

**Національний технічний університет України
„Київський політехнічний інститут імені Ігоря Сікорського”**

Кафедра автоматизації проектування енергетичних процесів і систем

КУРСОВА РОБОТА

з дисципліни: «Основи WEB-програмування – 1»

тема: «Ремонт побутової техніки»

Керівник Полягушко Л. Г.	Виконав Селіверстов Н. С.
Допущено до захисту	Студент 2 курсу
„___” _____ 2019р.	Групи ТВ-з 71
Захищено з оцінкою	залікова книжка №КВ-7214

2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

КУРСОВА РОБОТА

з дисципліни «Основи WEB-програмування – 1»
(назва дисципліни)

на тему: «Ремонт побутової технік »

Студента групи ТВ-371
спеціальність **121 «Інженерія програмного забезпечення»**
напряму підготовки **6.050103 Програмна інженерія**

Селіверстов Н. С.
(прізвище та ініціали)

Керівник Полягушко Л.Г.

звання, науковий ступінь, прізвище та ініціали)

(посада, вчене

Національна оцінка _____
Кількість балів: _____ Оцінка: ECTS _____

Члени комісії

_____	_____
(підпис)	(вчене звання, науковий ступінь, прізвище та ініціали)
_____	_____
(підпис)	(вчене звання, науковий ступінь, прізвище та ініціали)
_____	_____
(підпис)	(вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2019 рік

**Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет ТЕПЛОЕНЕРГЕТИЧНИЙ

(повна назва)

Кафедра автоматизації проектування енергетичних процесів та систем

(повна назва)

Напрямок підготовки 6.050103 Програмна інженерія

(шифр і назва)

**З А В Д А Н Н Я
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Селіверстову Никіті Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи – «Ремонт побутової техніки»

керівник курсової роботи – Полягушко Любов Григорівна

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи – «16» травня 2019 р.

3. Вихідні дані до проекту (роботи): мова JavaScript (JavaScript), середовище розробки Microsoft Visual Studio 2019 Community Preview

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) – розробити веб-додаток для абонементу бібліотеки, використовуючи інструментальні можливості мови JavaScript, зокрема створити функціональні можливості для відображення розміщеної в базі даних інформації, авторизації адміністратора, управління вакансіями (додавання, змінення, видалення).

5. Дата видачі завдання – 21 січня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання курсової роботи	Строк виконання етапів роботи	Примітка
1.	Налаштування середовища розробки		
2.	Проектування архітектурної побудови завдання		
3.	Підключення бази даних та створення відповідних «Колекцій» (MongoDB)		
4.	Проектування шаблону інтерфейсу користувача		
5.	Реалізація виведення інформації с БД, авторизації користувачів		
6.	Розробка та тестування зазначеного у змісту завдання функціоналу		
7.	Оформлення пояснювальної записки		

Студент

(підпис)

Селіверстов Н. С.

(прізвище та ініціали)

Керівник курсової роботи

(підпис)

Полягушко Л. Г.

(прізвище та ініціали)

АНОТАЦІЯ

На мові програмування ECMAScript (JavaScript), в редакторі Visual Studio Code створено обробник заказів для ремонту побутової техніки.

Завданням курсовій роботі з дисципліни «Основи Web-програмування» було створення веб-додатку для абонементу бібліотеки, на якому авторизовані користувачі додавати та дивитися власні заклази, які зберігаються в базі даних. Завданням було реалізувати функціонал елементарних дій над записами (створення / редагування / видалення заказів / огляд історії заказів) та обліковими записами користувачів (створення нового користувача, відображення імені користувача під створеними ним записами, можливість редагувати тільки свої заклази), внесення змін у базу даних та їх відображенню на сайті.

Робота міститься на HOMEP сторінках пояснювальної записки,

Додатково до роботи додано HOMEP рисунок, HOMEP додатка і HOMEP використаних літературних джерел.

ABSTRACT

In the programming language ECMAScript (JavaScript), in the editor of Visual Studio Code created a handler of orders for repair of home appliances.

The objective of the course work on "Fundamentals of Web-programming" was to create a Web application for a library subscription, where authorized users add and see their own orders stored in the database. The task was to implement the function of elementary actions over the records (creation / editing / deleting orders / reviewing the history of orders) and user accounts (creating a new user, displaying a username under the records he created, the ability to edit only their orders), making changes to the database, and their display on the site. The work is on the 42 pages of the explanatory note,

In addition to the work added HOMEP drawings, HOMEP application and HOMEP used literary sources.

ЗМІСТ

Вступ.....	6
1.Загальні відомості про web-додаток.....	8
1.1.Опис вибраних засобів розробки.....	8
1.1.1.Обґрунтування вибору мови програмування та фреймворку.....	8
1.1.2.Обґрунтування вибору середовища розробки.....	10
1.1.3.Обґрунтування вибору СКБД.....	11
1.2.Технічне завдання	12
2.Розробка програмного забезпечення.....	14
2.1.Розробка веб-сайту.....	14
2.2 Інтерфейси веб-додатку.....	14
2.3.Колекції бази даних mongo db	18
2.4.Компоненти програми	20
2.5. Розбір файлу app.js.....	21
2.6 Запуск веб-додатку.....	23
Висновок.....	24
Список літературних джерел	25
Додаток 1.....	28
Додаток 3.....	36

ВСТУП

Об'єктивною реальністю сьогодення є широке впровадження у сфери життєдіяльності особи, суспільства та держави сучасних інформаційних технологій, розгортання на їх основі локальних і глобальних інформаційних систем, призначених для прискорення обміну інформацією та доступу до різноманітних інформаційних ресурсів.

Інтернет є надзвичайно привабливим засобом комунікації, що може пояснити його безпрецедентний розвиток. Новітні засоби зв'язку дозволили об'єднати розрізнені комунікаційні системи у глобальну мережу. Завдяки цьому людина отримала можливість обмінюватися інформацією в межах всієї планети, не залежно від кордонів і відстаней. Саме Інтернет є однією з ознак переходу суспільства від індустріальної стадії розвитку до інформаційної.

Стрімкий розвиток мережевих інформаційних технологій, окрім помітного зниження тимчасових і просторових бар'єрів в розповсюдженні інформації, відкрив нові перспективи. Можна з упевненістю стверджувати, що в сучасному світі має місце тенденція злиття різноманітних інструментів і інформаційних технологій, відбувається формування на цій основі принципово нових інтегрованих технологій в сфері керування кадрами, заснованих, зокрема, на Інтернет-технологіях.

Інформаційні технології — процес, що використовує сукупність засобів і методів збору, обробки і передачі даних, для отримання інформації нової якості про стан об'єкту, процесу або явища.

Інформаційна технологія — це комплекс взаємозв'язаних, наукових, технологічних, інженерних дисциплін, що вивчають методи ефективної організації праці людей, зайнятих обробкою і зберіганням інформації; обчислювальну техніку і методи організації і взаємодії з людьми і виробничим устаткуванням, їх практичні додатки, а також пов'язані зі всім цим соціальні, економічні і культурні проблеми.

Зручність і гнучкість гіпертекстового представлення матеріалу, оперативний доступ до інформації, розташованої в різних регіонах і країнах, висока оперативність

оновлення і інші переваги інтернет-технологій дозволили достатньо швидко упровадити їх в практику багатьох навчальних закладах.

Web-додаток для обробки заказів дозволяє дистанційно приймати заклази по ремонту побутової техніки, відображати їх статус (як і для користувачів так і для робітників, які виступають виконавцями), що мінімізує матеріальні потреби у менеджменті сервіса.

Таким чином, даний додаток буде корисним і актуальним як для сервісу ремонту побутової техніки, так і для будь якого сервісу, який хоче пропонувати свої послуги дистанційно.

1. ЗАГАЛЬНИЙ АНАЛІЗ ТЕХНОЛОГІЙ ВЕБ-ПРОГРАМУВАННЯ

1.1. Опис вибраних засобів розробки

1.1.1 Обґрунтування вибору мови програмування та фреймворку

JavaScript — скриптова мова, призначена для створення інтерактивних веб-сторінок, проте на цьому сфера її застосування не закінчується. JavaScript активно використовують при розробці серверної частини веб-додатку чи навіть ігор.

У JavaScript є свій стандарт: ECMAScript, специфікація якого знаходиться на сайті [<https://www.ecma-international.org/ecma-262/9.0/index.html>] в розділі «стандарт мови».

За допомогою JavaScript можна виконувати наступні дії:

- змінювати сторінку, писати на ній текст, додавати і видаляти теги, міняти стилі елементів;
- реагувати на події: скрипт може чекати, коли щось станеться (клік, закінчення завантаження сторінки) і реагувати на це виконанням функції;
- виконувати запити до сервера і завантажувати дані без перезавантаження сторінки;
- встановлювати і зчитувати cookie, валідувати дані, виводити повідомлення і багато іншого.

До сильної сторони даної мови належить повна підтримка браузерами. Наприклад, такі технології як ActiveX, VBScript, XUL — підтримуються не в кожному браузері. Такі технології як Flash, Silverlight, Java — не повністю інтегровані з браузером, працюють в своєму оточенні.

Дана мова програмування не позбавлена недоліків, таких, як, наприклад, нестрога типізація, що є причиною неявного переведення типів. Однак усі недоліки перекриваються поширеністю та кросбраузерністю мови.

Для створення додатку використовувалась Node.js — платформа з відкритим кодом для виконання високопродуктивних мережових застосунків, написаних мовою JavaScript. Якщо раніше Javascript застосовувався для обробки даних в браузері на сторонні користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

- асинхронна однопотокова модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- рушій JavaScript Google V8.

Для більш продуктивного і простого процесу розробки було обрано web-фреймворк Express, в якому використовується мова JavaScript як на клієнті, так і на сервері, а також в роботі з базою даних, що дозволяє виконувати один і той самий код на фронт-енді та бек-енді одночасно.

Ядро Express складається з дюжини пакетів, які використовуються в більшості додатків, наприклад, пакет, який обробляє вхідні HTTP-підключення, і який дозволяє писати HTML-шаблони, автоматично оновлюванні при зміні даних. Також існують додаткові пакети, які надають додаткового функціоналу для вашої програми. І крім цих "офіційних" пакетів, присутні сотні інших, написаних співтовариством, які розміщені на ресурсах як GitHub та інші.

Утиліта npm — це менеджер пакунків для мови програмування JavaScript. Для середовища виконання Node.js є менеджером пакунків за замовчуванням. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через веб-сайт npm. Npm може управляти пакунками, які є локальними залежностями певного проекту, а також глобально інстальованими інструментами JavaScript. При

використанні npm як менеджера залежності для локального проекту, можна встановити одною командою всі залежності проекту через файл `package.json`.

У файлі `package.json` кожна залежність може визначати діапазон дійсних версій, використовуючи схему семантичної версії, що дозволяє розробникам автоматично оновлювати свої пакети, одночасно уникаючи небажаних змін. Одним з пакетів npm, який збирає воедино всі вихідні коди і статичні файли додатку є Gulp. Gulp - task-менеджер для автоматичного виконання завдань (наприклад, мініфікації, тестування, об'єднання файлів), написаний на мові програмування JavaScript. Програмне забезпечення використовує командний рядок для запуску завдань, визначених у файлі `Gulpfile`. У режимі розробки також буде корисним додаток Nodemon, який відслідковує зміну визначених файлів та перезапускає сервер розробки "на льоту", тобто кожен раз, коли ви вносите будь-які зміни у файл, ви відразу ж бачите це в браузері. Утиліта npm дуже проста у використанні і до того ж легко розширюється.

1.1.2 Обґрунтування вибору середовища розробки

Під час розробки web-додатку використовувався редактор Visual Studio 2019 Community Preview.

Visual Studio 2019 Community Preview – редактор для створення, редагування та налагодження сучасних веб-додатків і програм для хмарних систем. Visual Studio 2019 Community Preview розповсюджується безкоштовно, доступний у версіях для платформ Windows, Linux і OS X (кросплатформенний). Редактор містить вбудований дебагер, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки, підтримує плагіни, розроблені спільнотою. Велика кількість плагінів, які в рази покращують продуктивність і швидкість роботи програмістів, зручний і зрозумілий в користуванні, і позиціонується як легкове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки.

На мою думку, нині це один з найпотужніших текстових редакторів з повною свободою налаштування та автоматизації й великою кількістю створених завдяки цьому розширень та надбудов.

1.1.3 Обґрунтування вибору СКБД

Напевне найголовнішим і найвідповідальнішим етапом під час розробки веб-додатку є проектування і розробка бази даних.

Основою бази даних є модель даних — фіксована система понять і правил для представлення даних структури, стану і динаміки проблемної області в базі даних. У різний час послідовне застосування одержували ієрархічна, мережна і реляційна моделі даних. У наш час усе більшого поширення набуває об'єктно-орієнтований підхід до організації баз даних.

Система керування базами даних — це сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують керування створенням і використанням баз даних.

Реляційна модель орієнтована на представлення даних у вигляді двовимірних таблиць. Таке уявлення зручно для користувачів. Але реляційні бази даних не можуть справлятися з навантаженнями актуальними в наш час.

Три основних проблеми РСКБД:

- горизонтальна масштабування при великих обсягах даних;
- продуктивність кожного окремого сервера;
- негнучкий дизайн логічної структури.

Все це змушує розробників використовувати альтернативи реляційних баз даних, які відносяться до сімейства NoSQL.

Термін NoSQL був придуманий Еріком Евансом. NoSQL в інформатиці — термін, що позначає ряд підходів, проектів, спрямованих на реалізацію моделей баз даних, що мають суттєві відмінності від використовуваних реляційних СКБД з доступом до даних засобами мови SQL.

NoSQL — важливий і корисний інструмент, але він не може вважатися універсальним. Основна мета підходу — розширити можливості БД там, де SQL недостатньо гнучкий, і не витіснити його там, де він справляється зі своїми завданнями. В основі ідеї NoSQL лежать:

- нереляційних модель даних;
- відкритий вихідний код;
- хороша горизонтальна масштабованість.

Характерним представником СКБД сімейства NoSQL є MongoDB— документно-орієнтована СКБД. Вона офіційно підтримується платформой Node.js. Саме цю СКБД використовує веб-додаток кадрового агентства.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінга (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера в кластер проводиться без зупинки роботи БД простим додаванням нових машин.

1.2 Технічне завдання

1.2.1 Загальні відомості

Розробка веб-додатку для створення обробника заказів ремонтного сервісу побутової техніки. Призначення розробки — є засобом для створення, редагування та перегляду заказів користувачів для подальшого їх виконання/відхилення

працівниками. Веб-додаток обробник заказів створюється з метою автоматизації роботи з користувачами в мережі Інтернет, надання відповідних можливостей відповідно до типу користувача.

Інформаційна система повинна бути проста для користувача без додаткових навиків.

Щоб користуватися сервісом необхідно мати комп'ютер з доступом до інтернету та будь-який сучасний браузер.

2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розробка веб-додатку

Розробка веб-додатку проходила в декілька етапів. В наступних розділах буде описаний кожен з цих етапів.

Провівши аналіз предметної області, було зроблено висновок, що база даних повинна містити: користувачів їх заклади на інформацію про них. Сильною стороною документно-орієнтованої СКБД MongoDB є її без-структурність, а це означає, що структурна схема документів може видозмінюватись з часом.

Отже, інформаційна система повинна містити повну інформацію про користувачів, опубліковані заклади, та інші дані, необхідні для виведення інформації та зручну навігацію.

Попередній аналіз предметної області свідчить про необхідність реалізації наступних завдань:

- авторизація користувачів;
- зручне середовище для створення закладів користувачів;
- збереження та зміна статусу закладів;
- можливість редагування змісту створених закладів;
- можливість повторювати свої старі попередні заклади.

Після підсумування вимог можна зробити висновок, що веб-додаток буде мати зручний інтерфейс для користувача, доступний із будь-якого комп'ютера або мобільного пристрою із доступом до інтернету.

2.2 Інтерфейси веб-додатку

Важливим етапом розробки веб-додатку є створення інформативної сторінки з відображенням закладів. Приклад інтерфейсу користувача (рисунки 2.1). На головній сторінці веб-додатку розміщені всі заклади. Це важливо для зручного користування додатком і перевірки найактуальнішої інформації.

Тільки зареєстрований користувач має можливість користуватися веб-додатком. Це зумовлено тим, що обробник має чітке розподілення користувачів:

Адміністратори, Працівники та Замовники. (усі мають різні повноваження щодо перегляду та редагування інформації)

Кнопка «Make Order» відкриває сторінку створення нового заказу. Кнопка «Exit» закінчує сесію користувача, повертає его на сторінку авторизації. Кнопка «Change status» змінює статус заказу на «Done», якщо до цього він був «Active» та навпаки. Реалізована дана система для оновлення минулих заказів, та безпосередньо для взаємодії з користувачами-працівниками. Кнопка «Edit» відкриває сторінку з подібними полями на сторінку створення замовлення (рисунок 2.4) і надає можливість змінювати данні полів «Title» та «Text».

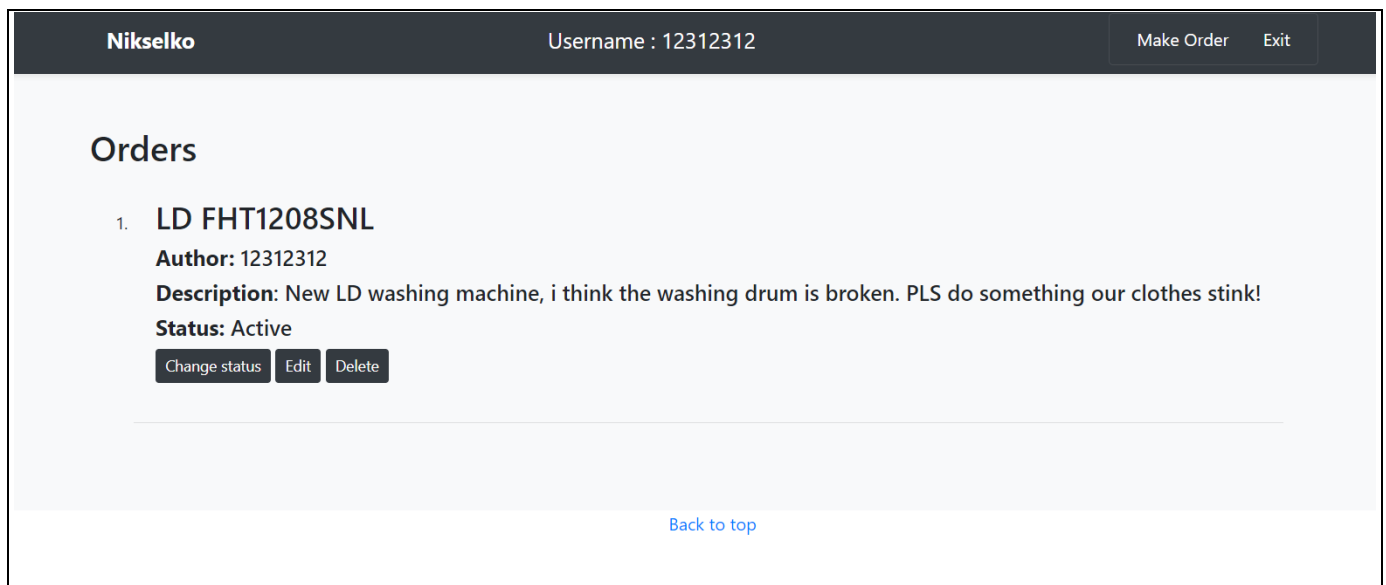
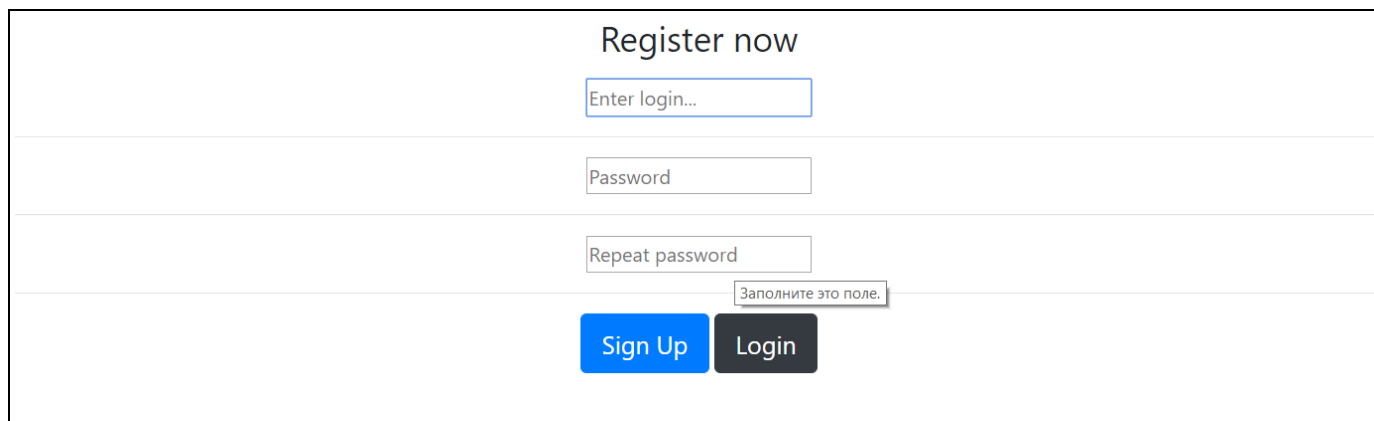


Рисунок 2.1 – Сторінка веб-додатку з тестовим заказом

Для реєстрації нового користувача передбачена сторінка реєстрації (рисунок 2.2). Потрібно заповнили поле «Логін». Після введення вибраного логіну, здійснюється перевірка на наявність подібного логіну в базі даних вже зареєстрованих користувачів. Також, за відсутності подібності, новий користувач має можливість вибрати будь-який логін, надруковане латинськими літерами та цифрами, але не коротше 3 та не більше 16 символів. Користувач має придумати свій власний пароль у полі «Пароль». Вимоги до нього: не коротше 8 символів, а також підтвердити

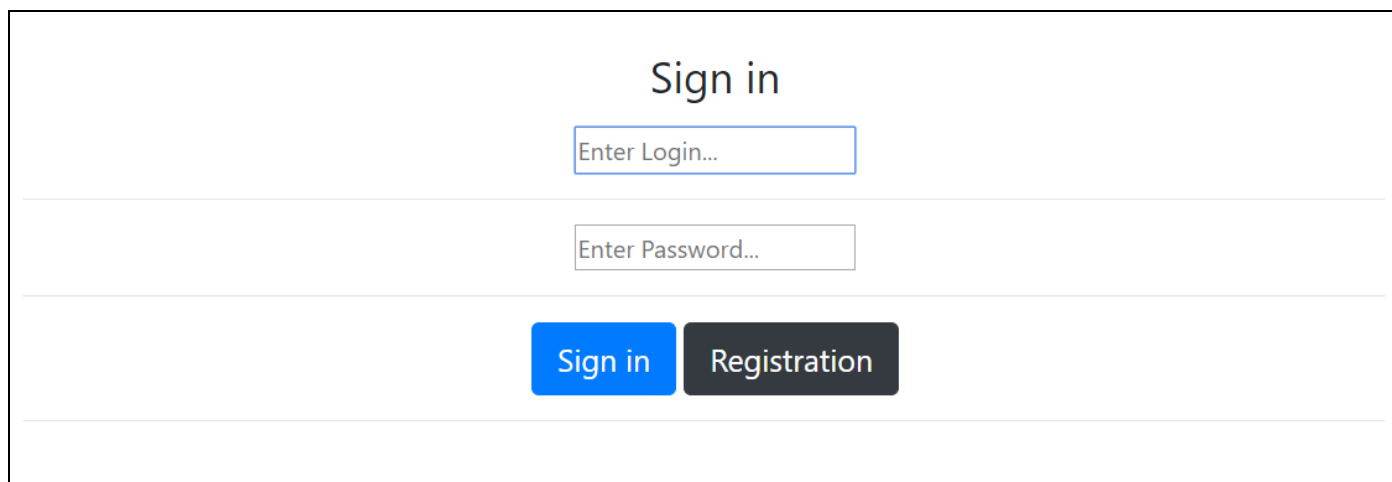
коректність вибраного паролю повторно, надрукувавши его у відповідному полі «Повторити пароль»



The registration form is titled "Register now". It contains three input fields: "Enter login...", "Password", and "Repeat password". Below the "Repeat password" field, there is a small tooltip that says "Заполните это поле." (Fill in this field.). At the bottom of the form, there are two buttons: a blue "Sign Up" button and a dark grey "Login" button.

Рисунок 2.2 – Сторінка реєстрації у веб-додатку

Авторизація користувача (рисунок 2.3) здійснюється виключно за наявності попередньої успішної реєстрації нового користувача, інформація про нового користувача була успішно додана до бази даних. Для коректної авторизації потрібно ввести в відповідні поля власний логін та пароль. Перевірка реалізована відповідності логіну та паролю, в разі будь-якої розбіжності видається помилка, користувач може спробувати ще раз ввести коректні дані для входу.

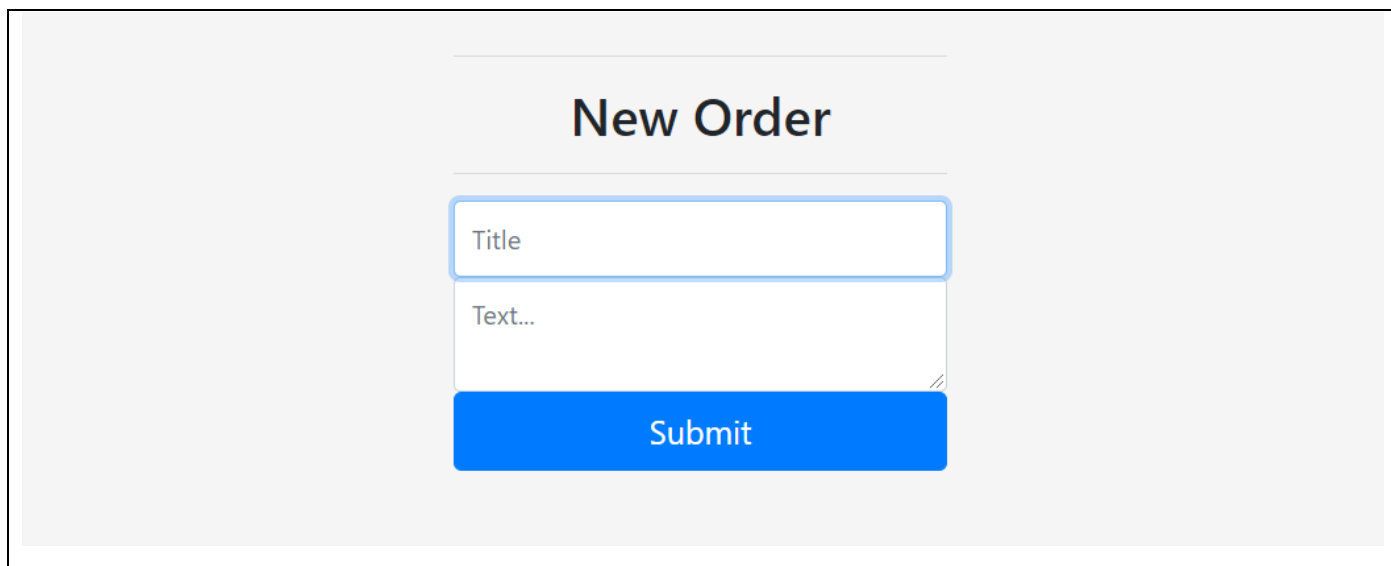


The sign in form is titled "Sign in". It contains two input fields: "Enter Login..." and "Enter Password...". Below the "Enter Password..." field, there is a small tooltip that says "Заполните это поле." (Fill in this field.). At the bottom of the form, there are two buttons: a blue "Sign in" button and a dark grey "Registration" button.

Рисунок 2.3 – Сторінка авторизації у веб-додатку

Як було зазначено вище в частині про головну сторінку веб-додатку, тільки авторизований користувач-клієнт може створювати або редагувати заказ. Важливо, що користувач-клієнт може змінювати тільки власні заказы. Користувачі працівники бачать усі заказы від усіх користувачів-клієнтів як і адміністратори, але змінювати

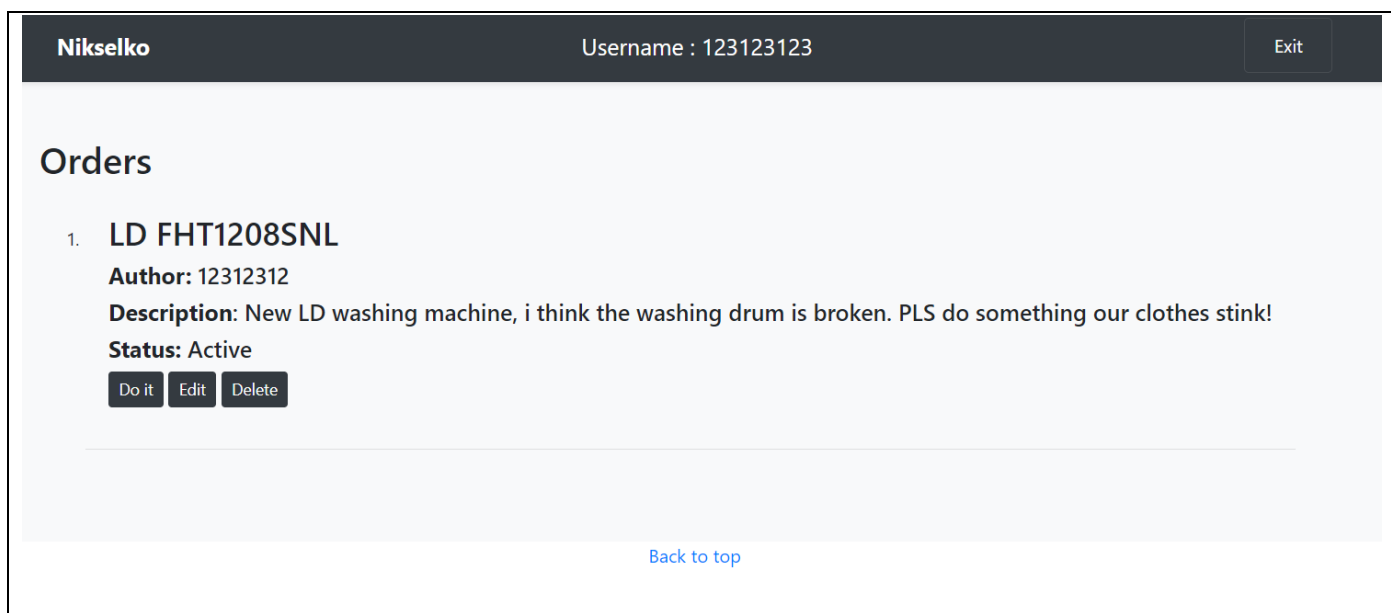
статус або склад заказів можуть тільки працівники. Приклад інтерфейсу редагування посту веб-додатку (рисунок 2.4).



The screenshot shows a web form titled "New Order". It contains a text input field labeled "Title", a larger text area labeled "Text...", and a prominent blue button labeled "Submit".

Рисунок. 2.4 – Сторінка створення нового заказу

На рисунку 2.5 зображена домашня сторінка для користувача-працівника. Кнопка «Do it» змінює статус заказу на «Done», після цього статус заказу може змінити тільки користувач-клієнт замовник саме цього заказу. Кнопка «Edit» відкриває сторінку з подібними полями на сторінку створення замовлення (рисунок 2.4) і надає можливість змінювати данні полів «Title» та «Text».



The screenshot displays a user interface for a worker. The top header includes the logo "Nikselko", the username "Username : 123123123", and an "Exit" button. Below the header, the section "Orders" lists a single order: "1. LD FHT1208SNL". The order details include "Author: 12312312", "Description: New LD washing machine, i think the washing drum is broken. PLS do something our clothes stink!", and "Status: Active". At the bottom of the order details are three buttons: "Do it", "Edit", and "Delete". A "Back to top" link is located at the bottom of the page.

Рисунок 2.5 – Домашня сторінка для користувача-працівник

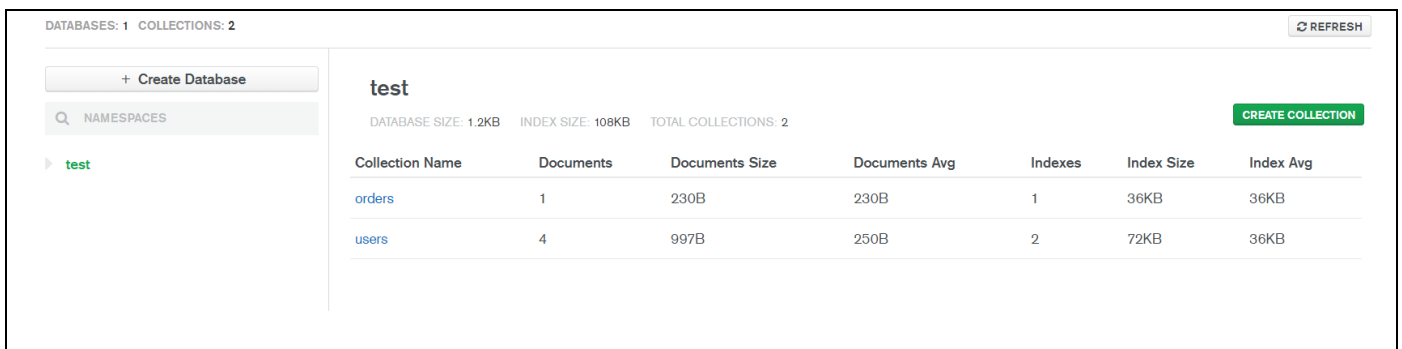
2.3 Колекції бази даних MongoDB

В базі даних MongoDB були створені колекції (рисунок 3.1) даних такі як:

1. Posts – колекція постів з книгами, в даному випадку наразі в базі розміщені 24 книги відображаються к сторінках веб-додатку, де зберігаються унікальні дані для ідентифікації постів в веб-додатку, так як: унікальний ідентифікатор, творець посту, дата створення, дата зміни та інші;

2. Sessions – колекція сесії користувачів веб-додатку, де зберігаються ідентифікатори сесії. Дані зберігаються в базі даних за замовчування, це найбільш безпечно, ніж зберігати їх в браузері користувача. Сесії використовуються для відновлення «стану» між користувачем і веб-додатком, а також дозволяє провести ідентифікацію користувача за допомогою зазначених вище ідентифікаторів. Дозволяє відновити роботу з веб-додатком одразу, виключаючи необхідність входити в додаток щоразу;

3. Users – колекція користувачів веб-додатку, де зберігаються унікальні дані для входу в веб-додаток, так як: логін, пароль, ідентифікатор та інші.



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'DATABASES: 1' and 'COLLECTIONS: 2'. A '+ Create Database' button is at the top. Below it, a search bar labeled 'NAMESPACES' contains the text 'test'. The main area displays the 'test' database details: 'DATABASE SIZE: 1.2KB', 'INDEX SIZE: 108KB', and 'TOTAL COLLECTIONS: 2'. A 'CREATE COLLECTION' button is in the top right. Below this, a table lists the collections:

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
orders	1	230B	230B	1	36KB	36KB
users	4	997B	250B	2	72KB	36KB

Рисунок. 3.1 – Колекції бази даних MongoDB

На рисунку 3.2 відображена детальна інформація про закази. Є унікальний ідентифікатор (для коректного пошуку заказу), заголовок, данні заказу, творець – ідентифікатор унікальний користувача, дата та час створення, зміни, а також посилання.

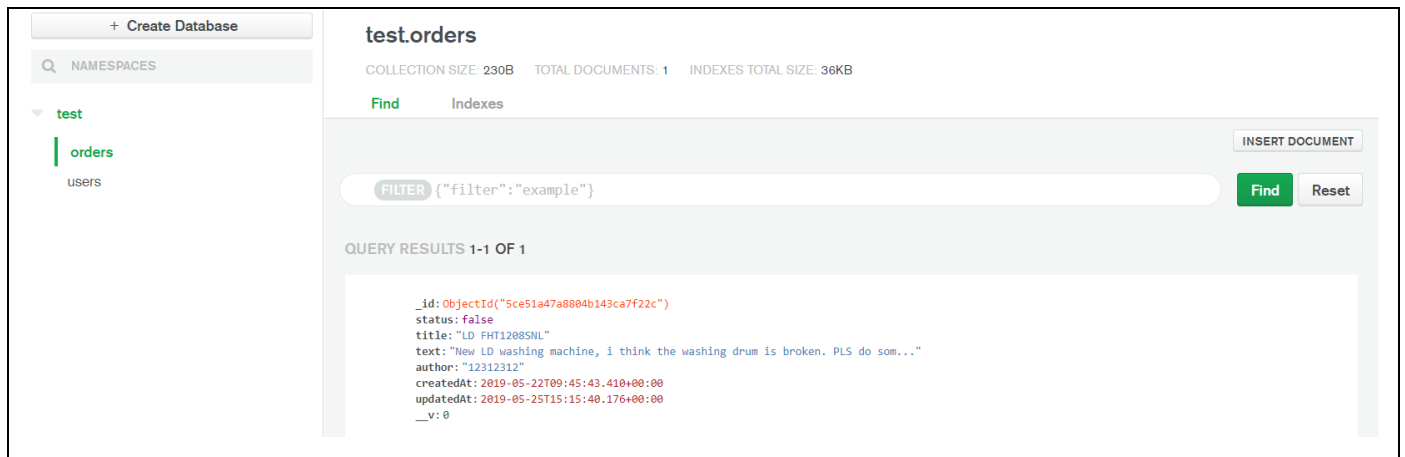


Рисунок. 3.2 – Інформація про закази баз даних MongoDB

На рисунку 3.3 відображена інформація про користувачів: ідентифікатор, логін, зашифрований пароль та дата і час створення та зміни даних користувача.

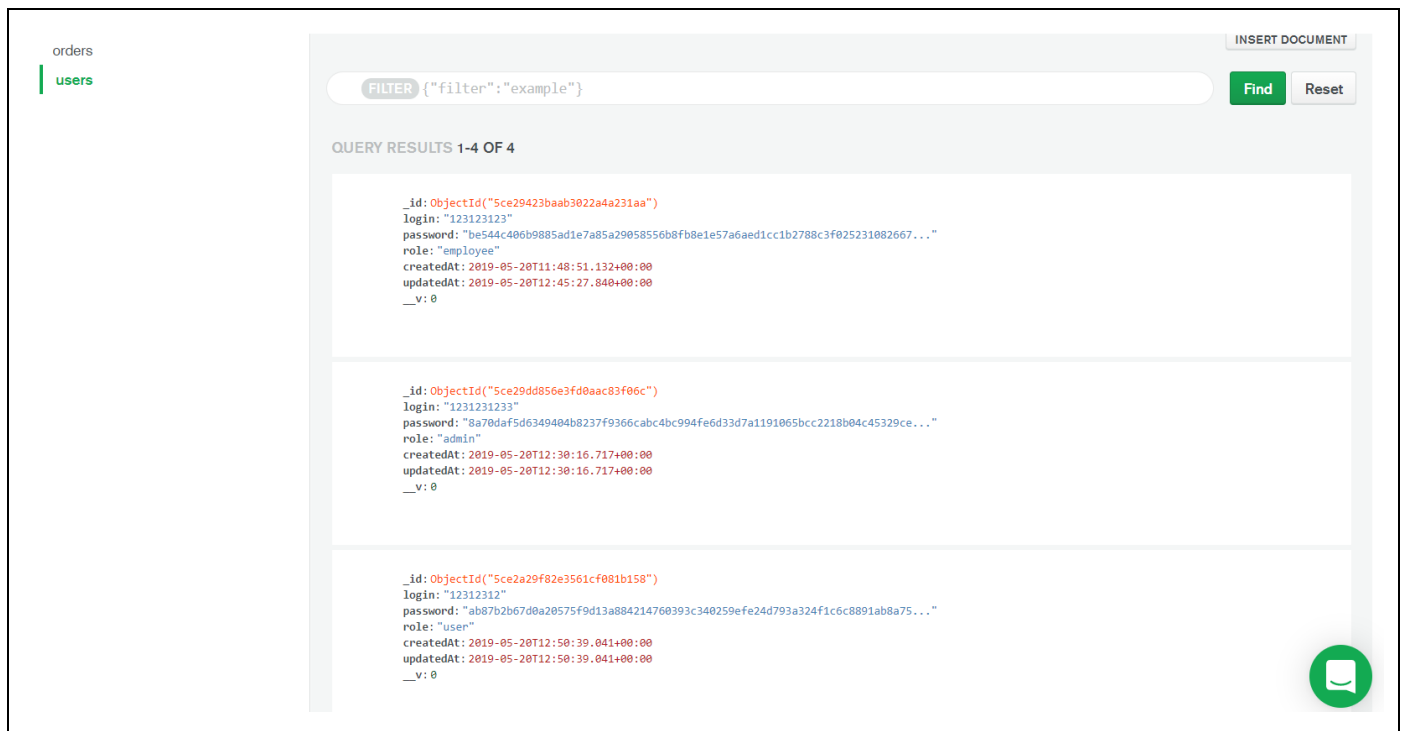


Рисунок. 3.3 – Інформація про користувачів колекції Users бази даних MongoDB.

2.4 Компоненти веб-додатку

До компонентів програми відносяться модулі (рисунок 4.1), які можна умовно розділити на логічні частини:

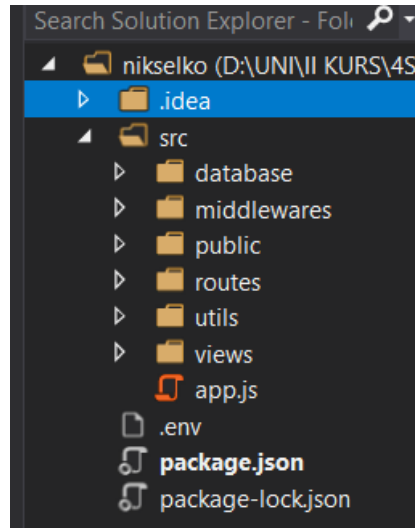


Рисунок 4.1 – Логічна структура побудови веб-додатку

модуль бази даних Models – для зчитування та запису у бази даних;

модуль шаблонів відображення Views – для форматування відображення сторінок веб-додатку;

модуль публічних даних веб-додатку Public — для зберігання мініфікованих стилів CSS та скриптів JS, зображень та всього іншого, що буде розміщено на реальному сервері, яким будуть користуватися користувачі;

модуль шаблонів шляхів Routes – для визначення напрямків потоку інформації в веб-додатку

файли ENV та ENV.Example – файли, в яких зберігаються змінні оточення

головний модуль app.js;

файл конфігурації config.js;

файл package.json, який зберігає список пакетів, необхідних для проекту з потрібними версіями;

2.5 Розбір файлу app.js

Підключаємо фреймворк Express за допомогою `const express = require('express')`.

Здавалося, навіщо встановлювати додатковий фреймворк, якщо можна скористатися готовим модулем `http`, який є в Node.js API. Express сам використовує модуль `http`, але разом з тим надає ряд готових абстракцій, які спрощують створення сервера і серверної логіки, зокрема, обробка відправлених форм, робота з куками і т.д.

Команда, записана у рядки кода: `const app = express()` створює об'єкт додатку, команда `const port = 3000` вказує номер порта, на якому запускається сервер.

Підключення решти необхідних модулів у поточний модуль:

```
const bodyParser = require('body-parser') – для обробки POST-запитів;  
const cookieParser = require('cookie-parser') – для читання cookie необхідно  
використовувати підключений тут модуль 'cookie-parser';  
const session = require('express-session') підключення сеансового модуля 'express-  
session, який зберігає дані про сеанс на сервері;  
const db = require('./database') – підключення модуля 'db' для роботи з MongoDB.
```

Підключення користувацьких модулів подано наступною частиною коду:

```
const { Catalog, User } = require('./models')  
const express = require('express')  
const path = require('path')  
const morgan = require('morgan')  
const config = require('./config')  
const busboyBodyParser = require('busboy-body-parser')
```

Підключення до бази даних MongoDB за допомогою `database.js`

Налаштування WEB – додатка проекту:

Встановлюємо який шаблонізатор будемо використовувати (механізм візуалізації). Після встановлення механізму візуалізації `views` не потрібно вказувати його або завантажувати модуль шаблонізатора в додаток; Express завантажує модуль внутрішніми засобами;

```
app.set('views', 'src/templates')
```

Визначаємо шлях, де знаходяться наші шаблони.

```
app.use(express.static('static'))
```

Вбудована функція проміжного програмного забезпечення в Express, яка визначає розташування статичних файлів (CSS, JS, Images).

```
app.use(session({ secret: 'cats' }))
```

Ключ для шифрування cookie файлів.

```
app.use(cookieParser())
```

Використовуємо cookieParser для зчитування файлів cookie.

```
app.use(bodyParser.urlencoded({ extended: true }))
```

Використовуємо фреймворк body-parser. Цим робимо доступний об'єкт req.body, а також в ньому поля форми.

```
app.use(bodyParser.json())
```

Для коректного зчитування POST-запитів.

```
app.use(passport.initialize())
```

Ініціалізація механізму аутентифікації і сесій.

```
app.use(passport.session())
```

Далі створена функція, яка робить запит щодо логіну користувача:

```
app.get('/login', (req, res) => {  
  if (req.cookies.failureMessage) {  
    res.locals.messages = [req.cookies.failureMessage];  
    res.clearCookie('failureMessage');  
  }  
  res.render('sign-in');  
})
```

Отримані дані попереднього запиту проходять перевірку. Якщо логін чи пароль невірний, виводиться відповідний запит. Код даної частини програми виглядає наступним чином:

```
app.post('/login', (req, res, next) => {  
  passport.authenticate('main', (err, user, info) => {  
    if (err) {
```

```

return next(err)
}
if (!user) {
res.cookie('failureMessage', 'Неверный логин или пароль')
return res.redirect('./login');
}
req.login(user, (err) => {
if (err) {
return next(err);
}
return res.redirect('./');
});
})(req, res, next);
})

```

2.6 Інструкція з запуску веб-додатку

Потрібно натиснути на кнопку «дебагу» і вибрати файл app.js (рис 2.5.1), натиснути F5

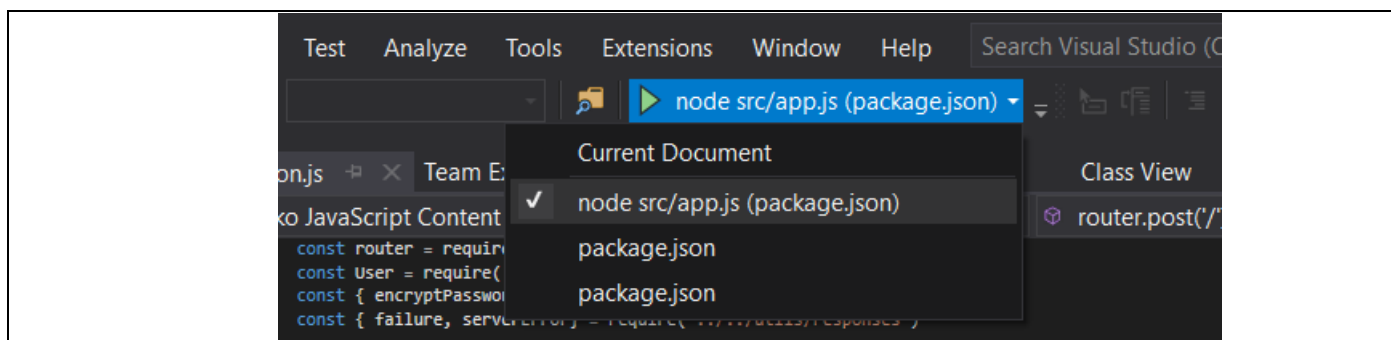


Рисунок 5.1– Виконання скрипту запуску серверу веб-додатку

ВИСНОВОК

Результатом даної роботи є практичне застосування веб - програмування для моделювання предметної області. Під час роботи було повністю реалізовано поставлену задачу – розробити веб-додаток для запису та виведення коментарів про книги, які опубліковані користувачами веб-додатку.

Проведено дослідження предметної області. На основі результатів розроблена нова структура представлення даних в інформаційних системах, створені моделі даних кожного розділу.

Таким чином, у ході даної роботи автору вдалося поглибити розуміння основних методів веб програмування та застосувати їх для цілей реального життя. Нам вдалося застосувати такі технології як MongoDB, NodeJS. Все це дозволило набути практичних навичок у веб програмуванні.

Для розгортання створеної програми достатньо мінімальних системних вимог, які на сьогодні має практично будь-який браузер

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Documentation Express [Електронний ресурс] — Документація веб-фреймворка:
<https://expressjs.com/ru/guide/routing.html>
2. Shelley Powers Learning Node. Moving to the Server Side, 2nd edition / Shelley Powers., 2016 — 228 с.
3. Shelley Powers Learning Node. Moving to the Server Side, 1st edition / Shelley Powers., 2014 — 396 с.
4. Documentation MongoDB [Електронний ресурс] — Документація офіційна бази даних MongoDB:
<https://docs.mongodb.com>
5. Metanit.com [Електронний ресурс] — Документація встановлення та підключення MongoDB:
<https://metanit.com/nosql/mongodb/1.1.php>
6. Итан Браун Веб-разработка с применением NODE и EXPRESS. Полноценное использование стека JavaScript / Итан Браун., 2017 — 336 с.
7. learn.javascript.ru [Електронний ресурс] — Інформаційний портал та онлайн-підручник з вивчення JavaScript:
<https://learn.javascript.ru/>

ДОДАТОК 1

Ремонт побутової техніки

Текст програмного модуля

УКР.НТУУ”КПІ” ім. І. Сікорського. ТЕФ АПЕПС КВ-7214

Аркушів 2

2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214		Пояснювальна записка
Комплекс		
Компоненти		
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214	app.js	Опис програмного модуля
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214	package.json	
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214	header.ejs	Модуль виведення верхньої частини інтерфейсу
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214	main.ejs	Модуль виведення основної частини інформації
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214	footer.ejs	Модуль виведення нижньої частини інтерфейсу
УКР.НТУУ“КПІ”. ТЕФ АПЕПС КВ-7214	login.ejs	Модуль реєстрації користувача

ДОДАТОК 2

Ремонт побутової техніки

Текст програмного модуля

УКР.НТУУ"КП" ім. І. Сікорського. ТЕФ АПЕПС КВ-7214

Аркушів 8

2019

```

app.js
const { join } = require('path')

require('dotenv').config({
  path: join(__dirname, '..', '.env'),
})

const express = require('express')
const cors = require('cors')
const db = require('./database')
const helmet = require('helmet')
const morgan = require('morgan')
const session = require('express-session')
const cookieParser = require('cookie-parser')
const { json, urlencoded } = require('body-parser')

const app = express()

app.use(cors())
app.use(helmet())
app.use(cookieParser())
app.use(session({
  secret: process.env.SECRET,
  resave: true,
  saveUninitialized: false
}))
app.use(json())
app.use(urlencoded({
  extended: true
}))
app.use(morgan('dev'))
app.use('/', express.static(join(__dirname, 'public')))
app.use(require('./routes'))

app.set('views', join(__dirname, 'views'))
app.set('view engine', 'ejs')

const port = process.env.PORT ? parseInt(process.env.PORT) : 3001

app.listen(port, () => console.log('listening on port : ', port))

process
  .on('unhandledRejection', (reason, p) => console.error(reason, 'Unhandled Rejection at Promise', p))

```

```

.on('uncaughtException', err => {
  console.error(err, 'Uncaught Exception thrown')
  process.exit(1)
})
.once('SIGINT', async () => {
  console.log('\nCtrl+C Server Stopped')
  await db.close()
  process.exit(0)
})

```

package.json

```

{
  "name": "nikselko",
  "version": "1.0.0",
  "description": "",
  "main": "src/app.js",
  "scripts": {
    "start": "node src/app.js"
  },
  "keywords": [],
  "author": "Nikselko",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cookie-parser": "^1.4.4",
    "cors": "^2.8.5",
    "dotenv": "^8.0.0",
    "ejs": "^2.6.1",
    "express": "^4.16.4",
    "express-session": "^1.16.1",
    "helmet": "^3.18.0",
    "mongoose": "^5.5.5"
  },
  "devDependencies": {
    "morgan": "^1.9.1"
  }
}

```

шаблон виведення верхньої частини інтерфейсу — header.ejs

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="Telegram bot's">
<meta name="author" content="Nikselko">
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon">
<title>CourseWork</title>
<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPM
O"
crossorigin="anonymous">
</head>
<body>
<header>
<div class="navbar navbar-dark bg-dark shadow-sm">
<div class="container d-flex justify-content-between">
<a href="/" class="navbar-brand align-items-center">
<strong>Nikselko</strong>
</a>
<% if(user !== false) { %>
<a href="/" class="navbar-brand align-items-center">
Username : <%=user.login%>
</a>
<% } %>
<button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarHeader" aria-controls="navbarHeader"
aria-expanded="false" aria-label="Toggle navigation">

<% if(isAuthorized) { %>
<% if(user.role === 'user') { %>
<a class="btn btn-dark" href="/orders/add" role="button">
Make Order
</a>
<% } %>
<a class="btn btn-dark" href="/authorization/logout" role="button">
Exit
</a>
<% } else { %>
<a class="btn btn-dark" href="/authorization/login" role="button">
Login
</a>
<a class="btn btn-dark" href="/authorization/registration" role="button">
Registration

```

```

        </a>
      <% } %>
    </button>
  </div>
</div>
</header>
</body>
</html>

```

шаблон виведення основної частини інформації веб-додатку — main.ejs

```

<main role="main">
  <div class="album py-5 bg-light">
    <div class="container">
      <div class="row">
        <% if(isAdmin) { %>
          <h2>
            Users
          </h2>
          <ol>
            <% for(let i = 0; i != users.length; i++) { %>
              <li>
                <div class="card-body">
                  <h3><%= users[i].login %> </h3>
                  <h5> <strong>Role:</strong> <%= users[i].role %></h5>
                  <% if(users[i].role !== 'admin') { %>
                    <a class="btn btn-sm btn-dark btn-secondary"
                      href="/users/admin/<%=users[i].id%>" role="button">
                      Update to Admin
                    </a>
                  <% if(users[i].role !== 'employee') { %>
                    <a class="btn btn-sm btn-dark btn-secondary"
                      href="/users/employee/<%=users[i].id%>" role="button">
                      Update to Employee
                    </a>
                  <% } else { %>
                    <a class="btn btn-sm btn-dark btn-secondary"
                      href="/users/user/<%=users[i].id%>" role="button">
                      Make user
                    </a>
                  <% } %>
                  <a class="btn btn-sm btn-dark btn-secondary"
                    href="/users/delete/<%=users[i].id%>" role="button">
                    Delete

```



```

        </a>
        <% } %>

    </div>
</li>
<hr>
<% } %>
</ol>
<hr>
<% } %>
<h2 class="user-orders">
    Orders
</h2>
<ol>
<% for(let i = 0; i != items.length; i++) { %>
<li>
    <div class="card-body">
        <h3><%= items[i].title %></h3>
        <h5><strong>Author:</strong> <%= items[i].author %></h5>
        <h5><strong>Description</strong>: <%= items[i].text %></h5>
        <% if(items[i].status) { %>
            <h6><strong>Status:</strong> Done!</h6>
        <% } else { %>
            <h5><strong>Status:</strong> Active</h5>
            <% if(user.role === 'employee') { %>
                <a class="btn btn-sm btn-dark btn-secondary"
                    href="/orders/do/<%=items[i].id%>" role="button">
                    Do it
                </a>
            <% } %>
        <% } %>
        <% if(user.role === 'user') { %>
            <a class="btn btn-sm btn-dark btn-secondary"
                href="/orders/update/<%=items[i].id%>" role="button">
                Change status
            </a>
        <% } %>
        <% if(items[i].author === user.login || isAdmin || user.role ===
'employee') { %>
            <a class="btn btn-sm btn-dark btn-secondary"
                href="/orders/edit/<%=items[i].id%>" role="button">
                Edit
            </a>
            <a class="btn btn-sm btn-dark btn-secondary"

```

```

        href="/orders/delete/<%=items[i].id%>" role="button">
        Delete
    </a>
    <% } %>
</div>
</li>
<hr>
<% } %>
</ol>
</div>
</div>
</div>
</main>

```

шаблон виведення нижньої частини інтерфейсу — footer.ejs

```

<footer class="text-muted text-center">
    <div class="container ">
        <p class="float-center">
            <a href="#">
                Back to top
            </a>
        </p>
    </div>
</footer>

</body>
</html>

```

шаблон реєстрації користувача login.ejs

```

<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="Sign Up">
    <meta name="author" content="Nikselko">
    <link rel="icon" href="/images/favicon.ico">
    <title>Log In</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
        rel="stylesheet"

```

```

        integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPM
O"
        crossorigin="anonymous">
    <link href="css/signin.css" rel="stylesheet">
</head>

<body class="text-center">
<form id="myform" method="post" action="/authorization/login">
    <br>
    <h1 class="h3 mb-3 font-weight-normal">
        Sign in
    </h1>
    <label for="inputLogin" class="sr-only">
        Login
    </label>
    <input type="text" name="login" id="inputLogin"
        placeholder="Enter Login..." minlength="8" maxlength="20" required autofocus>
    <hr>
    <label for="inputPassword" class="sr-only">
        Password
    </label>
    <input type="password" name="password" id="inputPassword"
        placeholder="Enter Password..." minlength="8" maxlength="20" required>
    <hr>
    <button type="submit" class="btn btn-lg btn-primary">
        Sign in
    </button>
    <a class="btn btn-lg btn-dark btn-secondary"
        href="/authorization/registration" role="button">
        Registration
    </a>
    <hr>
</form>
</body>
</html>

```

ДОДАТОК 3

Ремонт побутової техніки

Текст програмного модуля

УКР.НТУУ”КПІ” ім. І. Сікорського. ТЕФ АПЕПС КВ-7214

Аркушів 7

2019

АНОТАЦІЯ

Розроблений модуль адміністратора є незалежним під модулем системи комплексного абонементу бібліотеки. Він може працювати як частина системи, взаємодіючи з іншими модулями через спільну базу даних, так і незалежно від інших користувачів, реалізуючи базовий функціонал модуля.

Засоби розробки: середовище розробки Visual Studio 2019 Community Preview, мова програмування Node.js JavaScript, система керування базою даних MongoDB.

ЗМІСТ

1. Загальні відомості.....	39
2. Функціональне призначення.....	40
3. Вхідні данні.....	41
4. Вихідні данні.....	42

1. ЗАГАЛЬНІ ВІДОМОСТІ

Найменування програмного модуля — “Адміністратор”. Мова програмної реалізації — Node.js JavaScript. Середовище розробки — Visual Studio 2019 Community Preview.

Для використання програмного забезпечення необхідно :

- комп’ютер з довільною ОС;
- доступ до мережі інтернет.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Модуль був створений для завантаження книг адміністратором, та продивлятися книги користувачем.

Адміністратор має змогу:

- змінити статус заказу;
- видалити заказ;
- продивлятися інформацію заказу;
- змінювати інформацію заказу;
- переводити Користувача у статус працівника та навпаки.
- переводити Працівника у статус Адміністратора та навпаки.

Працівник має змогу:

- змінити статус заказу;
- продивлятися інформацію заказу;
- змінювати інформацію заказу;

Користувач має змогу:

- додавати заказ;
- змінити статус заказу;
- видалити заказ;
- продивлятися інформацію заказу;
- змінювати інформацію заказу;

3. ВХІДНІ ДАНІ

Вхідні дані є заголовок заказу, текст, які зберігається у базі даних. (рисунок 3.1).

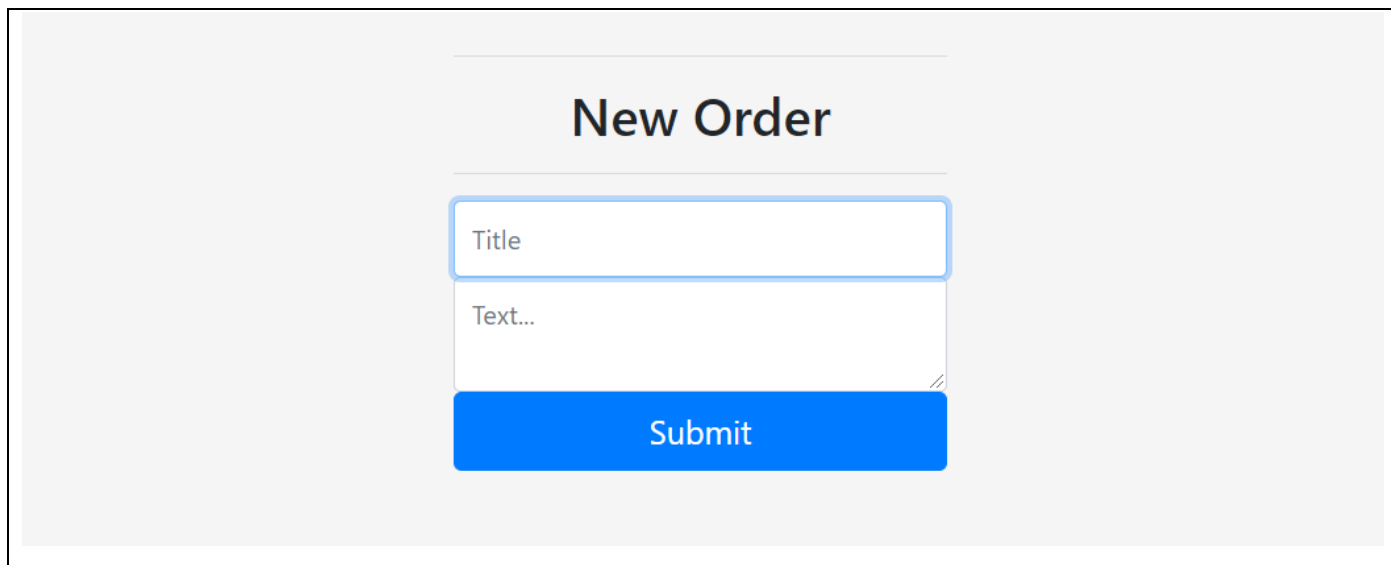
The image shows a web form titled "New Order" centered on a light gray background. The form itself is a white rectangle with a thin gray border. At the top of the form, the title "New Order" is displayed in a bold, black, sans-serif font. Below the title, there are two input fields. The first field is labeled "Title" in a small, gray font and has a light blue border. The second field is labeled "Text..." in a small, gray font and has a light gray border. To the right of the "Text..." field, there is a small icon of two overlapping lines. Below these two fields is a solid blue button with the word "Submit" written in white, centered text.

Рисунок. 3.1 – Сторінка створення нового заказу

4. ВИХІДНІ ДАНІ

Вихідними даними програмної системи є заказ: заголовок, замовник, склад заказу та статус (рисунок 4.1).

The screenshot displays a web application interface for managing orders. At the top, a dark header bar contains the text 'Nikselko' on the left, 'Username : 123123123' in the center, and an 'Exit' button on the right. Below the header, the main content area is titled 'Orders'. It lists a single order with the following details:

- Order ID: 1. LD FHT1208SNL
- Author: 12312312
- Description: New LD washing machine, i think the washing drum is broken. PLS do something our clothes stink!
- Status: Active

Below the order details, there are three buttons: 'Do it', 'Edit', and 'Delete'. At the bottom of the main content area, there is a 'Back to top' link.

Рисунок 4.1 — Сторінка редагування запису в веб-додатку