювати висвітлення робочого місця;
  + відрегулювати і зафіксувати висоту крісла, зручний для співробітника нахил його спинки;
  + включити апаратуру комп'ютера перемикачами на корпусі в послідовності: стабілізатор напруги, монітор, принтер (якщо необхідний друк), системний блок;
  + відрегулювати яскравість світіння екрана до яскравості навколишніх його поверхонь у робочій зоні і не більше 3:1;
  + при виявленні яких-небудь несправностей роботу не починати, повідомити про це керівника робіт.

### 5.3.2. Вимоги безпеки праці під час роботи на ПЕОМ

Користувачі ЕОМ повинні слідкувати за тим, щоб відео термінали, ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ були справними і випробуваними відповідно до чинних нормативних документів.

Щоденно перед початком роботи необхідно проводити очищення екрану відео терміналу від пилу та інших забруднень.

У період роботи за ПЕОМ необхідно передбачати перерви для відпочинку, які мають бути оптимальної тривалості (надто тривалі ведуть до порушення готовності до дії та розладу динамічного стереотипу). Тому, через кожні 40 - 45 хвилин необхідно робити перерву на 3 - хвилини, а через 2 години - на 15 хвилин. Середня сумарна тривалість роботи за ПЕОМ на день не повинна перевищувати 4 годин, а на тиждень - 20 годин.

При використанні захисного екрана або монітора із зниженим випромінюванням час роботи за ПЕОМ може бути збільшено.

Шкідливою є одна й та сама поза протягом тривалого часу. Тому в положенні сидячі треба час від часу змінювати фіксовані робочі пози, робити короткочасні перерви.

Під час роботи на ПЕОМ напружуються м'язи рук. Для підтримки їх тонусу під час перерви рекомендується проводити гімнастичні вправи.

З метою зниження негативного впливу монотонної діяльності доцільно чергувати операції введення тексту і цифрових даних, редагування тексту.

Періодично рекомендується виконувати комплекс вправ для очей:

* дивитися на мітку на віконному склі, що знаходиться на відстані 30 см від очей, потім перевести погляд вдалину (повторити кілька разів);
* рухи очима по колу до 10 разів за годинною стрілкою та навпаки спочатку швидко потім якомога повільніше (повторити вправу з заплющеними очима);
* самомасаж заплющених очей та шкіри навколо очей пальцями.

Є неприпустимими такі дії:

* виконання обслуговування, ремонту та налагодження ЕОМ безпосередньо на робочому місці користувача ЕОМ;
* зберігання біля відео термінала та ЕОМ паперу, дискет, інших носіїв інформації, запасних блоків, деталей тощо, якщо вони не використовуються для поточної роботи;
* відключення захисних пристроїв, самочинне проведення змін у конструкції та складі ЕОМ, устаткування або їх технічне налагодження;
* робота з відео терміналами, в яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані тощо;
* праця на матричному принтері зі знятою (трохи піднятою) верхньою кришкою.

Вимоги безпеки під час обслуговування, ремонту та налагодження ЕОМ

Монтаж, обслуговування, ремонт та налагодження ЕОМ, заміна деталей, пристроїв, блоків повинні здійснюватись тільки при повному відключенні живлення.

Забороняється з'єднувати та роз'єднувати кабелі при підключеній напрузі.

У тих випадках, коли монтаж, обслуговування, ремонт та налагодження ЕОМ або її пристроїв, блоків при відключеному живленні неможливі, виконання цих робіт допускається за умови додержання таких вимог:

* устаткування, допоміжна апаратура та прилади повинні бути заземлені;
* роботи виконуються не менше ніж двома працівниками;
* працівники повинні виконувати роботу інструментом з ізольованими ручками, стоячи на діелектричному килимку, або бути в діелектричних калошах.

Засоби захисту та інструмент необхідно щоразу перед застосуванням оглянути і при виявленні несправностей негайно замінювати.

Користування несправними захисними засобами та інструментом є неприпустимим.

Не можна користуватися біля ПЕОМ аерозолями (дезодорантами, тощо).

Не допускати попадання води та інших рідин в середину пристроїв комп'ютера.

При наявності електроструму на корпусі припинити роботу, вимкнути ПЕОМ від електромережі, сповістити керівника.

У випадку виникнення у співробітника зорового або дискомфорту інших неприємних суб'єктивних відчуттів, що настають, незважаючи на проходження санітарно-гігієнічних і ергономічних вимог, режимів роботи і відпочинку, варто застосувати індивідуальний підхід в обмеженні часу робіт і корекції тривалості перерв для відпочинку, або проводити заміну іншими видами робіт (не пов'язаних з використанням комп'ютера).

### 5.3.3. Вимоги безпеки праці після закінчення роботи на ПЕОМ

При завершенні роботи користувач повинен:

закінчити і записати в пам'ять комп'ютера файл, що знаходився в роботі. Вийти з програмної оболонки і повернутися в середовище операційної системи;

вимкнути системний блок, принтер, інші периферійні пристрої (якщо вони підключені до комп'ютера), вимкнути монітор. Вимкнути стабілізатор живлення, якщо комп'ютер підключений до мережі через нього. Штепсельні вилки витягнути з розетки. Накрити клавіатуру кришкою для запобігання потрапляння на неї пилу;

прибрати робоче місце. Забрати усі необхідні документи (чи покласти їх у шухляду);

доповісти керівнику про всі виявлені недоліки в роботі ПЕОМ.

### 5.3.4. Вимоги безпеки праці в аварійних ситуаціях

В разі ознак горіння (диму, запаху гару, тощо), припиненні подавання електроенергії або виявленні будь-яких несправностей необхідно негайно вимкнути ПЕОМ з електромережі. Якщо є потерпілі – надати їм першу медичну допомогу, викликати швидку допомогу за телефоном «103» або за телефоном найближчої медичної допомоги.

Якщо сталася пожежа, викликати пожежну частину за телефоном «101» та приступити до гасіння наявними засобами пожежогасіння. При виникненні аварійної ситуації виконувати всі вказівки керівника робіт по її усуненню.

# ВИСНОВКИ

В результаті виконаної роботи було розроблено програмний засіб, що надає програмні інтерфейси для перегляду статистичної інформації локомотивів.

Програмний продукт створено у вигляді веб-сервісу, для спрощення його обслуговування та модернізації з використанням інтерактивних діаграм, що дозволяє універсальний віддалений доступ до даних кінцевому користувачу.

В процесі розробки використовувалися новітні підходи до розробки програмного забезпечення. Проектування програми виконувалося за допомогою мови формального опису UML. Були також розроблені прототипи всіх веб-сторінок проекту.

Всі поставлені задачі було виконано в повному обсязі. Для розробки продукту було використано новітні технології для побудови промислових Web-сервісів, було вибрано мову програмування Java. Це забезпечує можливість роботи програмного продукту на будь якому сервері що підтримується Java. Також такий підхід зводить до мінімуму збитки від помилок в програмному продукті – у випадку помилки програма продовжить роботу, завершивши з помилкою лише збійний бізнес-процес.

Аналіз роботи продукту дозволить уточнювати перелік контрольованих параметрів, удосконалювати алгоритми контролю, періодичність опитування датчиків контролюючих параметри вузлів і агрегатів локомотивів, а також використовувати інформацію при розробці й проектуванні вузлів локомотивів для забезпечення їх придатності до контролю.

# ЛІТЕРАТУРА

1. Боднарь, Б.Е. Повышение эксплуатационной надежности подвижного состава: Развитие систем диагностирования [Текст] / Б.Е. Боднарь , А.Б. Очкасов // Локомотив-информ. – 2011.- №1-2. - С.56-58
2. Осяев, А.Т. О системе обслуживания локомотивов за рубежом [Текст] / А.Т. Осяев, В.А. Никифоров.// Вестник ВНИИЖТа. – 2012,- №5.- С.56 – 62.
3. Горский, А.В. Стратегия интелектуального ремонта локомотивов [Текст] / А.В. Горский, А.А. Воробьев, А.В. Скребков // Локомотив. – 2012. - №7.- С.33-35.
4. Боднарь, Б.Е. Использование диагностической информации при разработке системы управления техническим состоянием локомотивов [Текст] / Б.Е. Боднарь, Е.Б Боднарь, А.Б.Очкасов // Локомотив-информ. – 2011.- №3-4. - С.10-13.
5. ГКИУ. 468262.006 РЭ. Система диагностики «МАГИСТРАЛЬ-ДЭ1М». Руководство по эксплуатации.- Днеп.: УЭлНИИ. 1996.
6. Горский А.В., Воробьев А.А., Куанышев Б.М. Ремонт - только по результатам диагностики / Локомотив.- 1998.- № 12.- С. 37-39.
7. Бадьян И.О. Аппаратура микропроцессорной системы управления и диагностики электровоза // Системы транспортной автоматики.- 2004.- № 4.- С. 48-52.
8. Фаулер М. UML. Основы. / М. Фаулер, К. Скотт; пер. с англ. А. Леоненков – СПб: Символ-Плюс, 2002. – 192 с.
9. «ООП» [Електронний ресурс] — Режим доступу: <http://ru.wikipedia.org/wiki/%D0%9E%D0%9E%D0%9F>.
10. «Functional programming» [Електронний ресурс] — Режим доступу: <http://en.wikipedia.org/wiki/Functional_programming>.
11. «Client–server model» [Електронний ресурс] — Режим доступу: http://en.wikipedia.org/wiki/Client%E2%80%93server\_model.
12. «Веб-інтерфейс» [Електронний ресурс] — Режим доступу: <http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81>.
13. Основи веб-дизайну / О. Г. Пасічник, О. В. Пасічник, І. В. Стеценко : [Навч. посіб.]. — К.: Вид. група BHV. — 2009. — 336 с: іл.
14. «CRC-карта» [Електронний ресурс] — Режим доступу: <http://ru.wikipedia.org/wiki/CRC-%D0%BA%D0%B0%D1%80%D1%82%D0%B0>.
15. «Диаграммы UML» [Електронний ресурс] — Режим доступу:  
    <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl8/gl8.html>.
16. «Web MVC framework» [Електронний ресурс] — Режим доступу: http://docs.spring.io/spring/docs/current/spring-framework-кeference/html/mvc.html
17. ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».
18. НПАОП 0.00-1.31-99 «Правила охорони праці під час експлуатації електронно-обчислювальних машин».
19. ДСТУ 2293-99 «Охорона праці. Терміни та визначення основних понять»
20. НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків .та зовнішніх установок за вибухопожежною та пожежною небезпекою».
21. ДБН В.2.5-56:2010 «Пожежна автоматика будинків і споруджень».
22. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку».

ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

# ДОДАТКИ