**ЗМІСТ**

[**ВСТУП** 2](#_Toc506287006)

[**1.** **ОГЛЯД ПІДПРИЄМСТВА** 3](#_Toc506287007)

[**1.1** **СТРУКТУРА ТА НАПРЯМКИ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА** 3](#_Toc506287008)

[**1.1.1** **Загальна інформація про підприємство** 3](#_Toc506287009)

[**1.1.2** **Опис установи** 3](#_Toc506287010)

[**1.2** **КОМП’ЮТЕРИЗОВАНІ ЗАСОБИ ПІДТРИМКИ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА** 3](#_Toc506287011)

[**1.2.1** **Робоче місце при проходженні практики** 3](#_Toc506287012)

[**1.2.2** **Опис комп'ютерної техніки і різновидів використовуваного програмного забезпечення** 3](#_Toc506287013)

[**2.** **ВИКОНАННЯ ЗАВДАННЯ ПРАКТИКИ** 4](#_Toc506287014)

[**2.1** **Налаштування робочого середовища** 4](#_Toc506287015)

[**2.2** **Зміст завдання** 4](#_Toc506287016)

[**2.3** **Виконання поставленого завдання** 4](#_Toc506287017)

[**2.3.1** **Сторінка Login** 5](#_Toc506287018)

[**2.3.2** **Сторінка Drivers** 6](#_Toc506287019)

[**2.3.3** **Сторінка Logs** 8](#_Toc506287020)

[**3.** **Індивідуальне завдання від університету** 10](#_Toc506287021)

[**3.1** **Система контролю версій** 10](#_Toc506287022)

[**3.2** **Налаштування власного GIT-сервера** 10](#_Toc506287023)

[**ВИСНОВОК** 14](#_Toc506287024)

# **ВСТУП**

Метою виробничої практики є застосування студентом набутих навичок на підприємстві та здобуття нових за спеціальністю “розробка програмного забезпечення”.

Керівник підприємства, після ознайомлення студента зі структурою закладу та процесом роботи, дає індивідуальне завдання, яке студент має виконати в межах даної практики.

Основним завданням розробника є проектування, реалізація та тестування програмних продуктів.

Практика проходила під керівництвом штатного працівника фірми. Керівником практики було поставлене наступне завдання для студента: спроектувати, реалізувати програмними засобами та протестувати веб-додаток.

1. **ОГЛЯД ПІДПРИЄМСТВА**
   1. **СТРУКТУРА ТА НАПРЯМКИ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА**

### **Загальна інформація про підприємство**

ФОП “TAXI ELIT” – підприємство, яке займається перевезенням пасажирів на замовлення. Діяльність даного підприємства тісно пов’язана з різними сферами життя людини. Особливо важливою виступає роль ІТ, тому що активний розвиток молодого підприємства передбачає використання web-додатків та програмних комплексів для маркетингу і ефективнішої роботи в сучасних реаліях.

### **Опис установи**

Підприємство ФОП “TAXI ELIT” було створене у 2017 році. В офісній будівлі знаходиться орендований офіс, де працюють два диспетчери. Також в компанії є 15 штатних водіїв та близько 20 приватних перевізників, які співпрацюють з підприємством на контрактній основі.

* 1. **КОМП’ЮТЕРИЗОВАНІ ЗАСОБИ ПІДТРИМКИ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА**

### **Робоче місце при проходженні практики**

Робоче місце складалося з комп’ютерного столу, ноутбука ASUS X552MD, мишки. В наявності було також підключення до інтернету.

### **Опис комп'ютерної техніки і різновидів використовуваного програмного забезпечення**

На практиці використовувалась така комп’ютерна техніка: ноутбук із характеристиками, представленими в табл. 1:

На робочій станції встановлена ліцензійна операційна система Microsoft Windows 10.

Таблиця 1 – Характеристики комп’ютера

|  |  |
| --- | --- |
| Процесор | Intel(R) Celeron(R) CPU N2830 @ 2.16GHz |
| Материнська плата | ASUS X550MD |
| Оперативна пам'ять | 4GB |
| Відеокарта | Intel Bay Trail  nVIDIA GeForce 820M (Asus) |
| Жорсткий диск | 500 GB TOSHIBA MQ01ABF050 |

1. **ВИКОНАННЯ ЗАВДАННЯ ПРАКТИКИ**
   1. **Налаштування робочого середовища**

Одним із завдань практики було встановити на робочу станцію таке програмне забезпечення:

* + - PHPstorm 2017.1.4 (для розробки продукту на мовах HTML, CSS, JS, PHP);
    - XAMPP v3.2.2 (для тестування web-додатка в процесі розробки)
  1. **Зміст завдання**

Керівником практики було поставлено завдання розробити web-додаток для моніторингу кількості водіїв та їх статусу. Додаток має містити базу даних водіїв та їх статус.

Доступ до налаштувань додатку та внесення нових водіїв має лише адміністратор.

## **Виконання поставленого завдання**

Для реалізації завдання була обрана IDE PHPstorm. Розробка виконувалася у складі 1 студента та штатного web-розробника. Декомпозицію проекту на окремі завдання та їхній розподіл виконувалося web-розробником. Кожен модуль містив як взаємодію із сервером, так і створення інтерфейсу користувача.

Модуль, що розроблявся мною, відповідав за інтерфейс для користувачів та адміністратора.

База даних для проекту проектувалася та розроблялася штатним web-розробником.

Для надання таблицям зручного вигляду та реалізації деяких функцій пошуку, застосовувався плагін для бібліотеки jQuery DataTables 1.10.16.

У результати виконання завдання було отримано програмний компонент, який забезпечує функції моніторингу кількості водіїв та їх статусу.

### **Сторінка Login**

Сторінка Login являє собою сторінку входу до системи. На ній реалізовано 2 поля – Login та Pasword – за допомогою них здійснюється вхід в систему. Для валідації даних на стороні користувача застосовано плагін бібліотеки jQuery Validation Plugin v1.16.0.

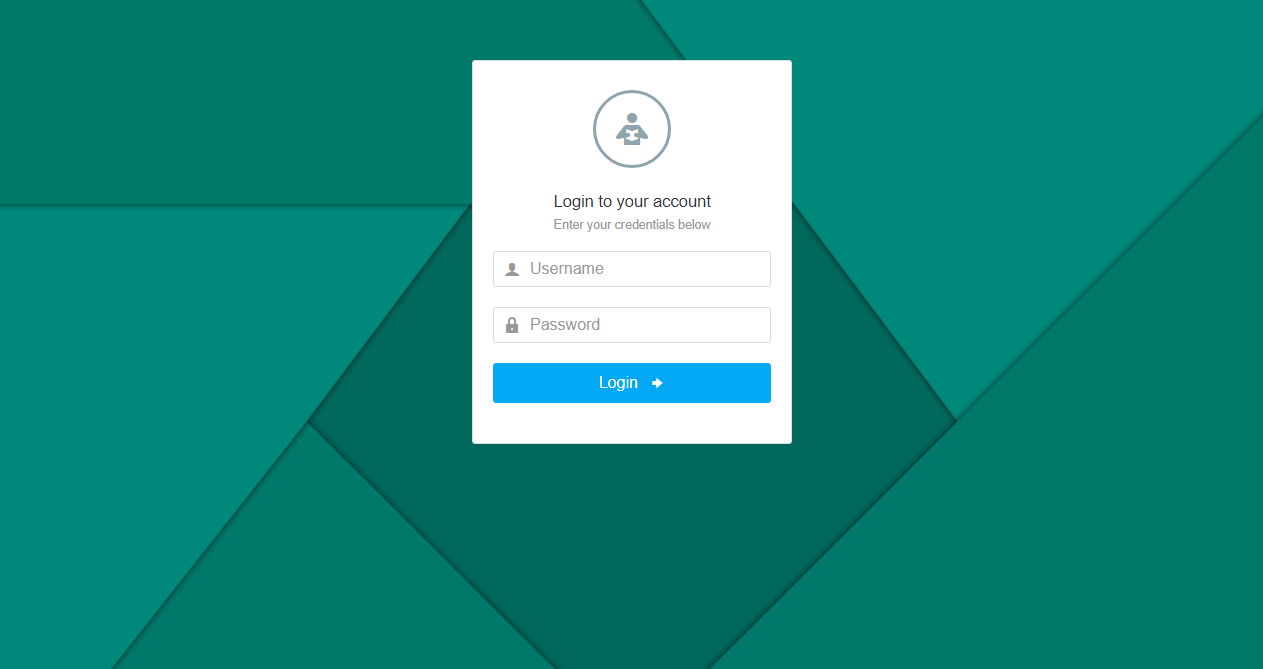


Рисунок 1 – Сторінка Login.

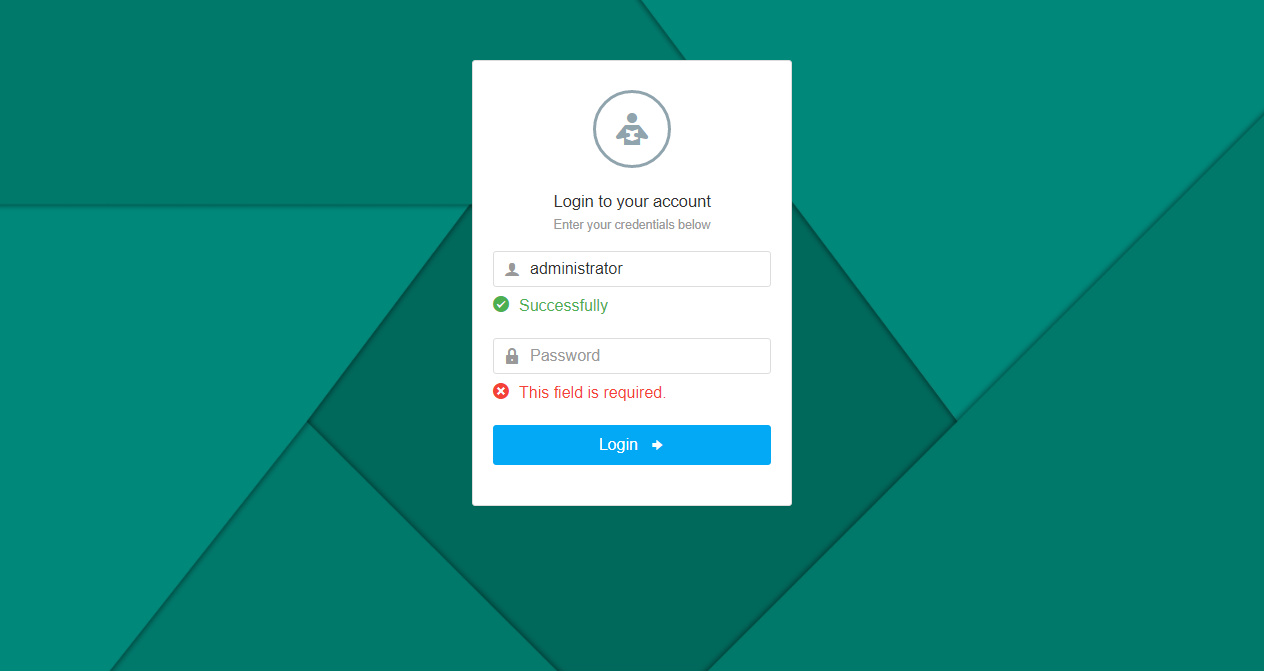


Рисунок 2 – Сторінка Login. Демонстрація валідації.

### **Сторінка** **Drivers**

Сторінка Drivers дозволяє адміністраторові вносити нових водіїв, видаляти їх та змінювати їх статус. На ній розташована таблиця, яка виводиться з допомогою плагіна для бібліотеки jQuery DataTables 1.10.16.

Також в наявності є 2 форми: одна для додавання нового водія, а друга для редагування вже наявних.

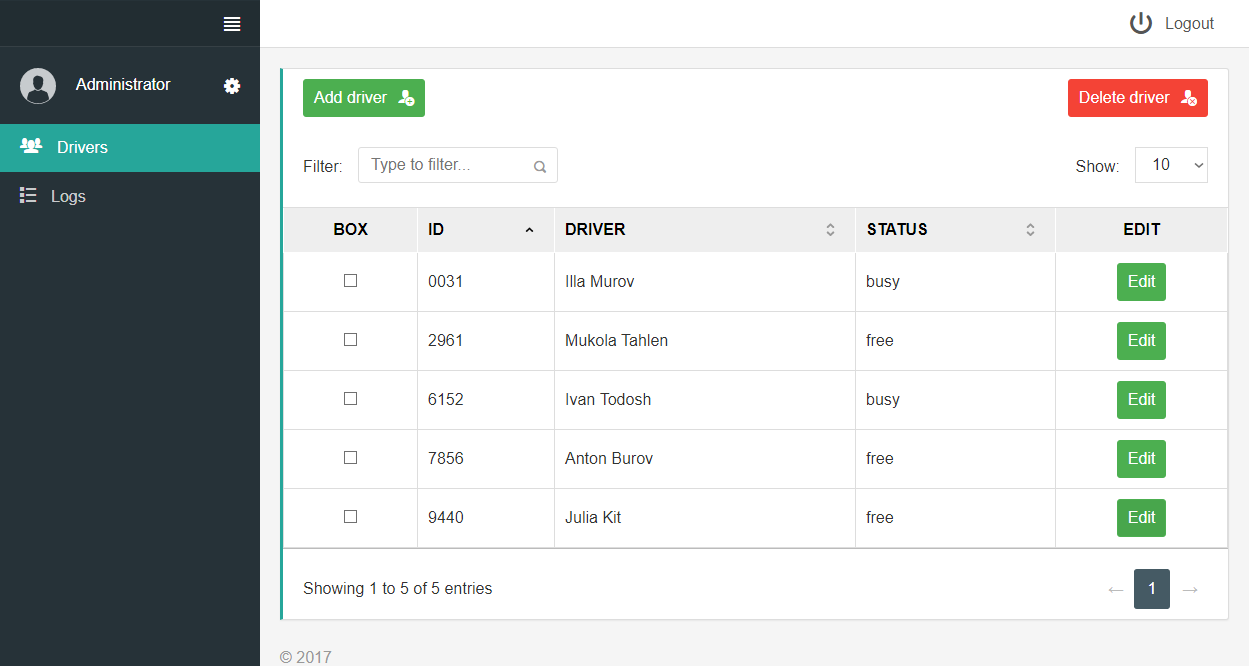


Рисунок 1 – Загальний вигляд сторінки Drivers.

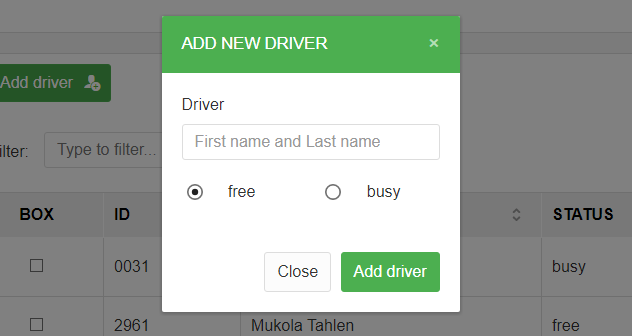


Рисунок 2 – Демонстрація модального вікна для додавання нового водія.

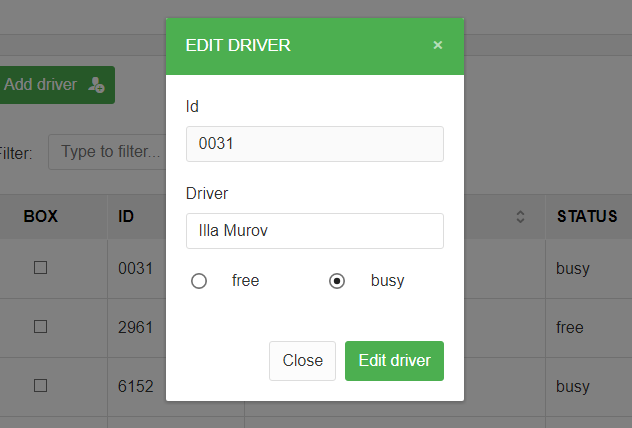


Рисунок 3 – Демонстрація вікна для редагування даних вже наявного водія.

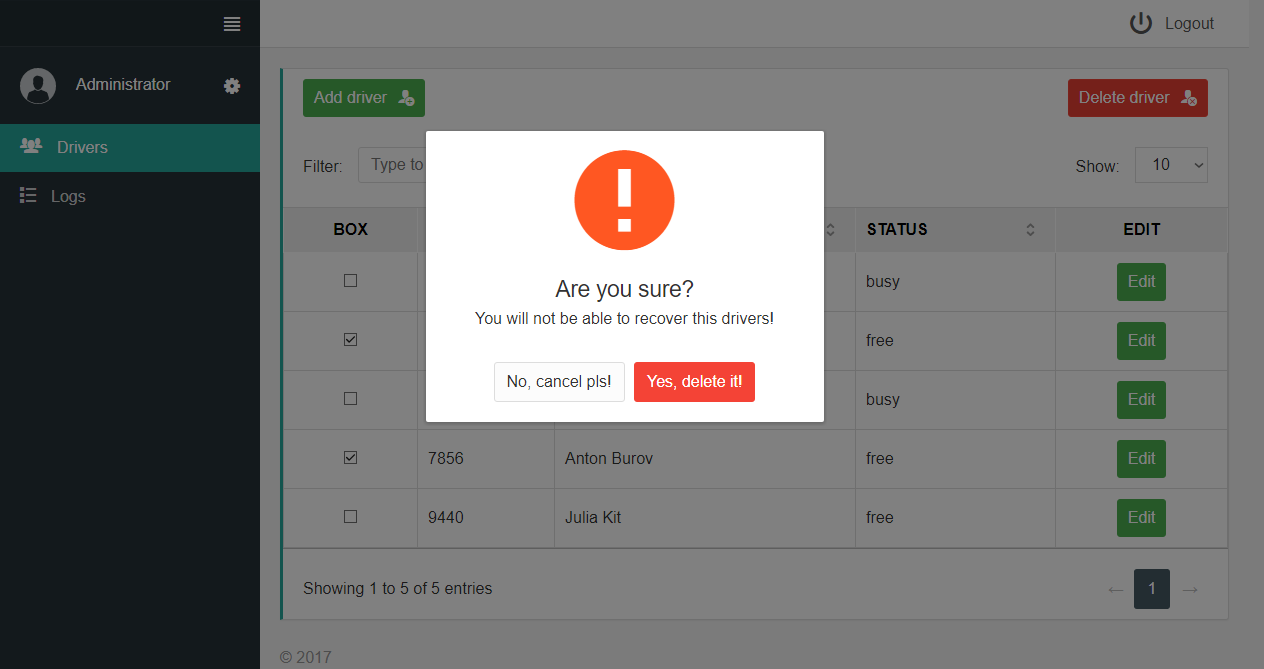


Рисунок 4 – Демонстрація вікна для видалення даних відмічених водіїв.

### **Сторінка Logs**

Сторінка Logs дозволяє переглянути логи дій, які виконувалися в системі. Це допоможе визначити, які дії могли спричинити ту чи іншу проблему, наприклад, втрату даних або поява невідомих об’єктів в системі. На ній розташована таблиця, яка виводиться з допомогою плагіна для бібліотеки jQuery DataTables 1.10.16.

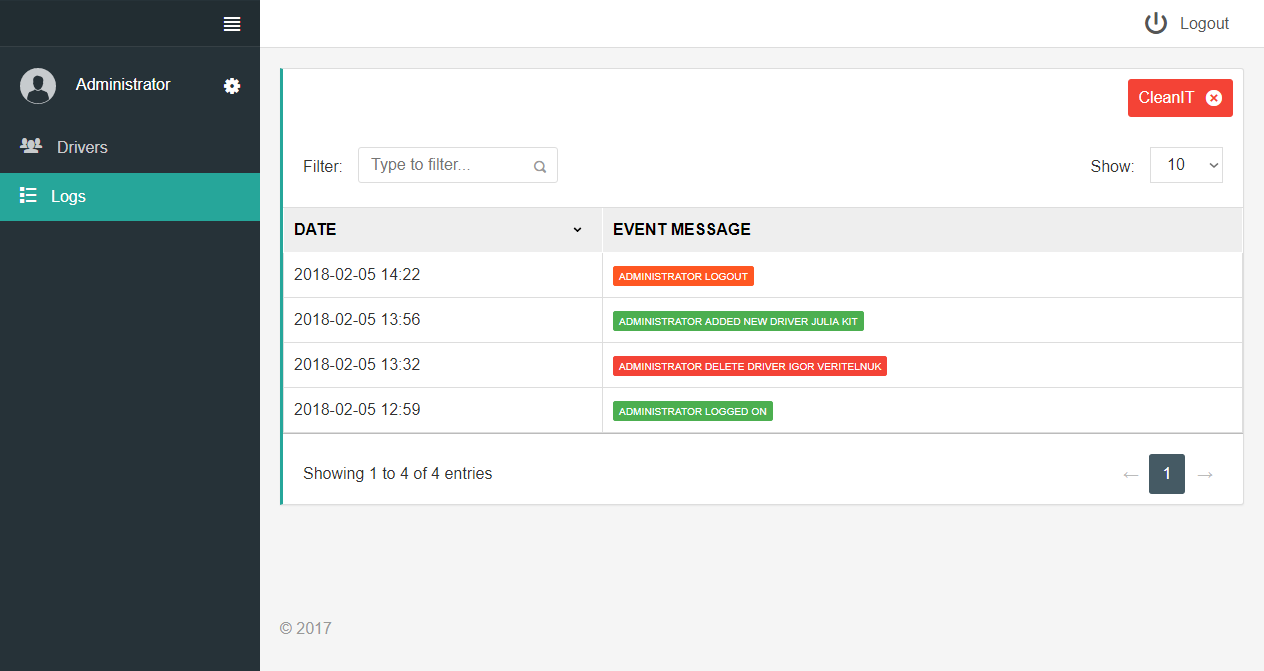


Рисунок 1 – Загальний вигляд сторінки Logs.

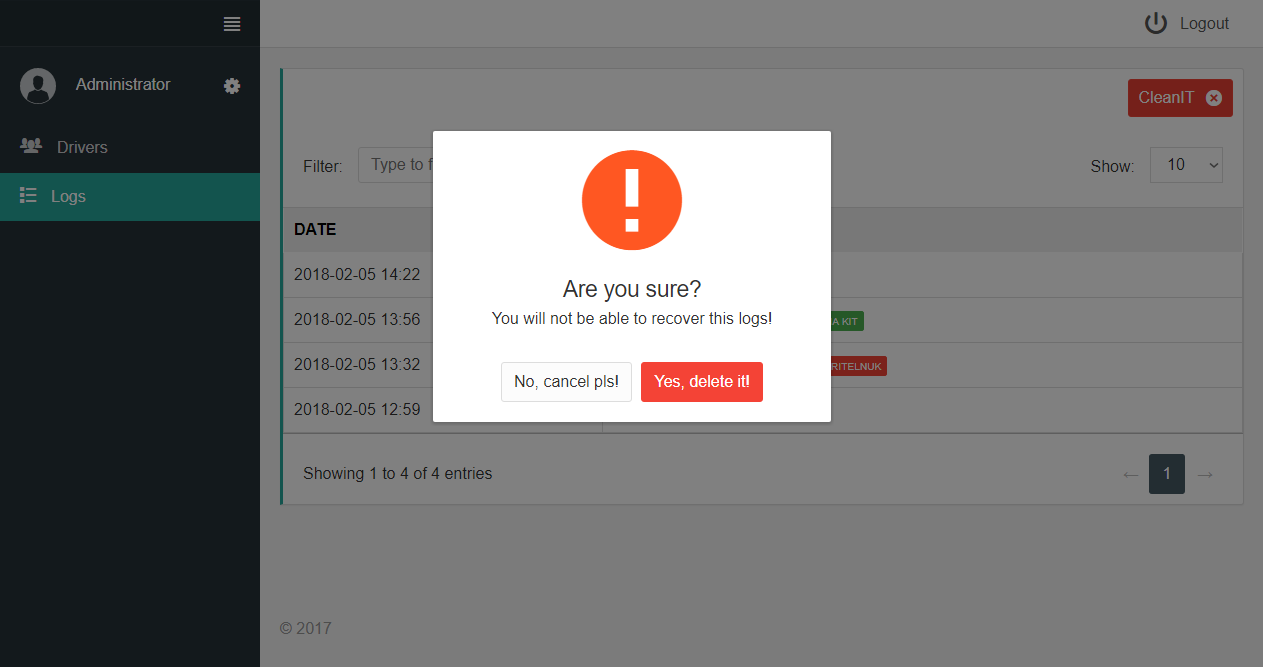


Рисунок 2 – Демонстрація вікна для очищення таблиці логів.

# **Індивідуальне завдання від університету**

## **Система контролю версій**

Функція Gіt, яка насправді робить її відмінною від практично будь-якої іншої SCM, є його розгалужена модель.

Git дозволяє і рекомендує вам мати кілька локальних гілок, які можуть бути цілком незалежними одна від одної. GIT дозволяє легко маніпулювати гілками: створення, об'єднання та видалення.

Це означає, що ви можете робити такі речі:

* Бути впевненим, що основний функціонал проекту ніколи не постраждає. Можна створити одну основну гілку на якій тримати лише вже готові та протестовані модулі проекту. Це дозволить розробляти інші модулі не переймаючись за втрату вже проробленої роботи.
* Нові гілки для кожної нової функції, з якою ви працюєте в даний момент. Це дозволить легко переключатися між ними, а потім видалити кожну гілку, коли ця функція об'єднується у основний проект.
* GIT дозволяє випробовувати багато різних ідей одночасно. Можна створити додаткову гілку, в якій поексперементувати над ідеями, а потім видалити її, якщо ідея не вдалася.

GitHub – чудовий сервіс на базі GIT, що дозволяє співпрацювати із багатьма учасниками команди одночасно. Однак він накладає певні обмеження. Особливо це помітно для невеликих студій, що не можуть дозволити собі платну підписку. А лише платна версія дозволяє користуватися приватними репозиторіями. До того ж чим більше таких репозиторіїв, тим більшою є ціна в місяць. Але публікувати код проекту у загальний доступ при розробці продукту також неприйнятна річ. В таких випадках багато хто із розробників

В таких випадках багато хто із розробників налаштовує власний GIT сервер. Це дозволяє не тільки заощадити кошти, але й більше контролювати свій сервер. Оскільки ніхто крім самого розробника не має до нього доступу. У більшості випадків більшість передових користувачів Linux вже мають власні сервери, оскільки налаштування GIT серверу не є складним завданням.

## **Налаштування власного GIT-сервера**

Оскільки GIT – це система, що в першу чергу, призвана полегшити контроль версій власного проекту. А це, в більшості випадків, передбачає одночасну роботу кількох осіб над одним проектом. Тому загальноприйнятою практикою є встановлення GIT на віддалений (remote) комп’ютер. Після цього відбувається налаштування локальних комп’ютерів (тобто особистих робочих станцій кожного із розробників).

GIT має бути встановлений на усіх комп'ютерах, які будуть взаємодіяти в системі контролю версій. GIT можна встановити із пакетів, оскільки він вже є доступних із репозиторії дистрибутиву. Для цього потрібно виконати наступну команду:

sudo apt-get install git-core

Після встановлення системи контролю версій, потрібно вказати користувачів:

sudo useradd git

passwd git

Для більшої зручності доступу до сервера, робочі станції розробників потрібно налаштувати на підключення до віддаленого сервера без пароля. Для цього потрібно створити SSH-ключ на робочій станції та передати його серверу.

ssh-keygen-t rsa

У ньому буде запропоновано вказати місце зберігання ключа, просто натисніть Enter, щоб використати розташування за замовчуванням. Друге питання полягає в тому, щоб вказати пароль, який буде необхідний для доступу до віддаленого сервера. Команда генерує два ключі - відкритий ключ і приватний ключ. Потрібно звернути увагу на розташування відкритого(публічного) ключа, оскільки він знадобиться при введені наступної команди.

Тепер потрібно скопіювати ці ключі на сервер, щоб дві машини могли спілкуватися один з одним. Для цього на робочій станції розробника потрібно виконати наступну команду:

cat ~/.ssh/id\_rsa.pub | ssh git@server "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized\_keys"

Тепер ssh на сервер і створити проектний каталог для Git. Ви можете використовувати потрібний шлях для репозиторію.

Тепер із допомогою SSH потрібно підключитися до віддаленого сервера і створити проектний каталог для GIT. Його можна створити у будь-якій зручній директорії:

git@server:~ $ mkdir -p /home/swapnil/project-1.git

Після того, як каталог був створений, потрібно перейти у нього і вже звідти працювати із системою контролю версій для даного проекту.

cd /home/swapnil/git/project

Тепер потрібно створити порожній репозиторій:

git init --bare

Ініціалізовано порожній репозиторій Git у /home/swapnil/project-1.git 

Тепер нам потрібно створити Git репозиторій на робочій станції програміста:

mkdir -p /home/swapnil/git/project

Щоб було зручніше працювати, перейдемо в створену директорію.

cd /home/swapnil/git/project

Тепер створіть файли, що потрібно буде додати у систему контролю версій. Після цього ініціалізуйте GIT:

git init  
Initialized empty Git repository in /home/swapnil/git/project

Тепер додавайте файли до репозиторія:

git add

Коли користувач вносить якісь зміни у файл чи додає нові файли, він повинен виконувати команду вище. Також до кожного коміта потрібно писати повідомлення. Це загальноприйнята практика і робиться щоб дати іншим розробникам зрозуміти, які саме зміни відбулися у гілці.

git commit -m "message" -a  
 [master (root-commit) 57331ee] message  
 2 files changed, 2 insertions(+)  
 create mode 100644 GoT.txt  
 create mode 100644 writing.txt

У цьому випадку у мене був файл GoT, і я зробив деякі зміни, тому, коли я запустив команду, вона зазначила, що в файл були внесені зміни. У наведеній вище команді опція "-a" означає, що цей опис використовується для всіх файлів у репозиторі. Якщо ви внесли зміни лише до одного файлу, ви можете вказати назву цього файлу, а не використовувати "-a". 

git commit -m "message" GoT.txt  
 [master e517b10] message  
 1 file changed, 1 insertion(+)

До цих пір ми працюємо на локальному сервері. Але щоб із цими файлами та їх змінами могли працювати інші учасники команди, потрібно занести їх на сервер:

git remote add origin ssh://git@remote-server/repo-<wbr< a="">>path-on-server..git

Інші учасники команди мають забрати зроблені кимось зміни із загального сервера:

git clone git@remote-server:/home/swapnil/project.git

Тепер щоб працювати із GIT сервером достатньо лише закомітити локально змінені файли та перенести цю інформацію на сервер:

git commit -m 'corrections in GoT.txt story' -a   
git push origin master

# **ВИСНОВОК**

Практика дозволила застосувати набуті в університеті знання та навички в умовах підприємства IT-галузі та здобути новий досвід. Під час проходження практики було виконане індивідуальне завдання. Отриманий програмний компонент, буде основою для написання глобальнішої системи.

Досвід роботи із MySQL та навички створення макетів сторінок і їх стилізації засобами HTML + CSS + JavaScript дуже знадобилися при вивченні розробці системи. В результаті був створений веб-додаток для моніторингу кількості водіїв та їх статусу.

Практика надала можливість попрацювати у сучасній компанії та отримати досвід співпраці із висококваліфікованими фахівцями, що поділилися досвідом та навчили розробляти web-додатки для реального використання на підприємстві.