Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра Обчислювальної Техніки

Лабораторна робота №1

з дисципліни "Безпека програмного забезпечення"

Тема: "Базова аутентифікація"

Виконав:

студент групи ІП-93

Домінський В.О.

Київ 2022

**Зміст**

[Виконання: 3](#_Toc122556203)

[1. Basic\_auth 3](#_Toc122556204)

[2. Forms\_auth 5](#_Toc122556205)

[3. Token\_auth 7](#_Toc122556206)

[4. Jwt\_auth 8](#_Toc122556207)

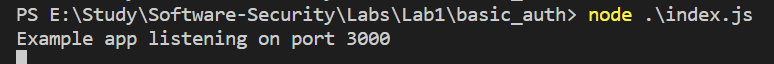
[Висновок: 12](#_Toc122556208)

[Посилання: 13](#_Toc122556209)

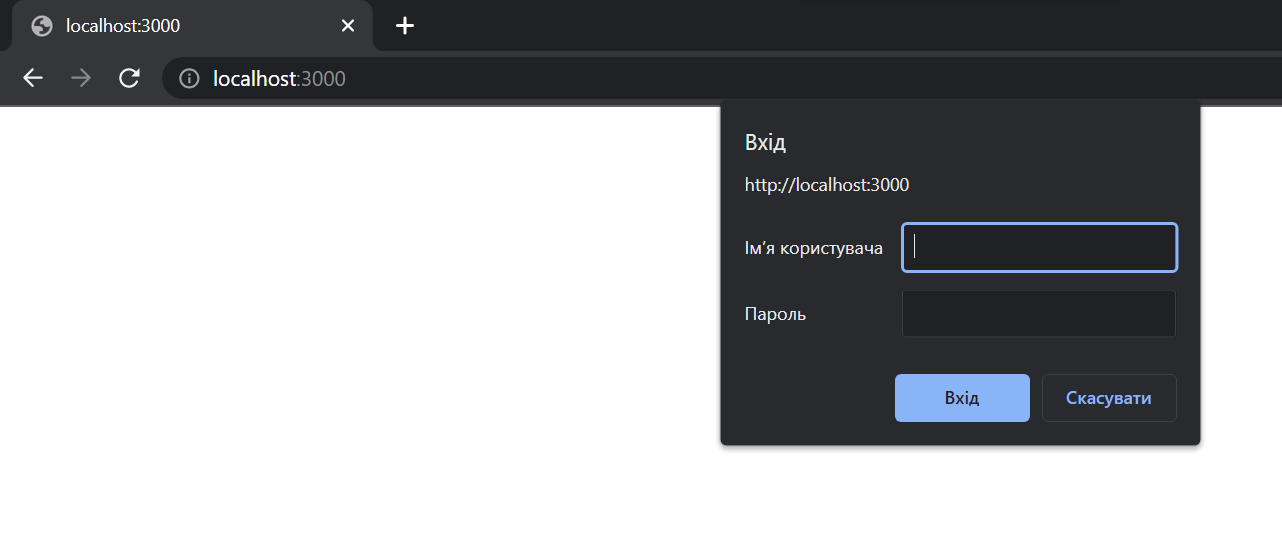
### Виконання:

## Basic\_auth

Для початку давайте запустимо даний файл:

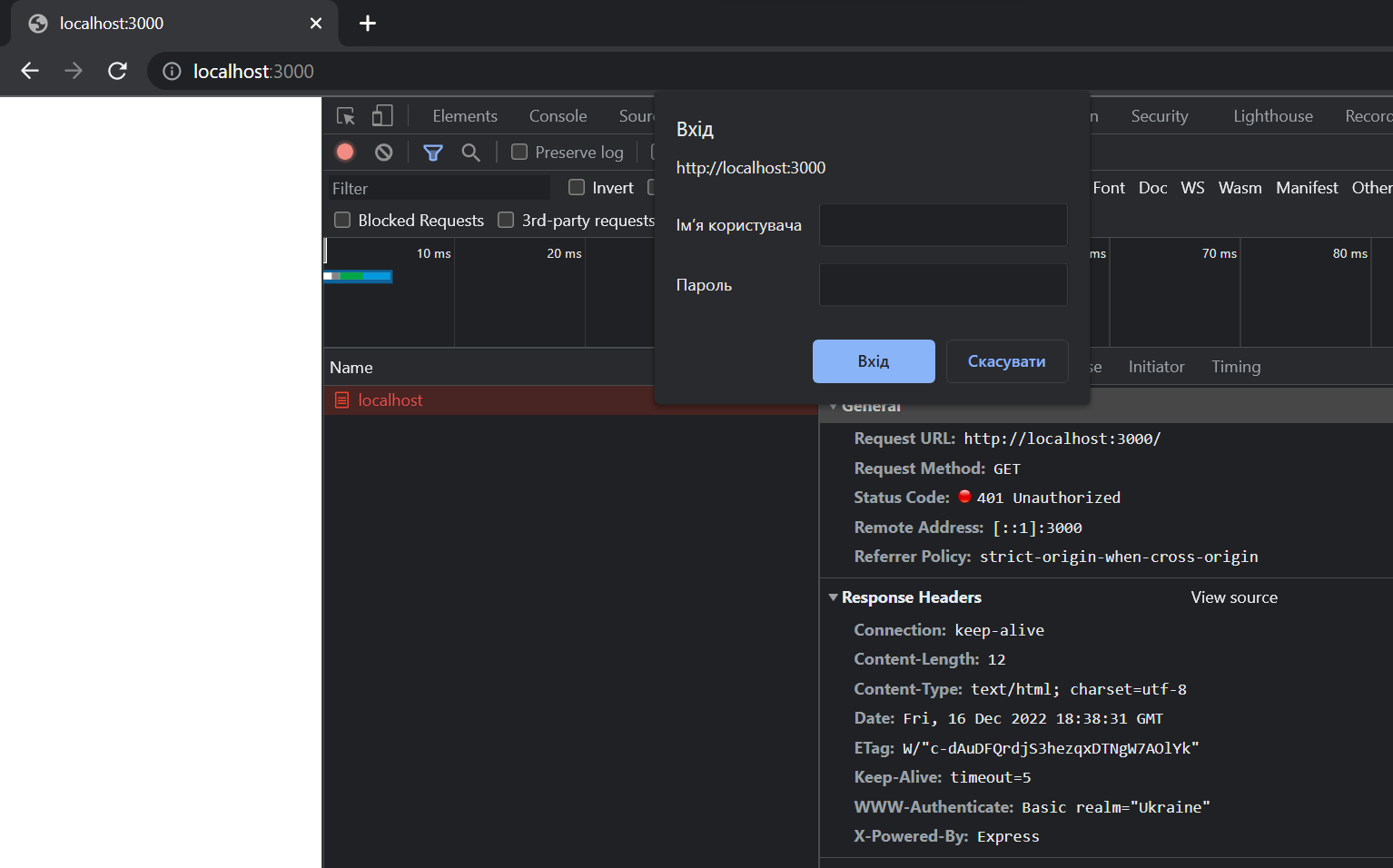


Як видно, він почав працювати на порту 3000

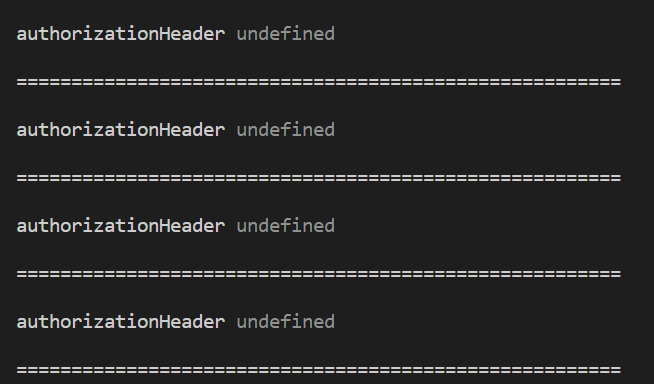


Якщо Ми зайдемо в Response Headers (тобто у відповідях), то можемо помітити ось такі важливі значення:

* Status Code – 401
* Вид аутентифікації – Basic

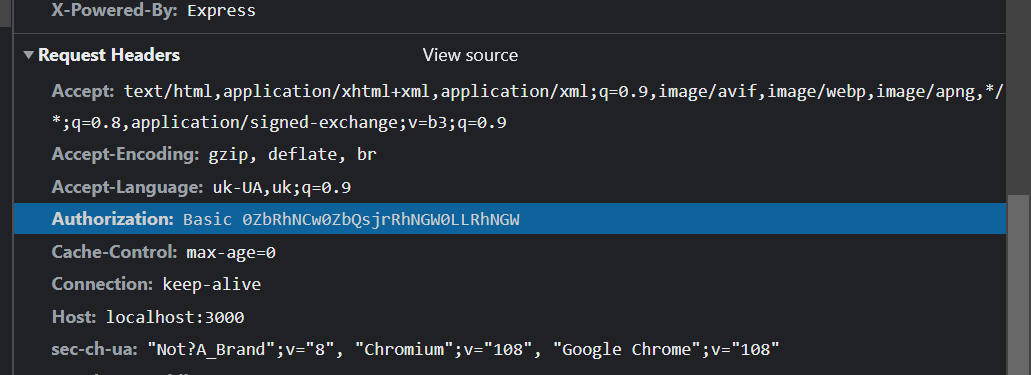


Консоль теж без діла не лежить:

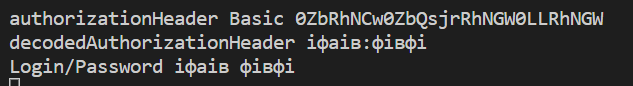


Ось тут виводиться значення одного з хедерів (Authorization), при якому запускається формування Header з базовою аутентифікацією

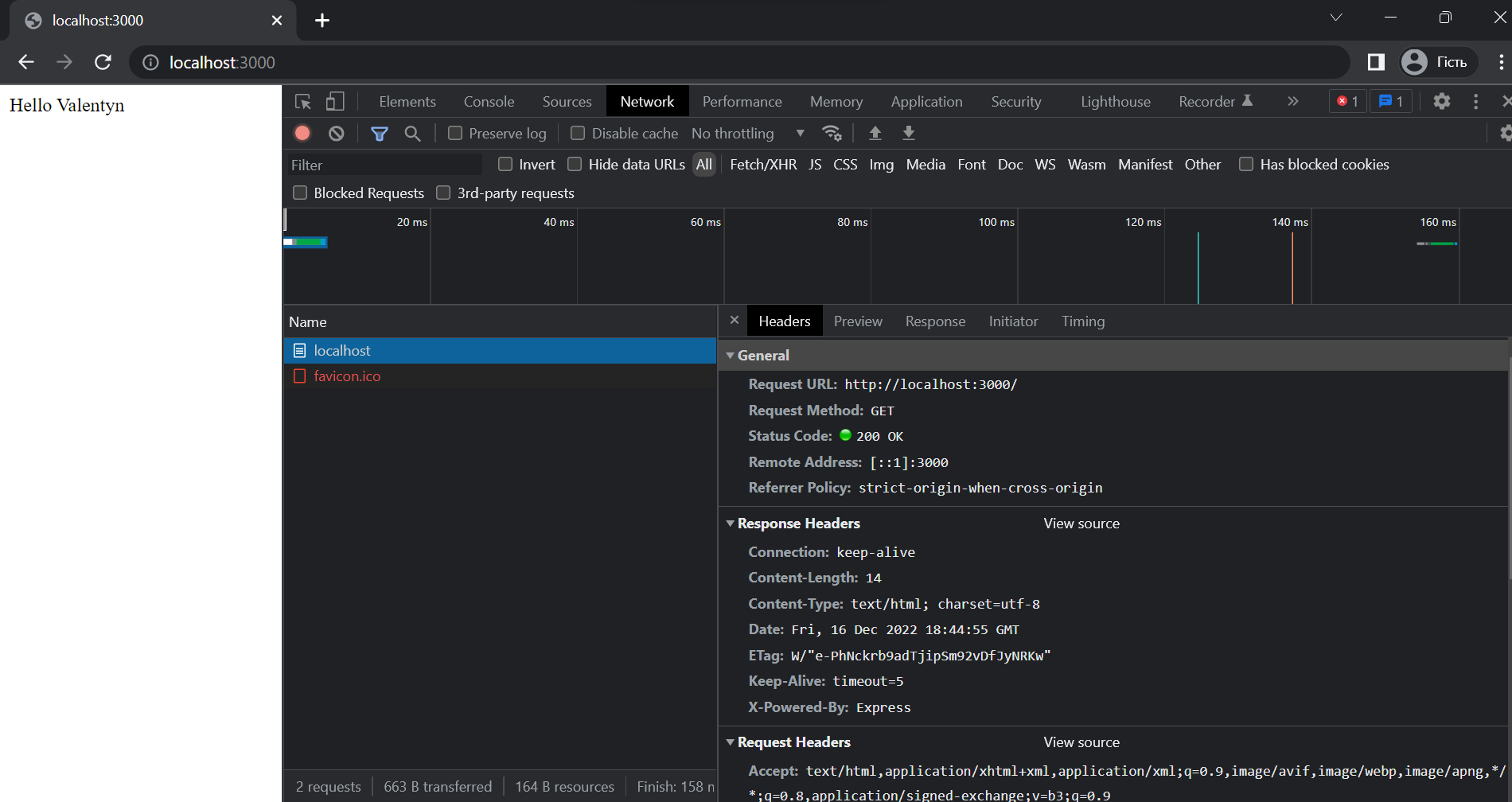
Тепер давайте спробуємо написати неправильні дані. На перший погляд майже нічого не змінилося – додався лише текст на самій сторінці, але це не так! Якщо проглянути Headers, то можна знайти новий хедер від браузеру з назвою «Basic» та Base64 хешом :

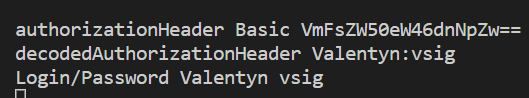


А в консолі Ми можемо побачити ще й розшифровані значення:



Тепер вводимо правильні значення:

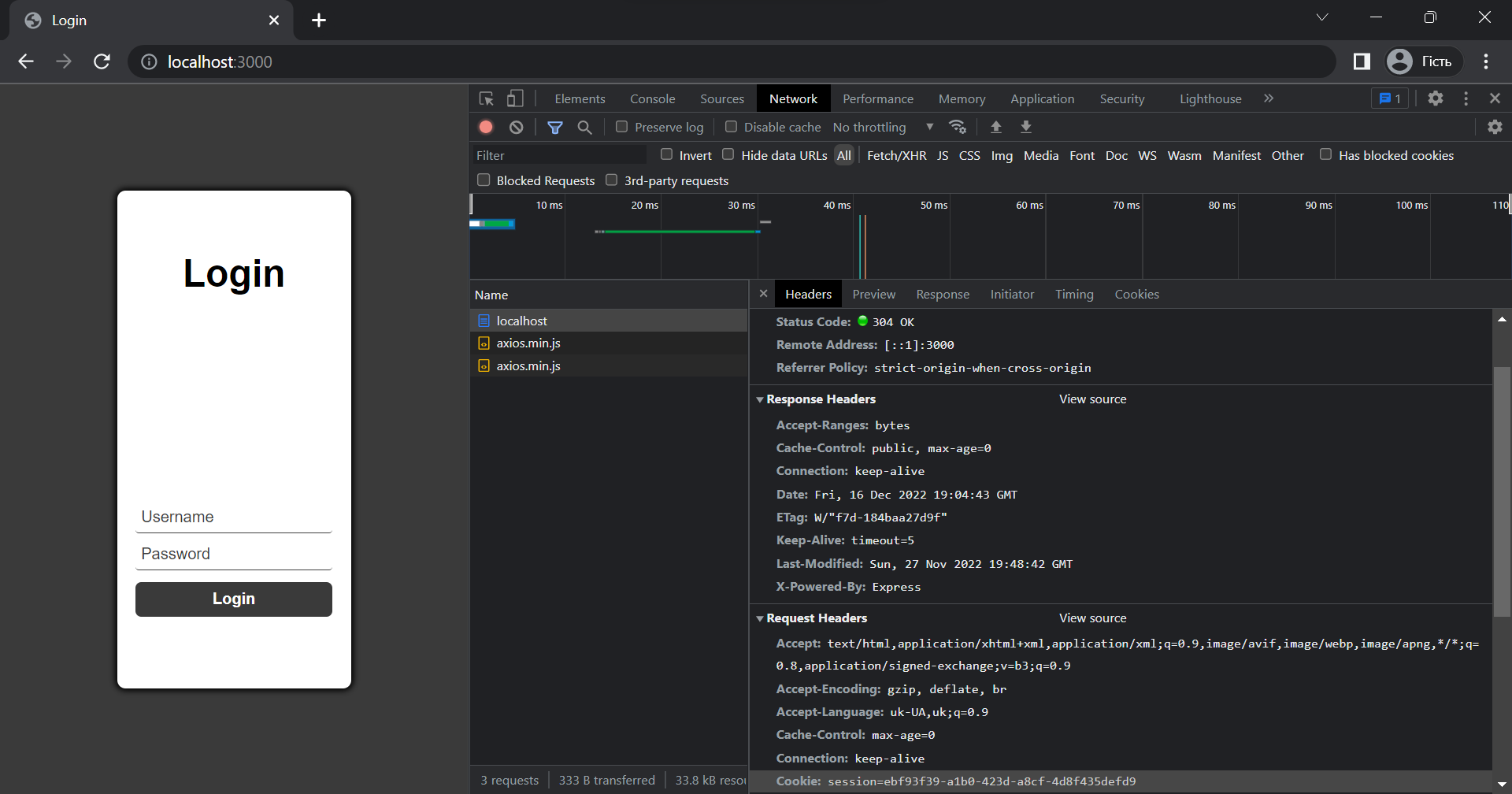




Таким чином Ми зробили найлегшу аутентифікацію!

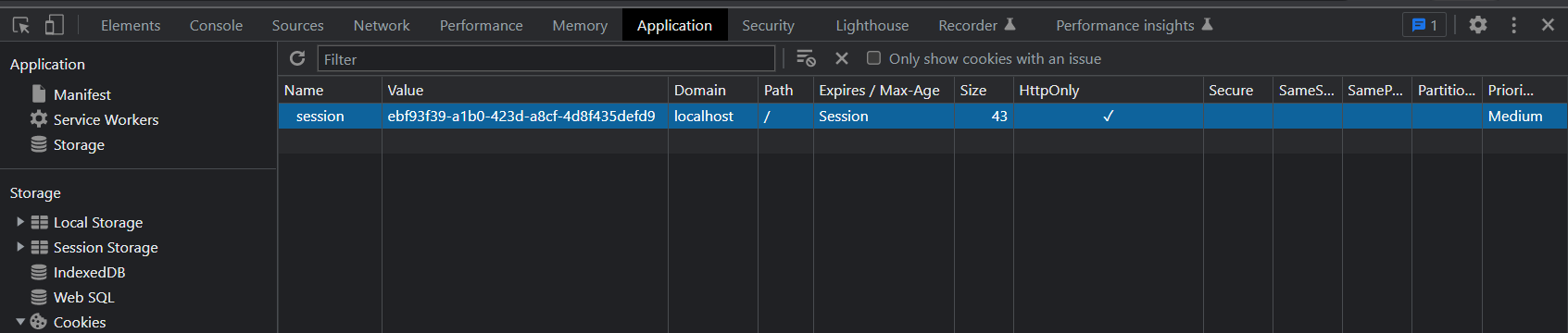
## Forms\_auth

При запуску Нас зустрічає ось таке вікно:



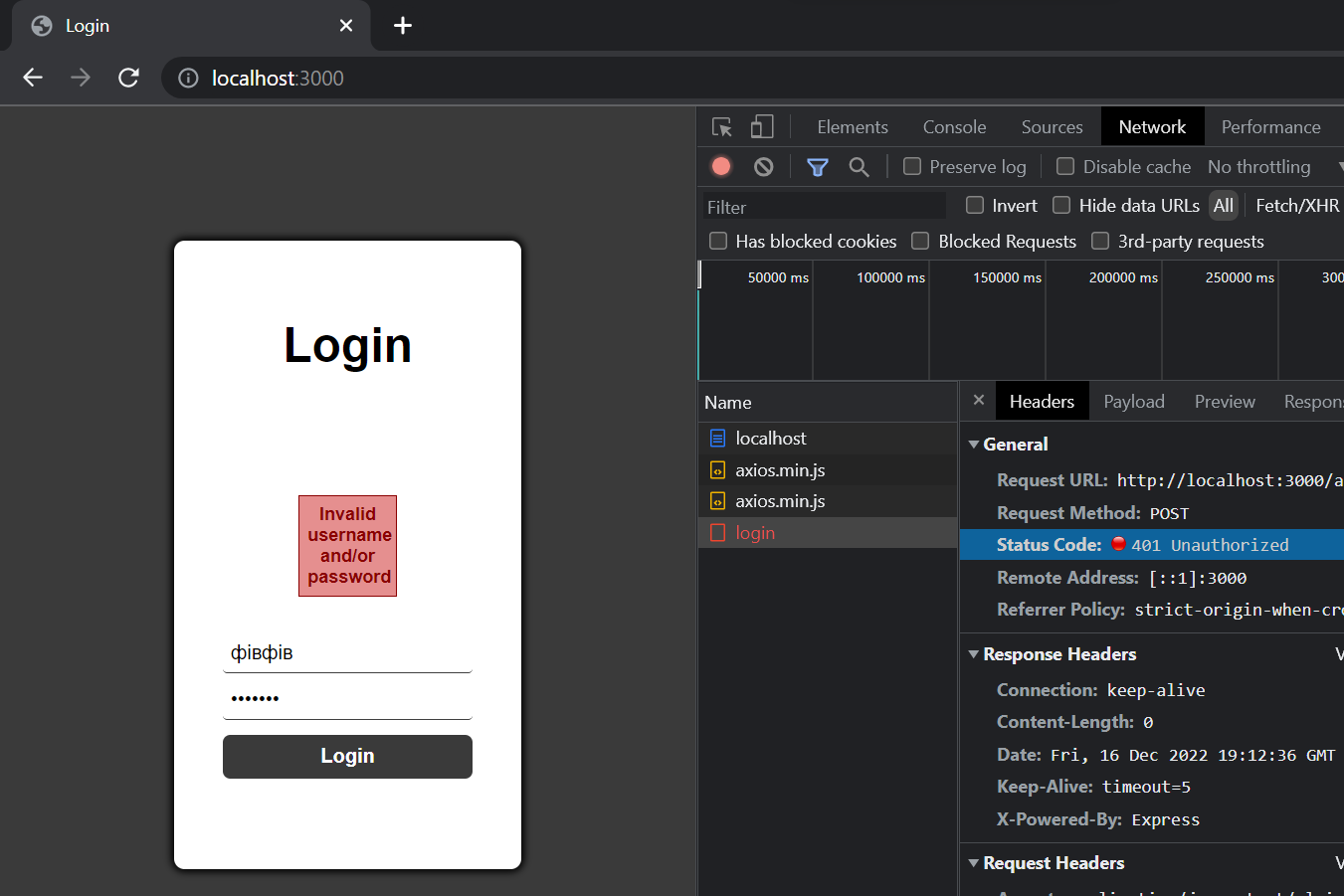
Можна помітити різницю між двома application у тому, що зараз Наш статус код – 304 та створюється значення cookie, тобто позначили цей браузер та сесію певним токеном і додатково створили файл sessions.json, куди його і помістили.

Відкривши додаткове меню з cookies Ми можемо знайти багато інформації щодо Наших токенів:



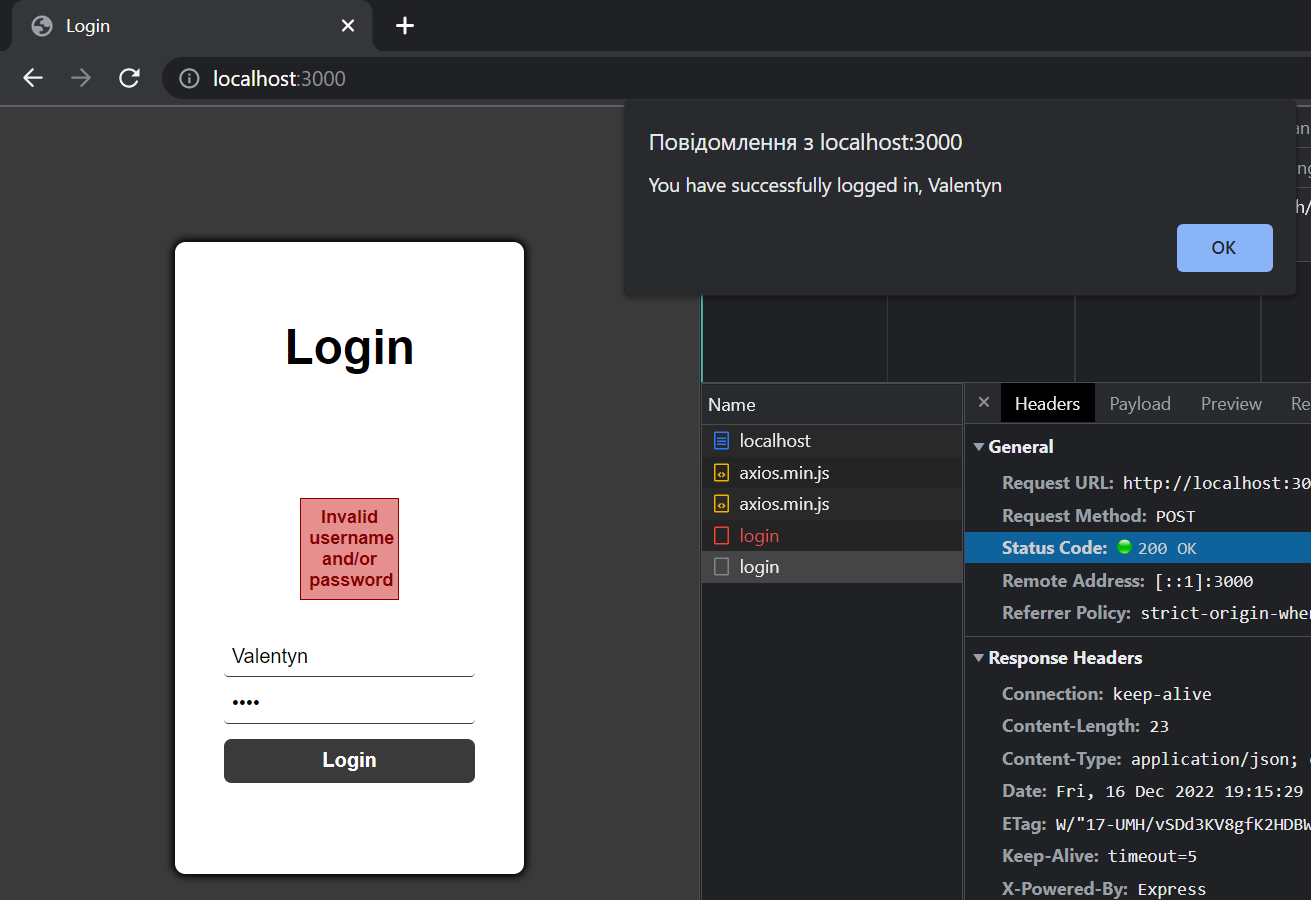
У Нашого cookie є поле HttpOnly, що означає, що вона буде постійно надсилатися на цей домейн, але її не можна редагувати з браузера

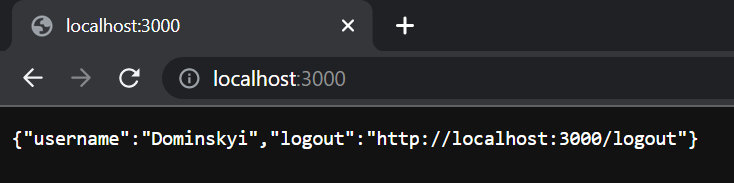
Тепер введемо неправильні логін та пароль:

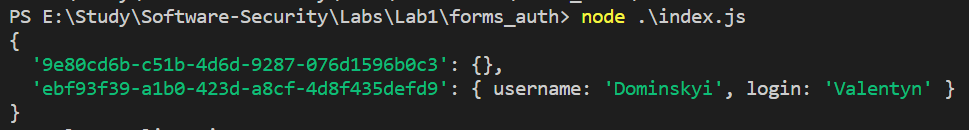


… та отримаємо загальний текст помилки і статус код 401.

Тепер зробимо все вірно:





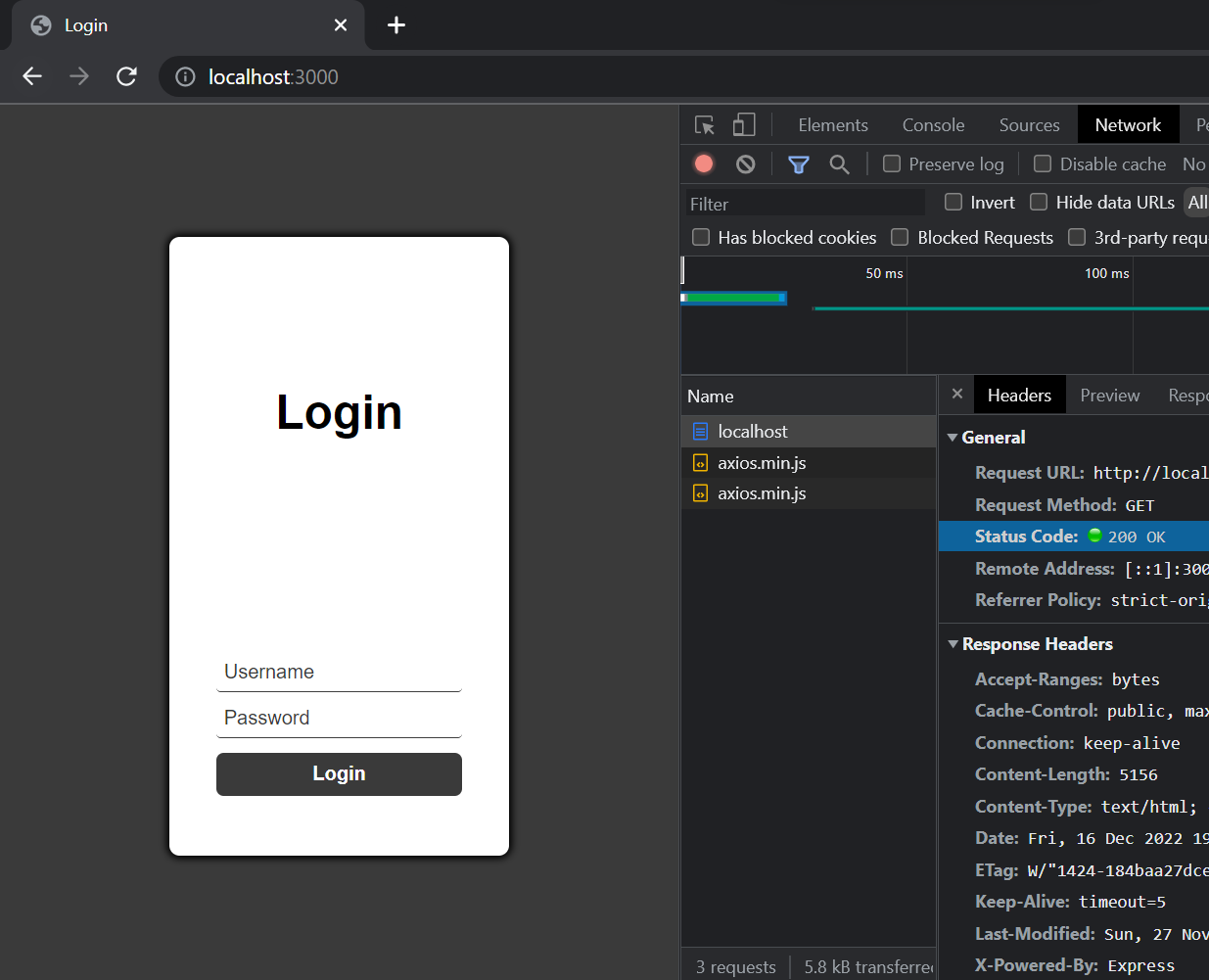


Як бачимо перед Нами з’являється нове вікно з повідомленням про вдалу авторизацію та інформацію про Нас.

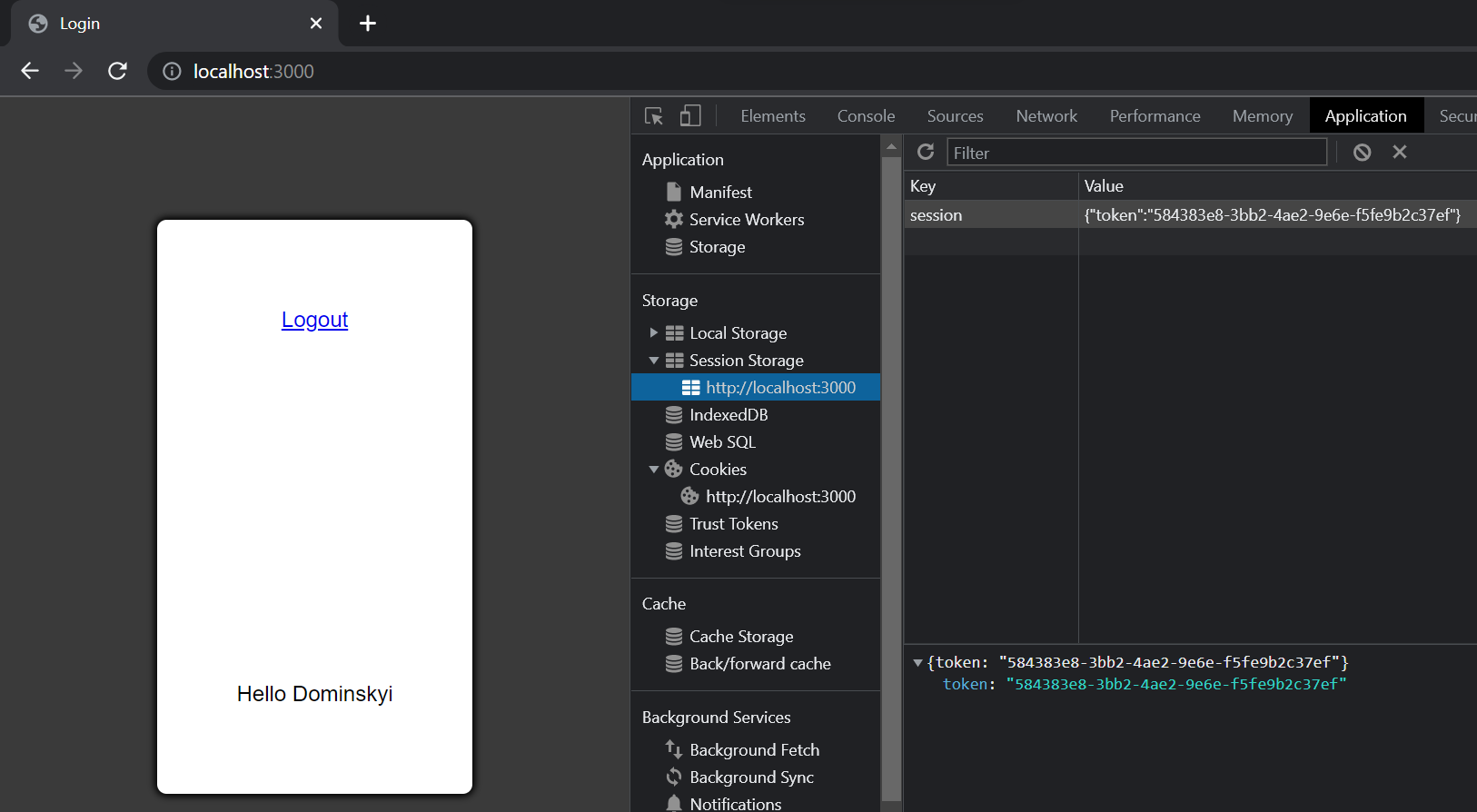
Якщо ж спробувати вийти зробити logout, то Наш токен сесії буде знищений та створено новий.

## Token\_auth

Великої к-сті змін Ми тут не побачимо. Основна суть цього застосунку – перенести зберігання даних з cookie:



При різній взаємодії сесія все-одно йде, але сервер не звертає на неї жодної уваги, тому давайте відразу перейдемо до введення правильних даних:



Після відправки запити на логін застосунок отримав токен, який не зберігає в сесії, а натомість – у браузері у Session Storage. І вже тут код браузеру відправляє дані в окремому хедері в Authorization. Таким чином Ми отримуємо більшу безпеку, оскільки куки відправляються браузером автоматично

### Jwt\_auth

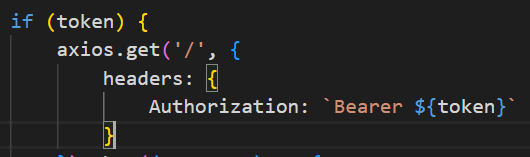
Для реалізації цього виду аутентифікації Я буду модифіковувати приклад з token\_auth.

Було зроблено такі речі:

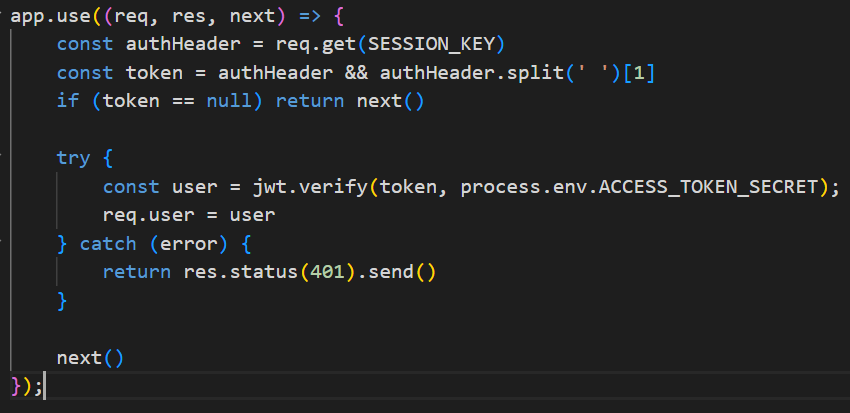
1. Імпортовано jsonwebtoken для jwt токену та dotenv для змінних середовища



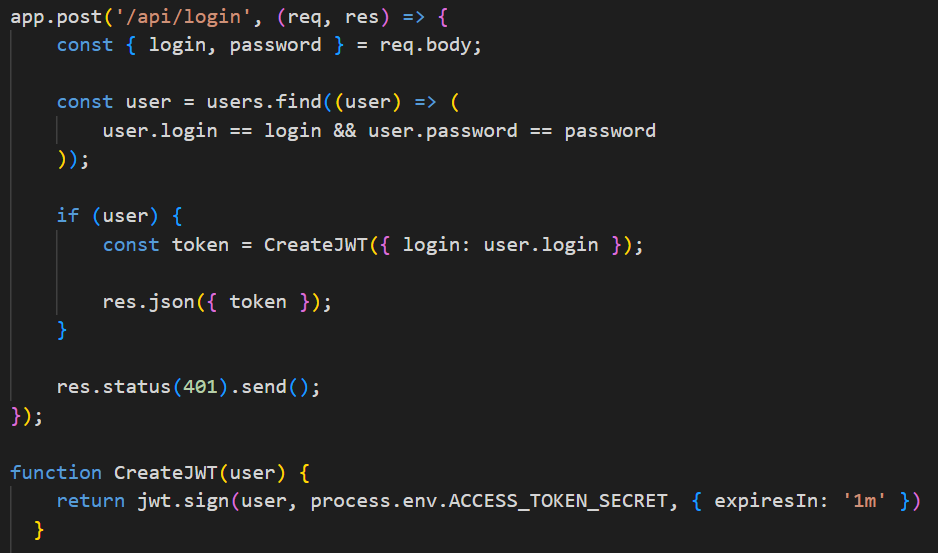
1. Видалено код для створення сесій
2. Додав до index.html приписку bearer до заголовку авторизації



1. До app.use додав перевірку на існуючий токен

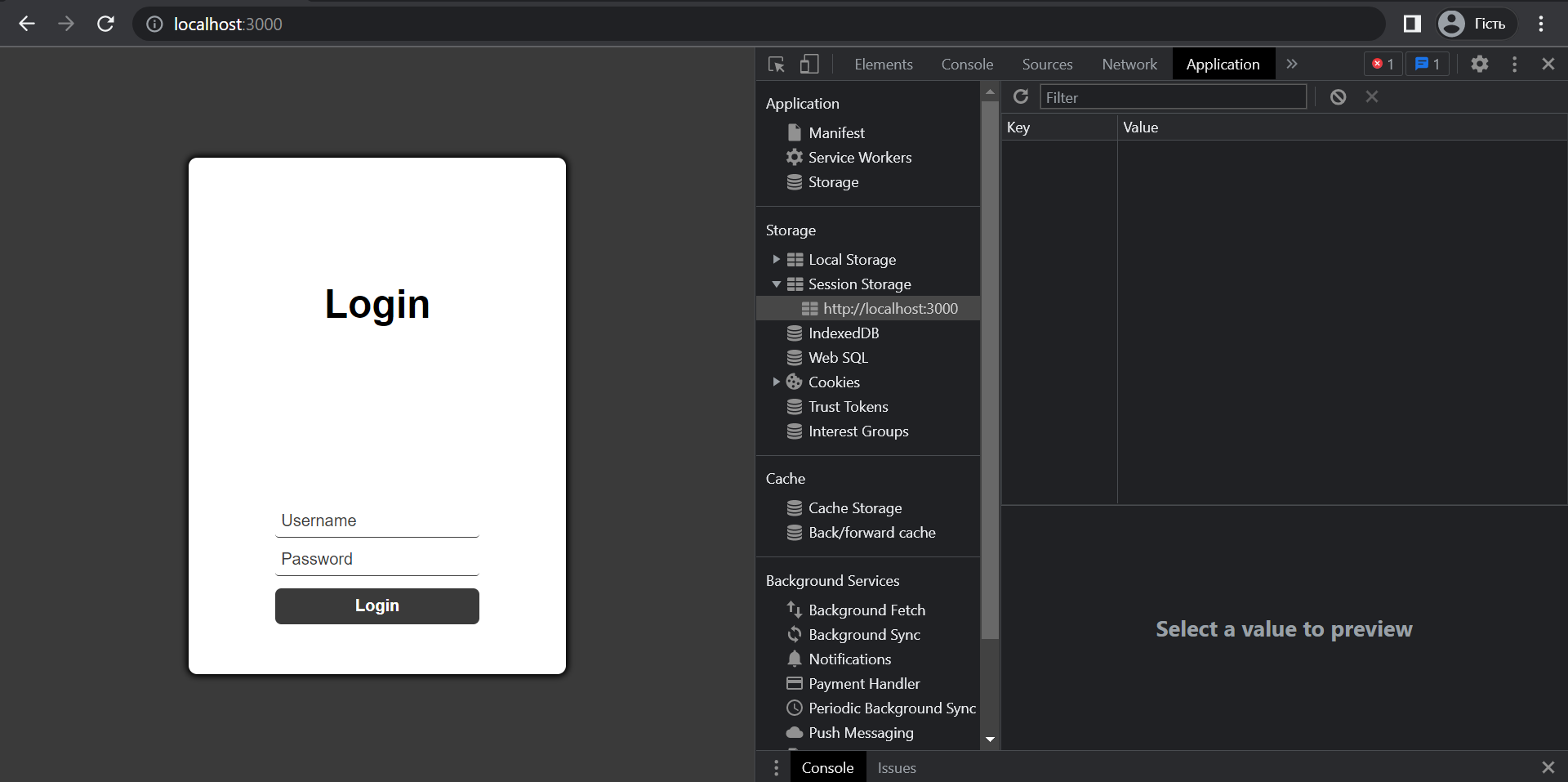


1. При спробі авторизації створюється jwt token

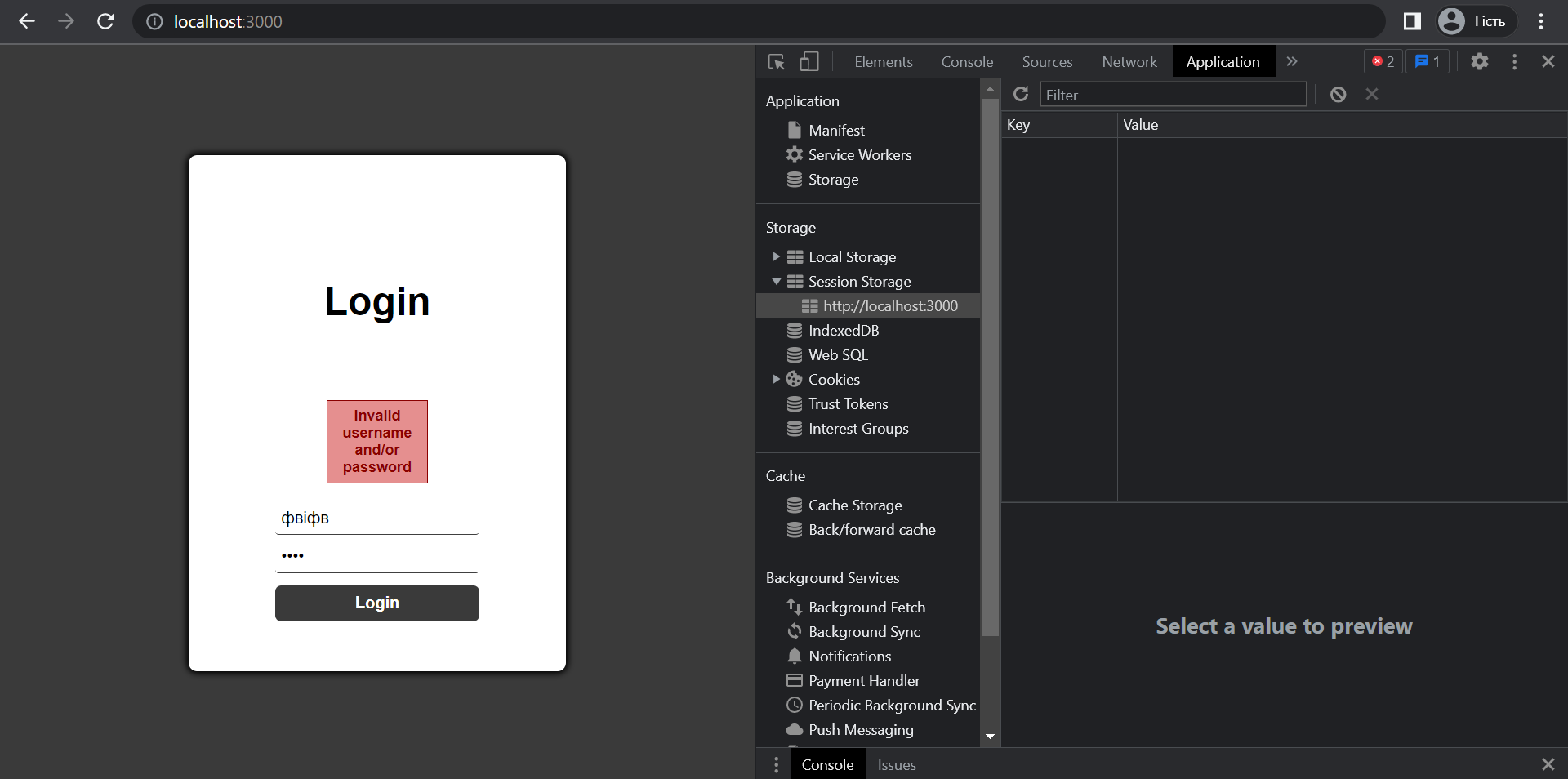


Тепер давайте подивимося, як це працює на реальному прикладі:

На самому початку у Session Storage Ми не маємо абсолютно нічого:

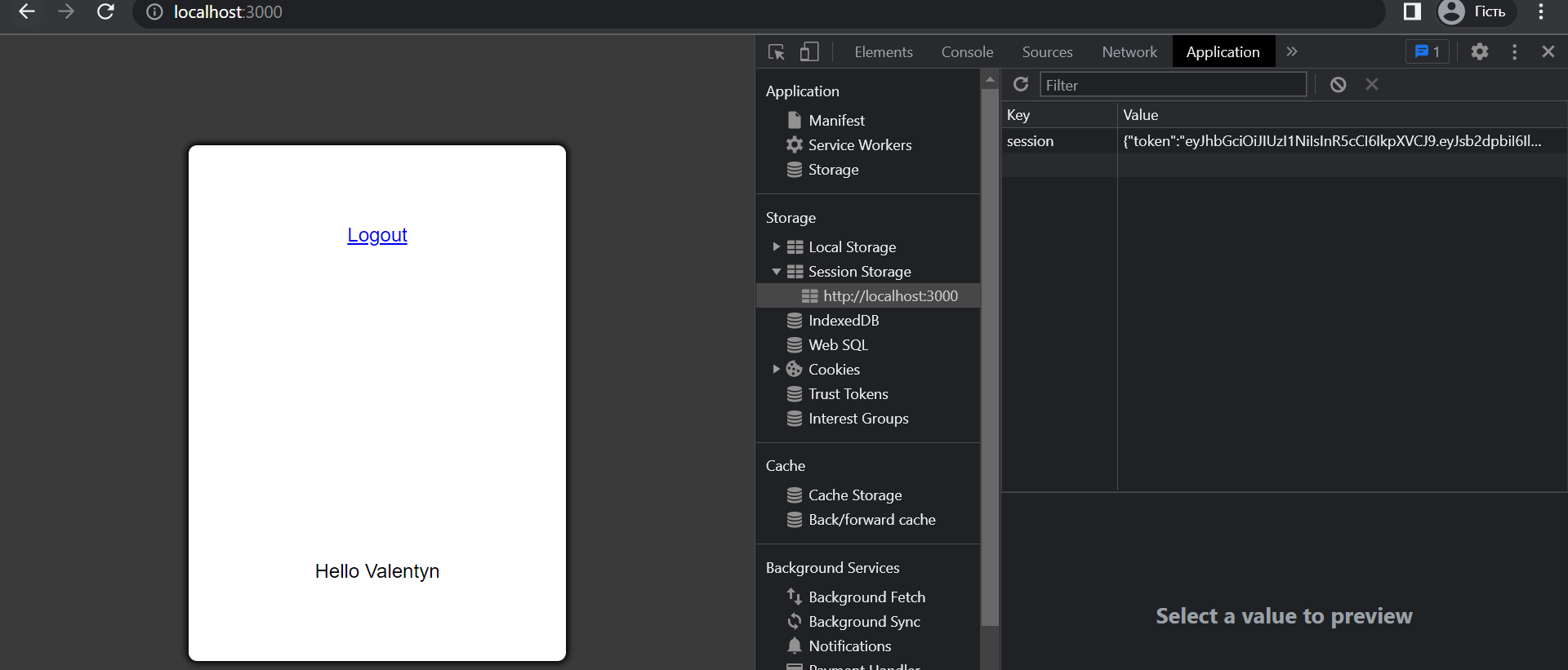


Нумо спробуємо ввести неправильні дані:



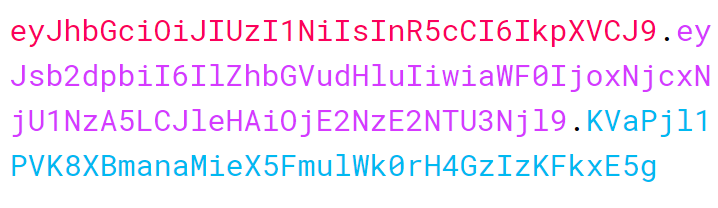
Як видно з малюнку - Нам показується лише попередження про їх невірність.

А тепер розглянемо діаметрально протилежний варіант:

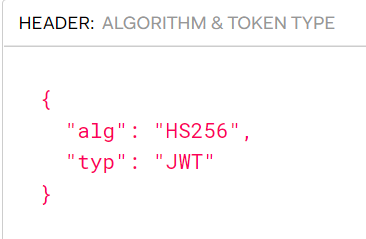


По-перше, Ми успішно зайшли до системи і перейшли до сторінки виходу з акаунту. По-друге, у закладці Session Storage згенерувався токен, який буде валідний ще одну хвилину, після чого при будь-якій дії Нас примусово перекине на початкову сторінку.

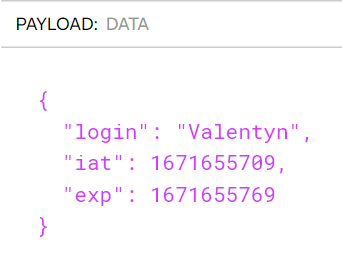
Давайте тепер розглянемо сам jwt token за допомогою сайту jwt.io:



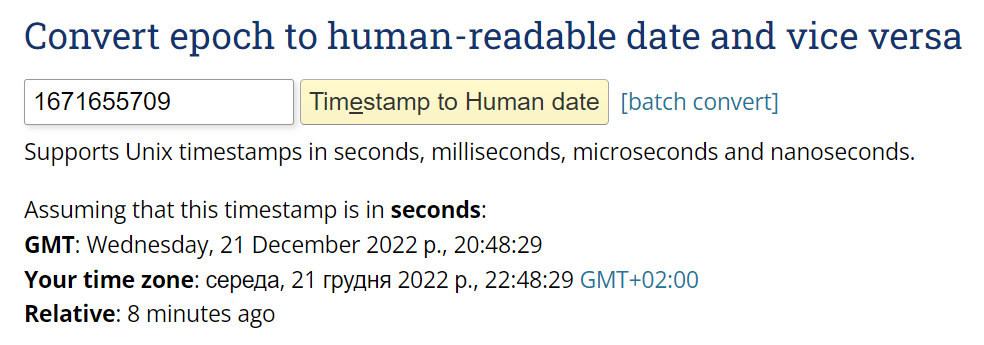
* Червона частина – це Header, який має алгоритм шифрування та тип токену:

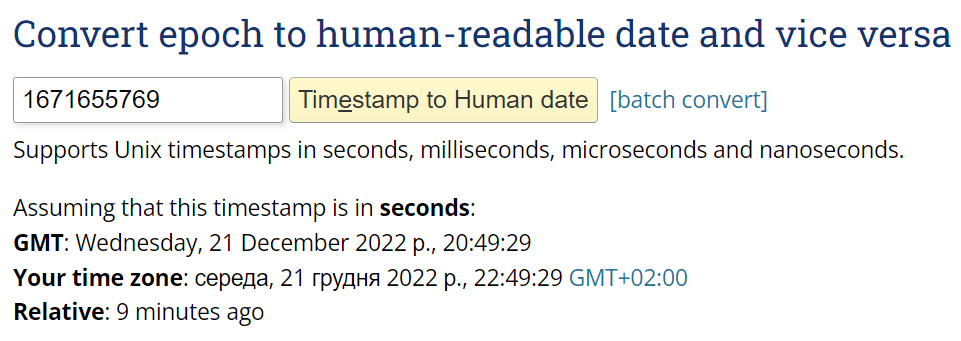


* Рожева – дані, котрі містяться в токені:



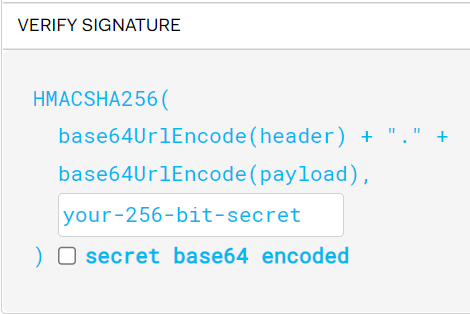
* + Тут видно дійсний логін, котрий Я і ввів, проте є й інші поля: iat та exp. Це час створення та час прострочення, які Ми можемо розшифрувати за допомогою сайту epochconverter.com:





Таким чином Ми можемо впевнитися, що Наш заданий час дійсності (1 хвилина) дійсно записується при створенні jwt токену

* Ну і в останній – синій частині – Verify Signature:



# Висновок:

Під час виконання роботи я розібрався з різними базовими видами аутентифікації, а також виконав розширене завдання щодо інтеграції JWT токену

# Посилання:

* [Проект на GitHub](https://github.com/VsIG-official/Software-Security)