Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З комп’ютерного практикуму № 1 з дисципліни

**«Безпека програмного забезпечення»**

Виконав студент ІП-13 Музичук Віталій Андрійович

(шифр, прізвище, ім'я, по батькові)

Перевірив Соколовський Владислав Володимирович

(прізвище, ім'я, по батькові)

Дата оцінювання \_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2024

**Лабораторна робота 1**

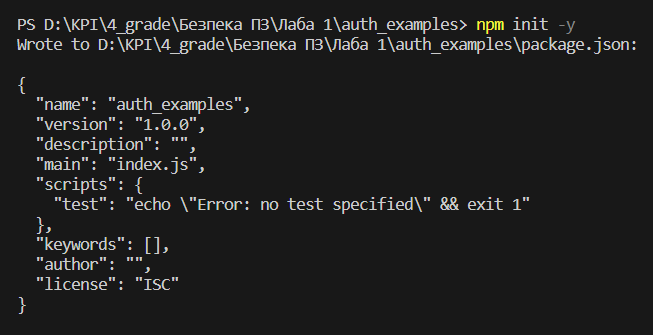
**Мета**: Роздивитись основні методи авторизації

**Завдання**: Викачати репозиторій з лекціями <https://github.com/Kreolwolf1/auth_examples>. Запустити кожен з 3 аплікейшенів та зробити скріншити запитів до серверу.

**Для отримання додаткового балу**: модифікувати token\_auth аплікейшен змінивши токен на JWT.

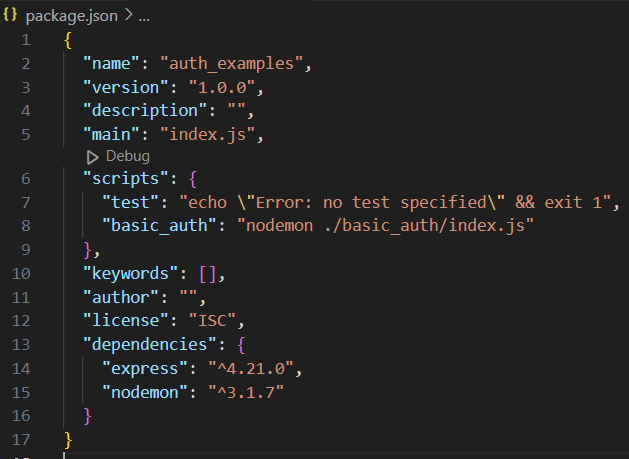
**Хід роботи**

Для початку, щоб запустити додатки ініціалізуємо проєкт через npm, куди будемо встановлювати необхідні залежності для кожного аплікейшину.

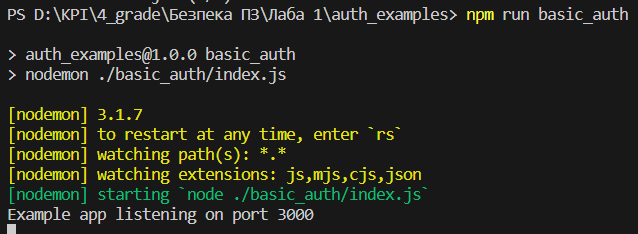


Додаток **basic\_auth**

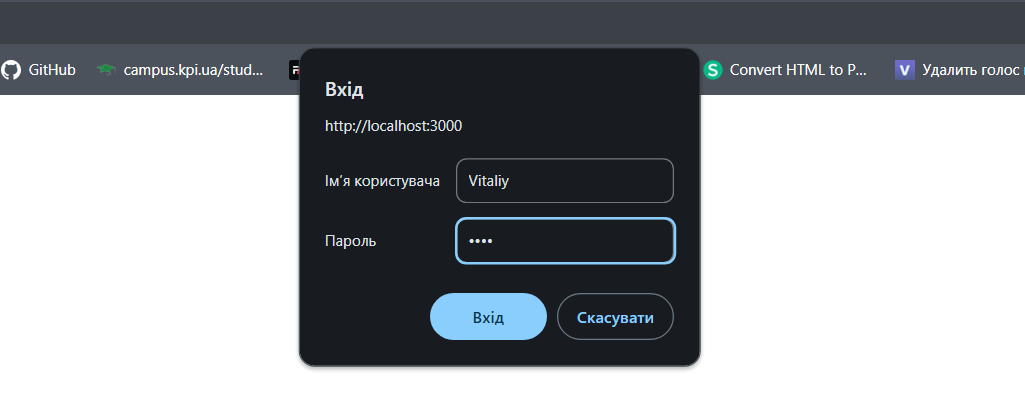
Для цього додатку встановимо базово express та nodemon й задамо скрипт для його запуску. Файл package.json виглядатиме наступним чином:

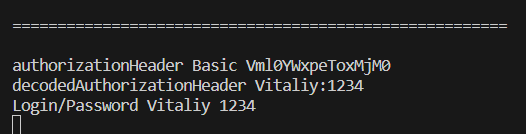


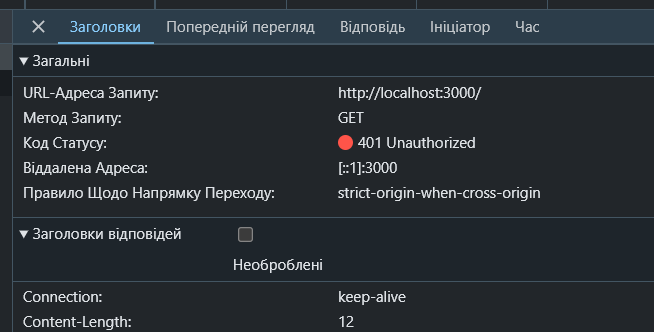
Запускаємо аплікейшн



Отже сервер працює, спробуємо авторизуватися. Спочатку введемо невірного користувача.

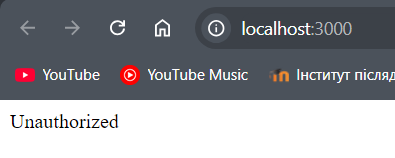


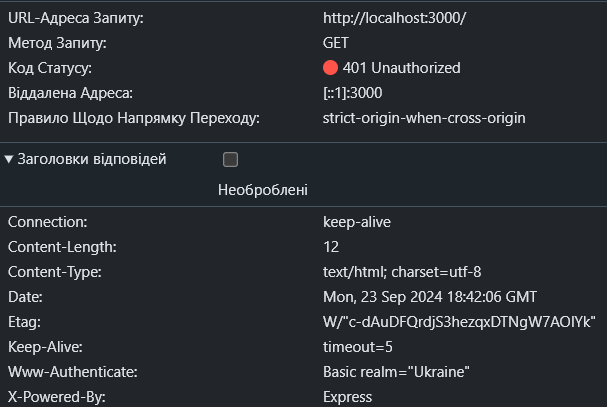




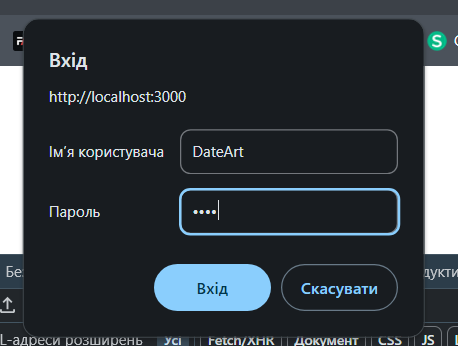
Інформація на сервер про користувача прийшла, проте він повернув 401 помилку, оскільки логін й пароль були неправильні.

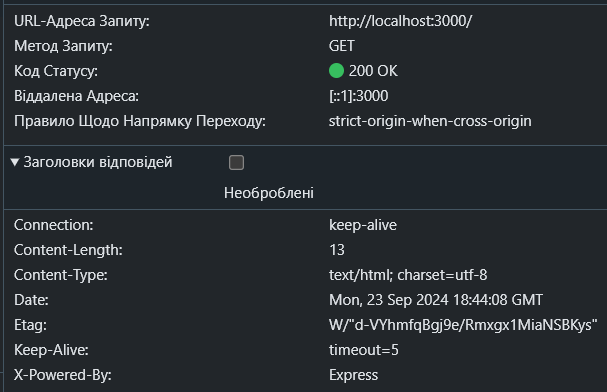
Якщо не заповнювати форму авторизації, то сервер поверну ту саму помилку й покаже на екрані, що користувач не авторизований.





Тепер введемо правильні дані користувача.





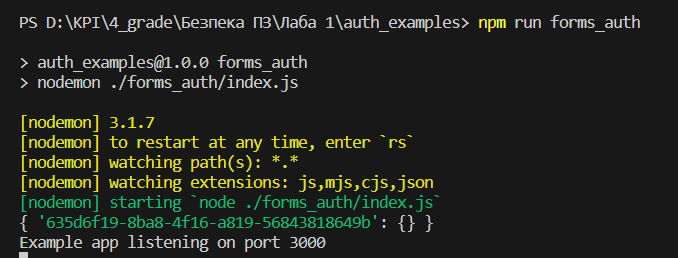
Тепер сервер дав відповідь 200 й не додав заголовку WWW-Authenticate: 'Basic realm="Ukraine".

Додаток **forms\_auth**

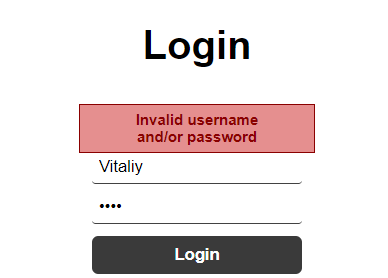
Для цього запуску нам знадобляться наступні залежності



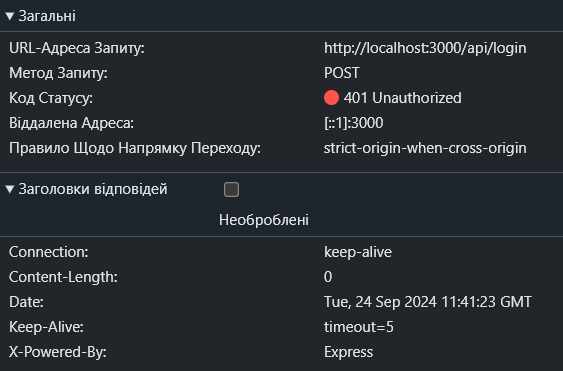
Запустимо додаток



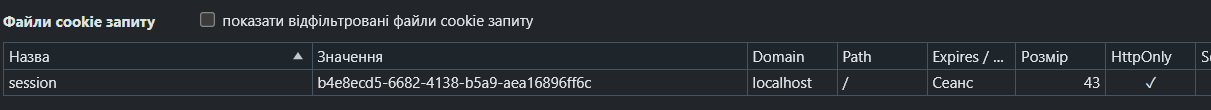
Авторизуємося як неправильний користувач



Авторизація невірна, тому сервер повернув нам 401 помилку

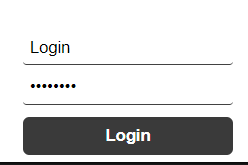


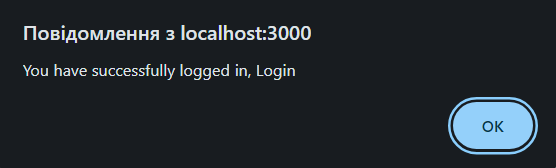
Також в куках й файлі session.json було збережено значення sessionId

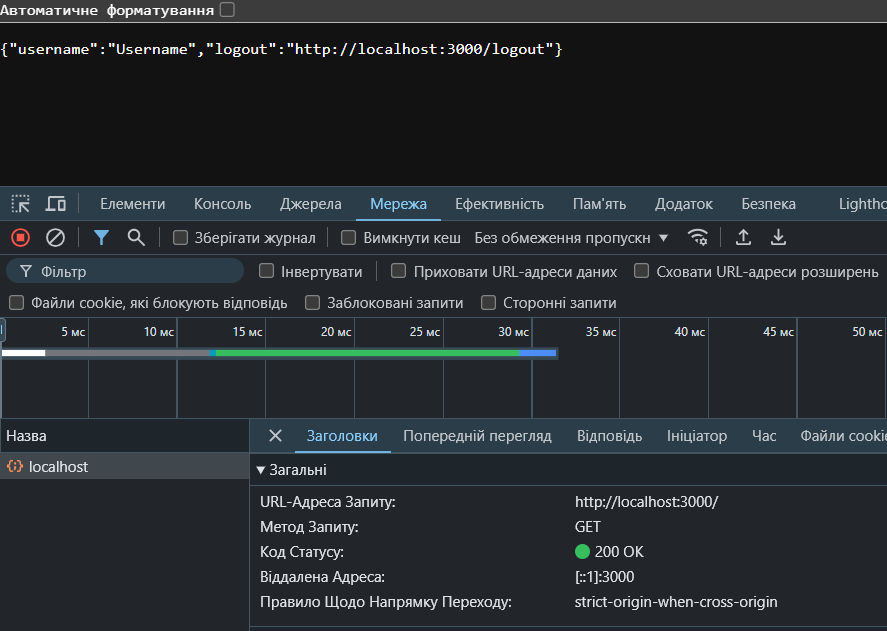




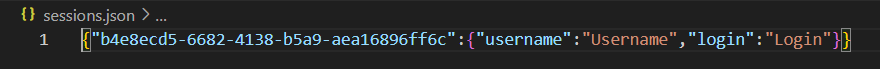
Тепер авторизуємося як валідний користувач



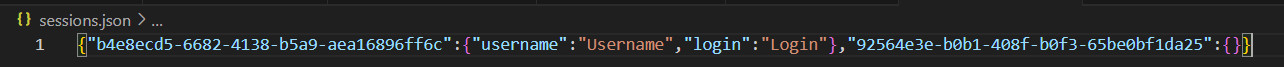




Тепер сервер повернув відповідь 200, username користувача й шлях, за яким можна розлогінитися. Тепер у файлі сесії, можна побачити, що там зберігається логін та юзернейм авторизованого користувача.

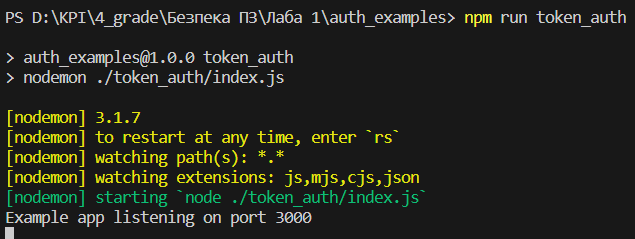


Якщо перейти за адресою /logout, сервер видалить куки з нашою авторизацією і у файлі сесій ми побачимо, що нам користувачу присвоєний новий sessionId без авторизації.

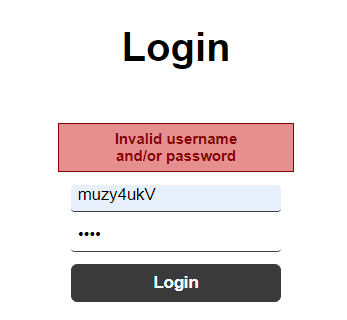


Додаток **token\_auth**

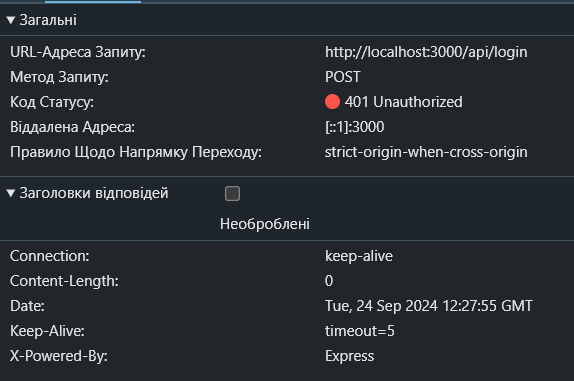
Залежності залишаються такими самими. Запустимо проект



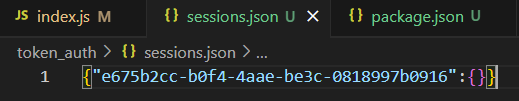
Аплікейшн успішно запущений, файл для збереження сесій був створений. Авторизуємося як невірний користувач



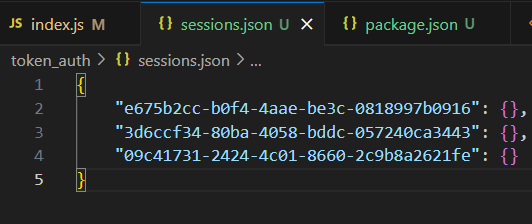
Сервер повернув 401 відповідь, але на цей раз куки з номером сесії не створювалися.



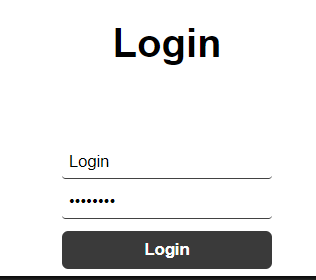
Сесія була збережена у файлі session.json на сервері.

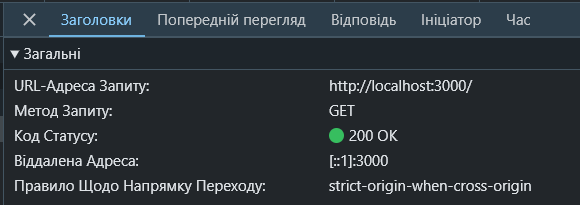


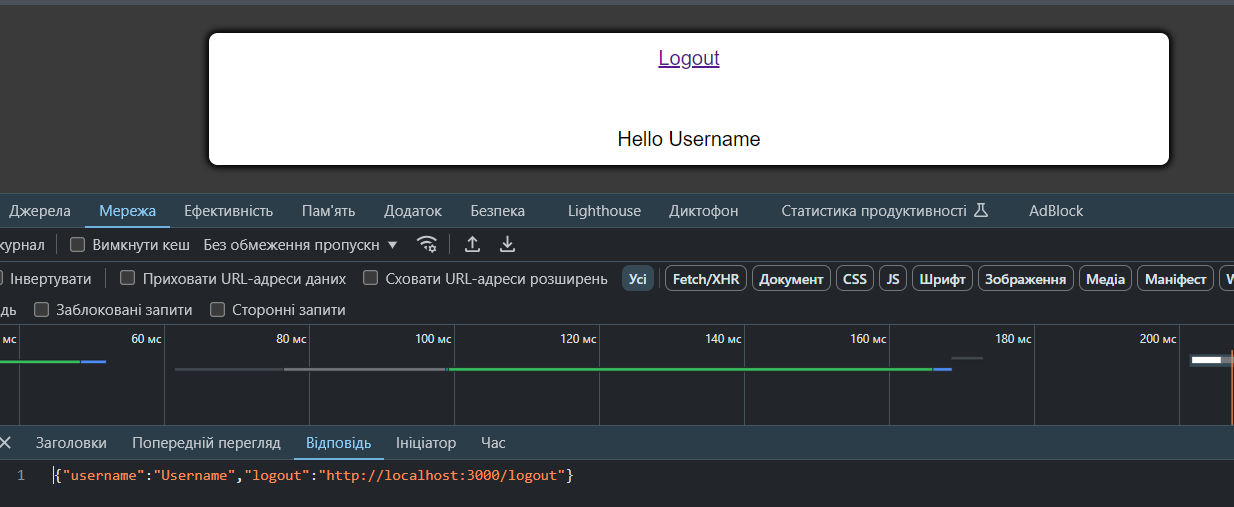
Звідси виходить, якщо кожен раз перезавантажувати сторінку, сервер буде створювати нові сесії, оскільки браузер як клієнт не зберігає інформацію про попередню сесію, яку сервер надіслав.



Тепер залогінимося, як правильний користувач

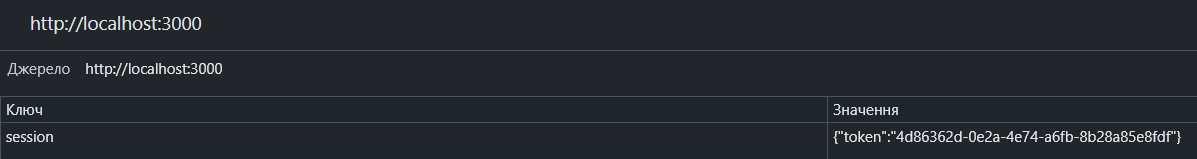






Отже авторизація пройшла успішно, сервер відповів 200 й привітався з нами )

Токен авторизації зберігається клієнтом у sessionStorage й доступний, поки відкрита вкладка.



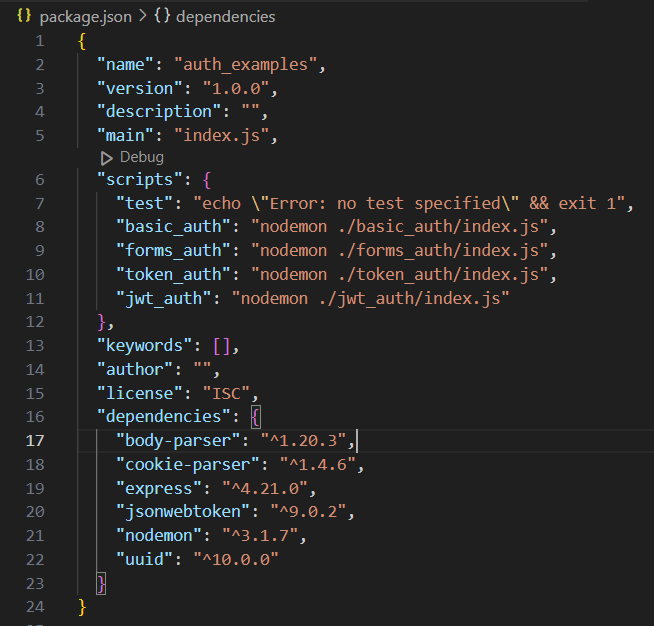
В даному додатку токен аутентифікації передавався через заголовок Authorization. Після переходу за /logout ця сесія видаляється з клієнтської сторони і ми стаємо знову як неавторизований користувач.

**Додаткове завдання:**

Додаток **jwt\_auth**

Модифікуємо попередній додаток **token\_auth**, змінивши токен на JWT.

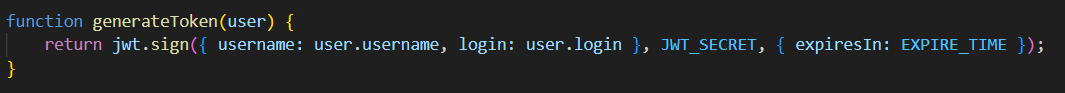
Спочатку завантажимо необхідний модуль. Тепер package.json виглядатиме наступним чином.



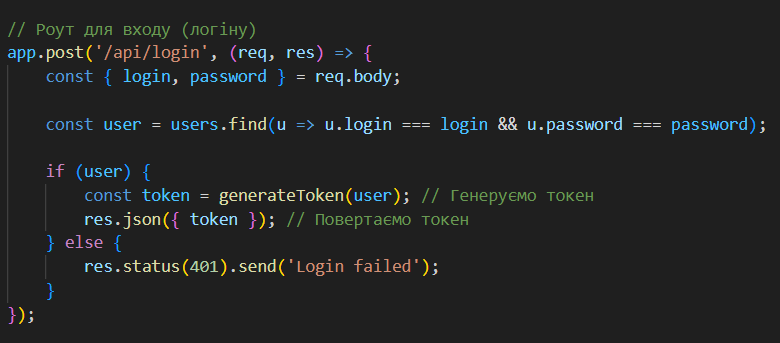
Додаємо секретний ключ та час життя для JWT токену.



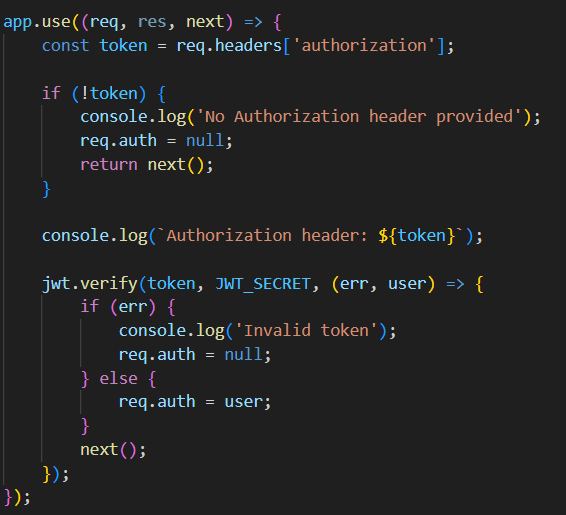
Визначаємо функцію для створення JWT токену



Створюємо JWT токен при кожному логіні

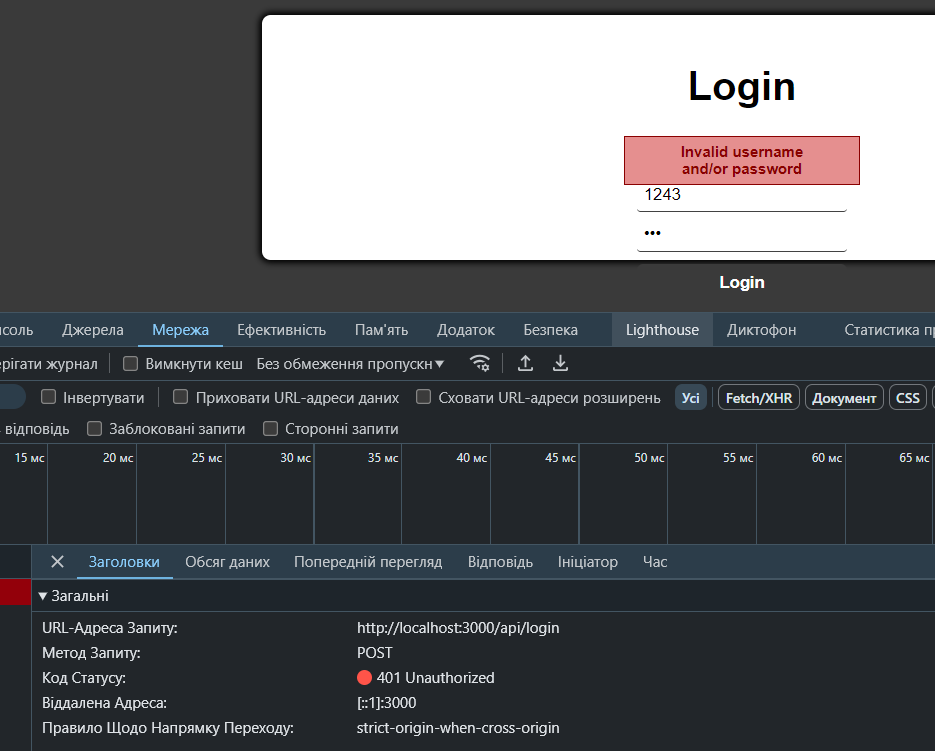


Змінюємо middleware так, щоб перевіряти JWT токен на валідність

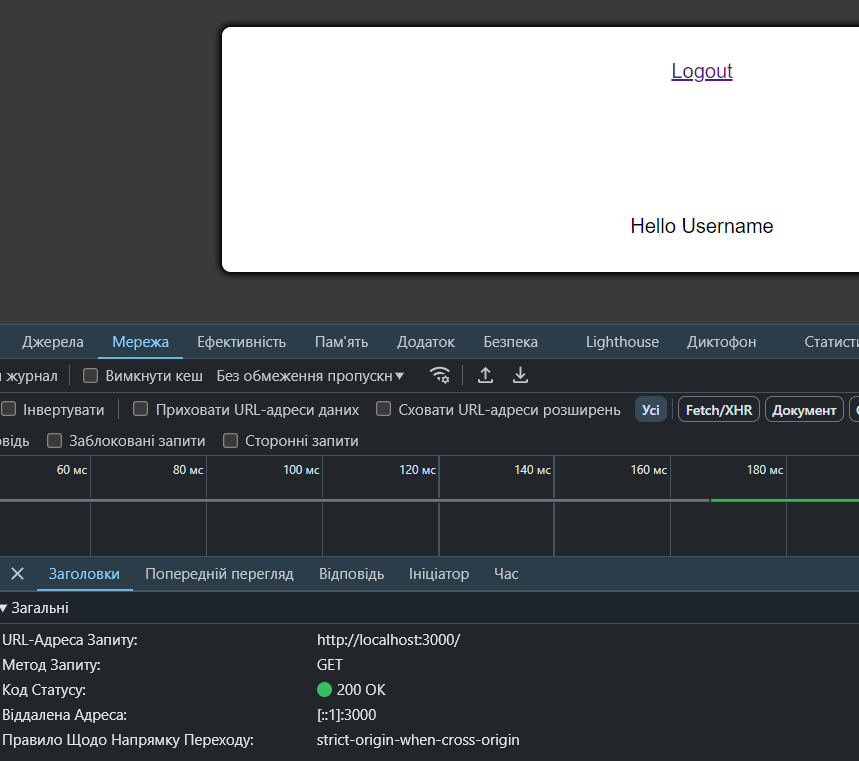


Тепер запустимо додаток й перевіримо його роботу.

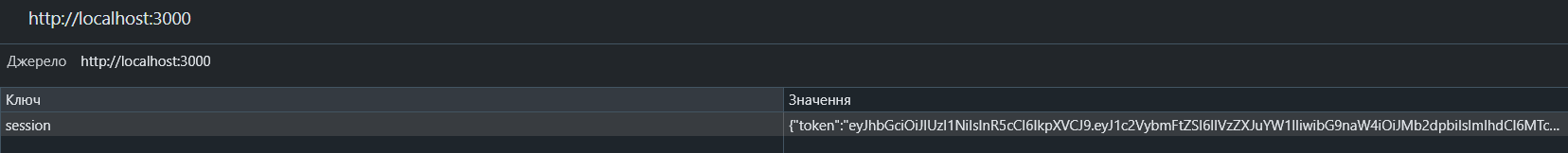
Якщо ввести невірні дані, то отримаємо 401 помилку – Не авторизований



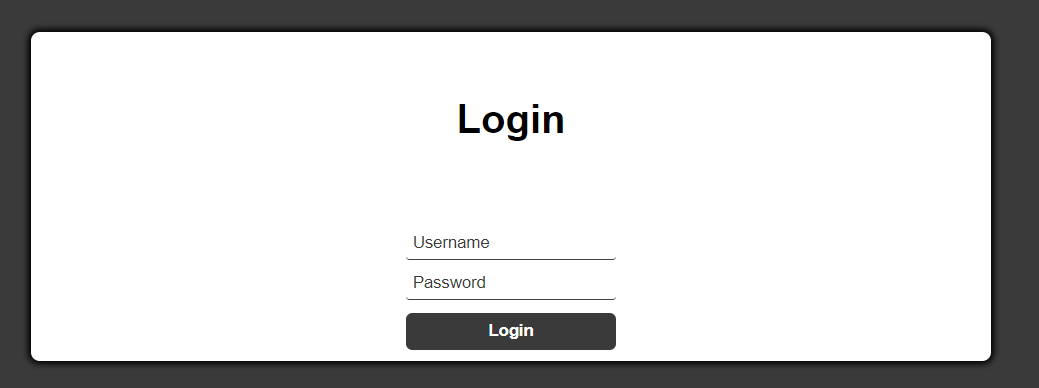
При введені правильних авторизаційних даних отримуємо відповідь 200 і авторизаційний JWT токен, який зберігається в пам’яті сеансу.

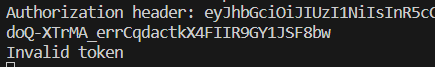


Ось токен в пам’яті сеансу



Після закінчення часу життя JWT токена, сайт не авторизовує нас, а в консолі сервера можемо побачити лог про неправильний токен.





**Висновок**

У ході лабораторної роботи було досліджено основні методи авторизації: Basic Auth, Forms Auth і Token Auth. Кожен із них використовує різні підходи для збереження та передачі даних про користувача між сервером і клієнтом.

Basic Auth передає логін і пароль у заголовку кожного запиту, тоді як Forms Auth використовує сесії, зберігаючи їх на сервері. Token Auth використовує токени, які клієнт передає в заголовках запитів, що усуває потребу в серверних сесіях.

Для додаткового завдання було модифіковано додаток із впровадженням JWT (JSON Web Token), що забезпечує безпечну передачу інформації про користувача. JWT підписується сервером і передається клієнтом через заголовок. Перевірка токена на сервері дозволяє аутентифікувати користувача без використання сесій.

У результаті всі завдання виконані, включаючи реалізацію сучасної JWT авторизації.