

Міністерство освіти і науки України

Національний технічний університет України „КПІ імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки Кафедра інформатики і програмної інженерії

# ЗВІТ

лабораторної роботи № 2

з курсу «Основи WEB - технологій»

Тема: «Створення системи авторизації сайту за допомогою JWT токенів»

|  |  |
| --- | --- |
| Перевірив:  Викл. Альбрехт Й.О. | Виконав:  Студент ІП-15 Мєшков А.І |

Київ 2024

# Завдання

1. При реєстрації та при оновленні користувача пароль зберігати в зашифрованому вигляді;

2. Створити запит /users/login на вхід

3. Застосувати авторизацію: в запиті відправити auth Token в заголовку Authorization. Сервер отримує токен, верифікує його і авторизує користувача (або ні в іншому випадку).

Забезпечити використання ролей (user та admin) з відповідними діями.

# Хід роботи

Було створено програму, яка містить файли серверу, підʼєднання бази даних, моделі таблиць бд.

**server.js**

require('dotenv').config();

const express = require('express');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const connectDB = require('./utils/db');

const User = require('./models/User');

const app = express();

app.use(express.json());

connectDB();

const JWT\_SECRET = process.env.JWT\_SECRET;

const authMiddleware = (roles = []) => (req, res, next) => {

const token = req.headers['authorization'];

*if* (!token) *return* res.status(401).json({ message: 'Access denied' });

*try* {

const decoded = jwt.verify(token, JWT\_SECRET);

req.user = decoded;

*if* (roles.length && !roles.some(role => req.user.roles.includes(role))) {

*return* res.status(403).json({ message: 'Forbidden' });

}

next();

} *catch* (error) {

res.status(401).json({ message: 'Invalid token' });

}

};

app.post('/users/register', async (req, res) => {

*try* {

const { username, password} = req.body;

const existingUser = *await* User.findOne({ username });

*if* (existingUser) *return* res.status(400).json({ message: 'User already exists' });

const hashedPassword = *await* bcrypt.hash(password, 10);

const user = new User({ username, password: hashedPassword});

*await* user.save();

res.status(201).json({ message: 'User registered successfully' });

} *catch* (error) {

res.status(500).json({ message: 'Server error' });

}

});

app.post('/users/login', async (req, res) => {

*try* {

const { username, password } = req.body;

const user = *await* User.findOne({ username });

*if* (!user) *return* res.status(400).json({ message: 'Invalid credentials' });

const isPasswordValid = *await* bcrypt.compare(password, user.password);

*if* (!isPasswordValid) *return* res.status(400).json({ message: 'Invalid credentials' });

const token = jwt.sign({ userId: user.\_id, roles: user.roles[0] }, JWT\_SECRET, { expiresIn: '1h' });

res.json({ token });

} *catch* (error) {

res.status(500).json({ message: 'Server error' });

}

});

app.get('/auth/admin', authMiddleware(['ADMIN']), (req, res) => {

res.json({ message: 'Welcome, Admin' });

});

app.get('/auth/user', authMiddleware(['USER', 'ADMIN']), (req, res) => {

res.json({ message: `Welcome, User with roles: ${req.user.roles}` });

});

app.get('/auth/users', authMiddleware(['ADMIN']), async (req, res) => {

*try* {

const users = *await* User.find({});

res.json(users);

} *catch* (error) {

res.status(500).json({ message: 'Server error' });

}

});

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

**db.js**

const mongoose = require('mongoose');

require('dotenv').config();

const dbUrl = process.env.DB\_URL || '';

const connectDB = async () => {

*try* {

const data = *await* mongoose.connect(dbUrl);

console.log(`Database connected with ${data.connection.host}`);

} *catch* (error) {

console.log(error.message);

setTimeout(connectDB, 5000);

}

};

module.exports = connectDB;

**User.js**

const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({

username: { type: String, required: true, unique: true },

password: { type: String, required: true },

roles: { type: [String], default: ["USER"]}

});

module.exports = mongoose.model('User', UserSchema);

# Отримані результати

На рис 2.1 можна побачити запит реєстрації користувача.

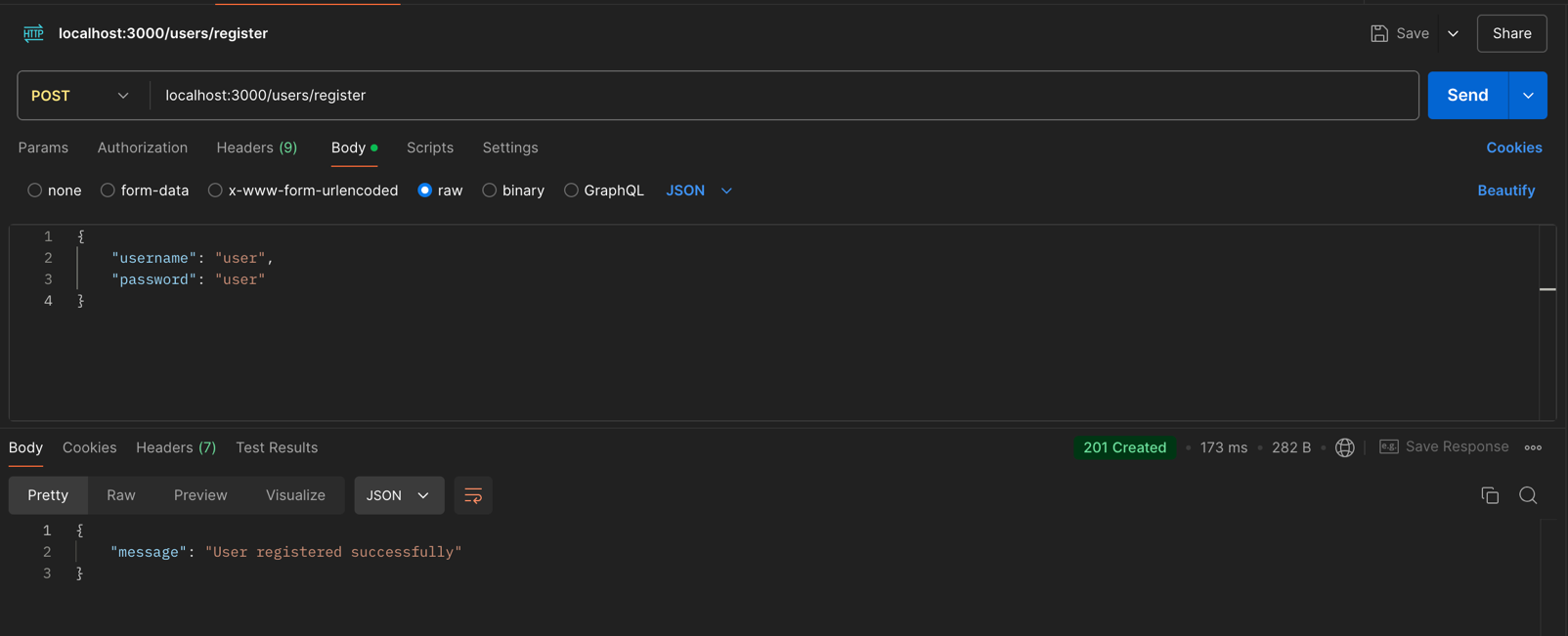


Рис. 2.1. Запит реєстрації користувача

На рис 2.2 показано запит авторизації.

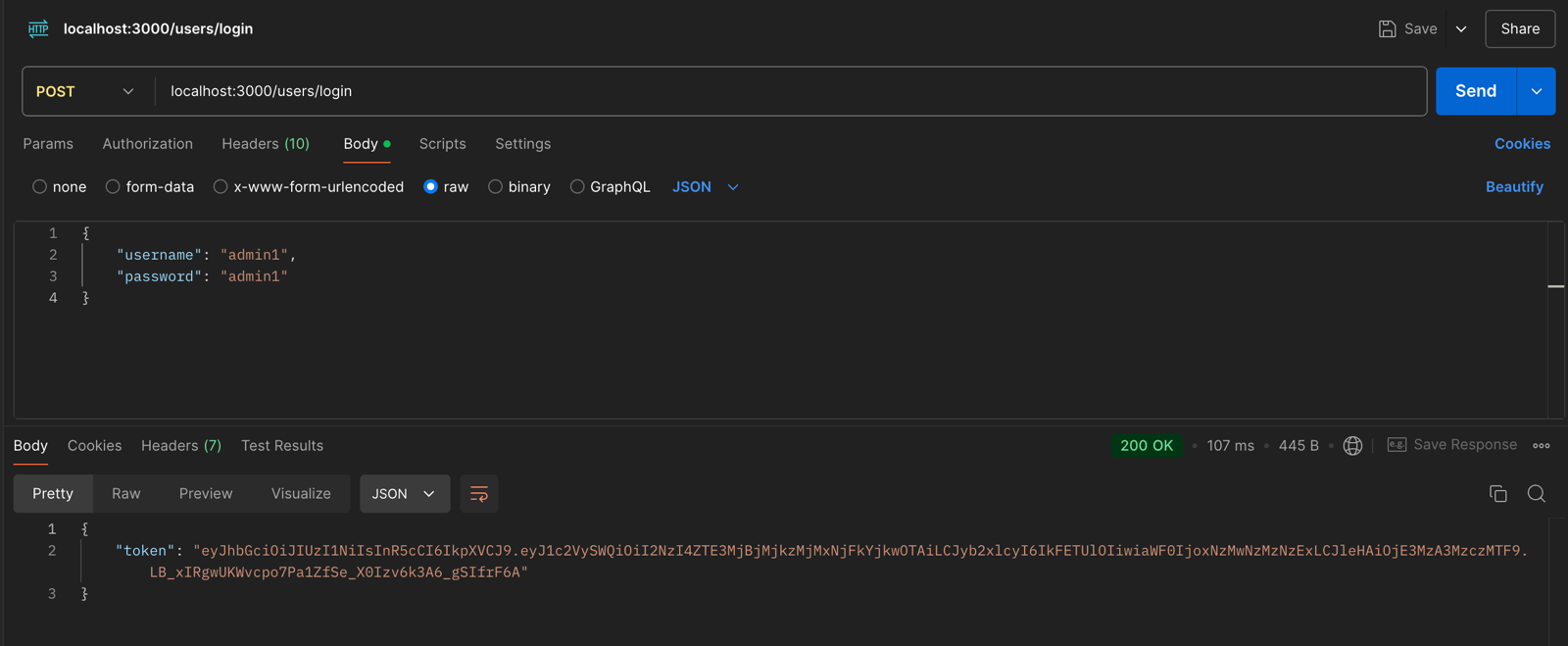


Рис. 2.2. Запит авторизації

На рис 2.3 показано запит з застосуванням авторизаційного токену.

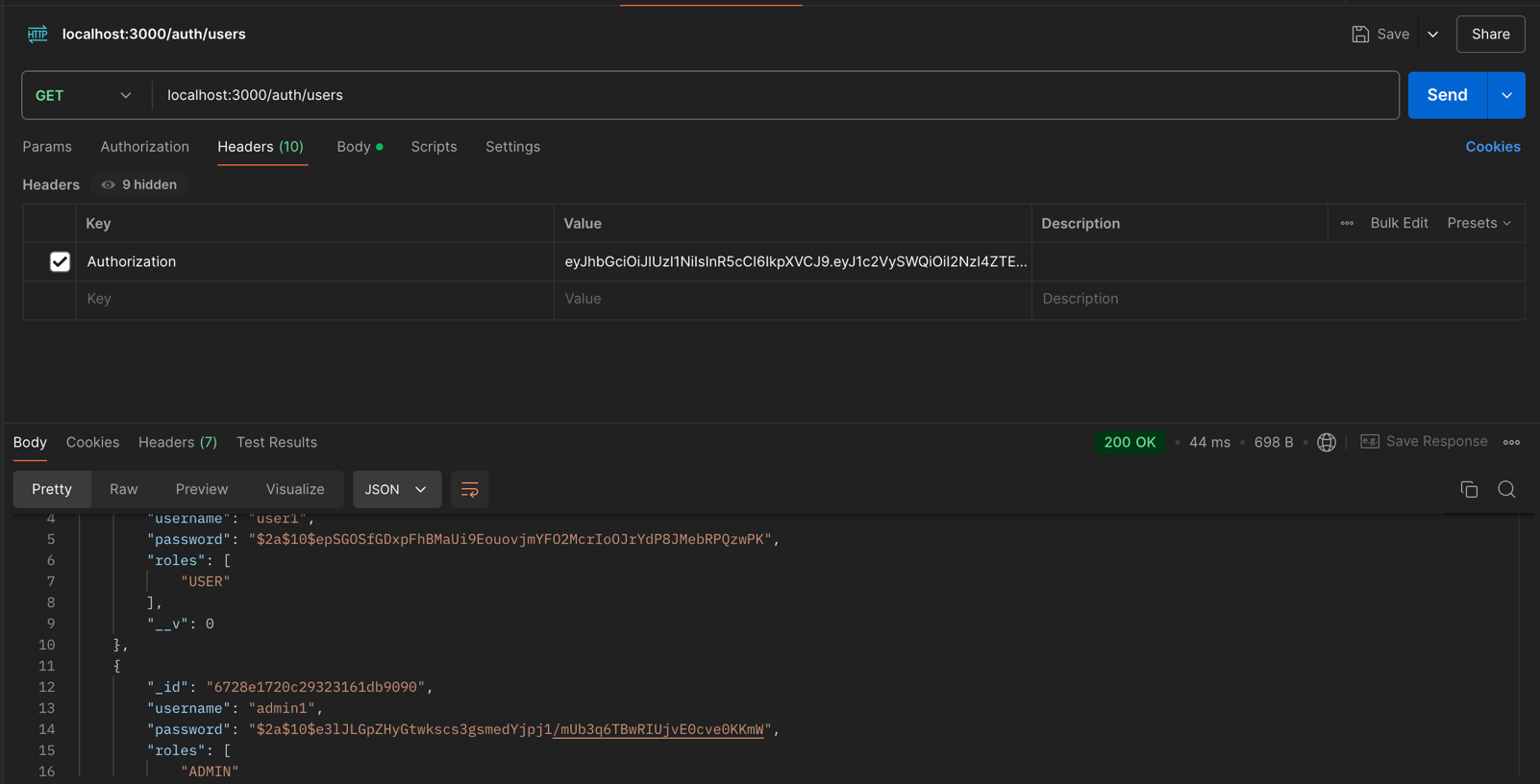


Рис. 2.3. Запит

На рис 2.4-2.5 показано запит з відсутністю прав.

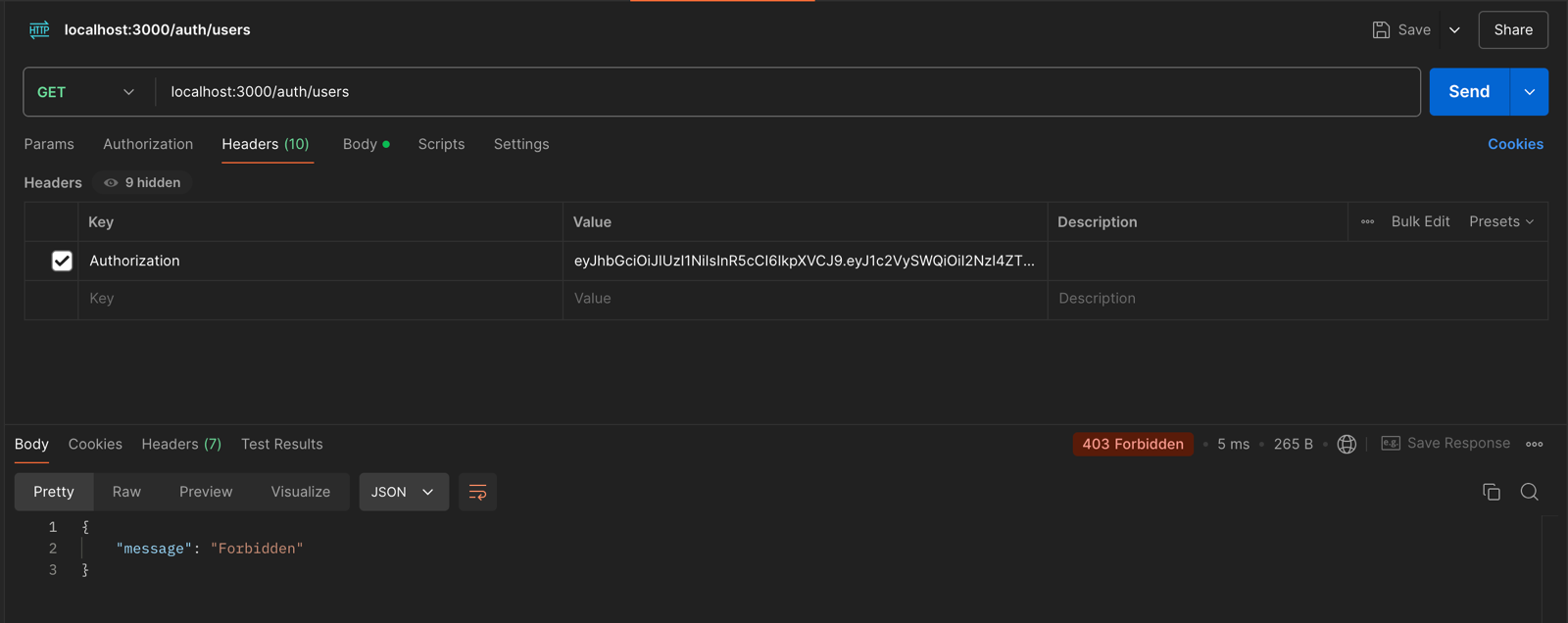


Рис. 2.4. Результат запиту

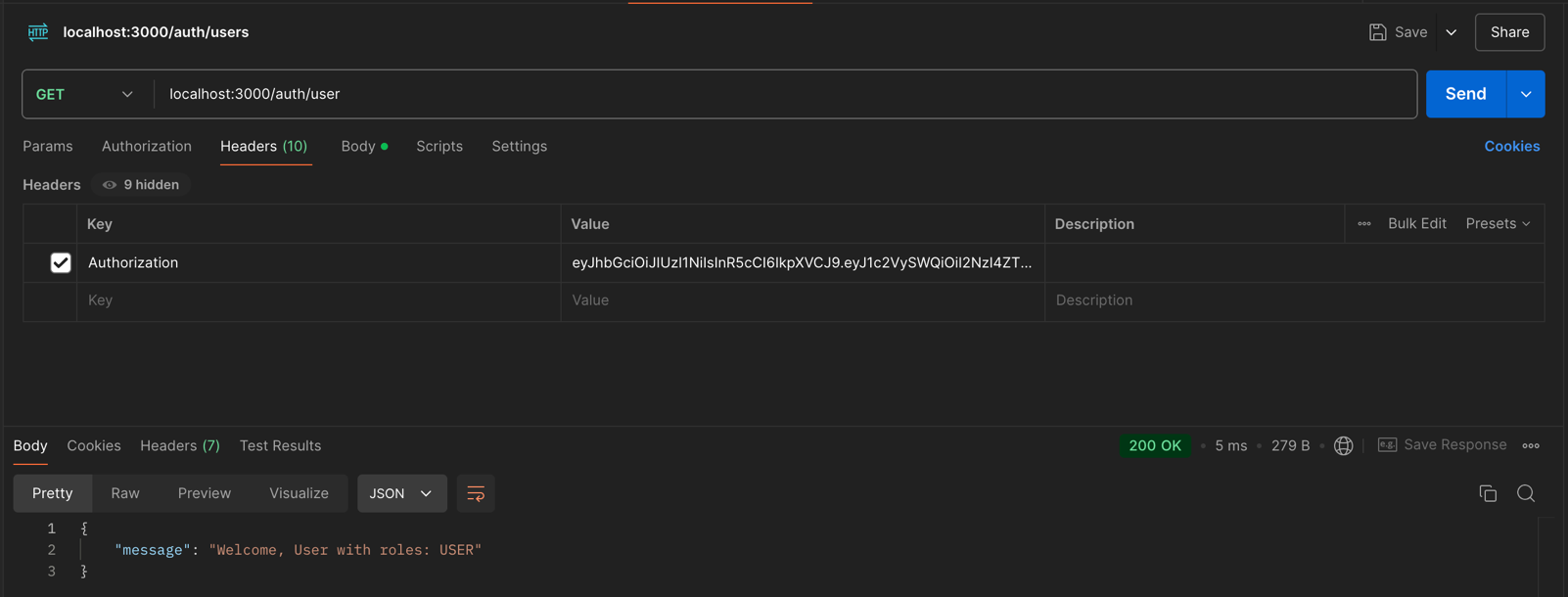


Рис. 2.5. Результат запиту

На рис 2.6 показано таблицю користувачів.

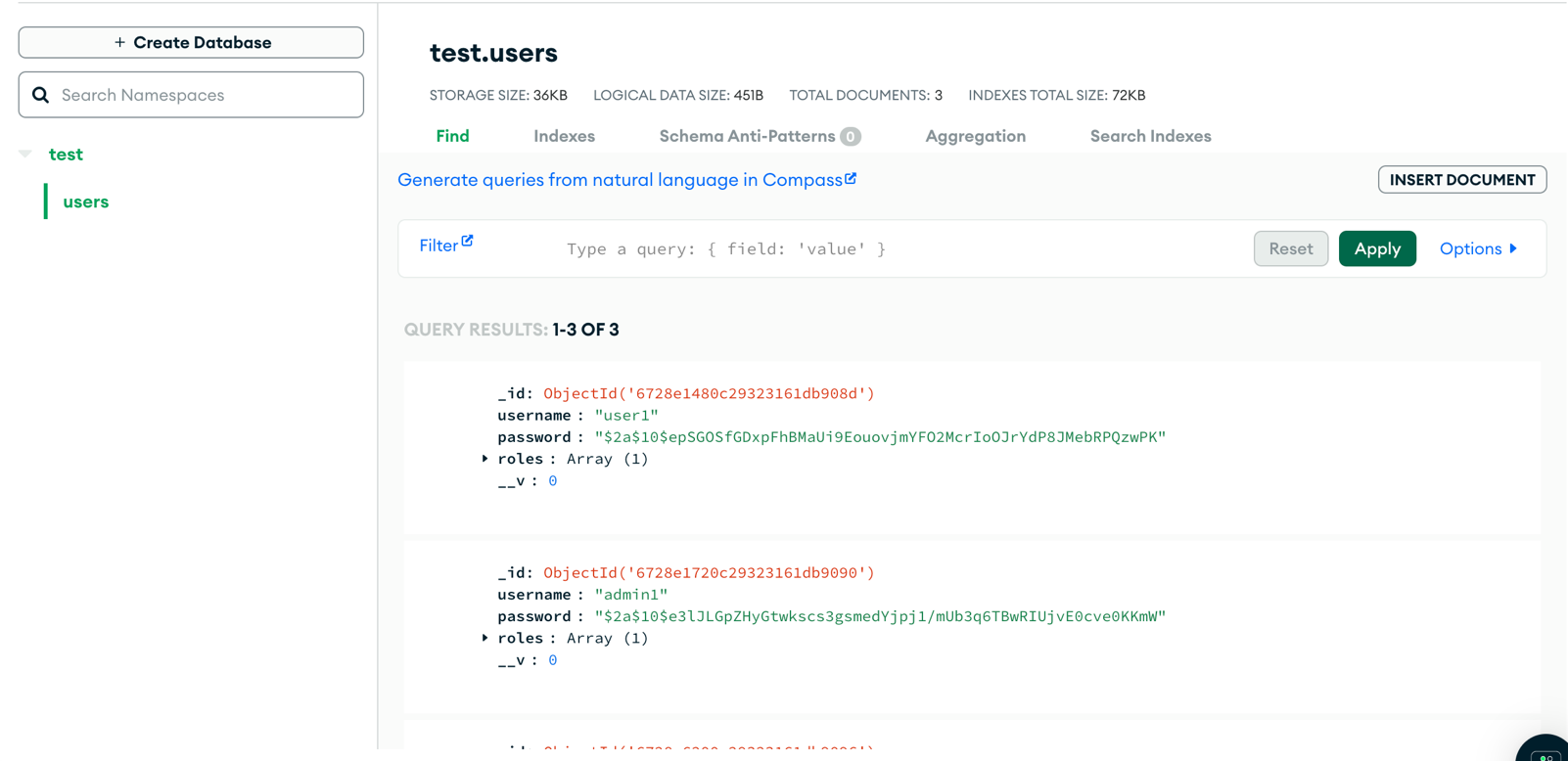


Рис. 2.6. Таблиця користувачів

# ВИСНОВОК

Під час виконання даної лабораторної роботи було розроблено базову систему авторизації та аутентифікації для вебсайту з використанням JSON Web Tokens (JWT) і ролей. Основні етапи роботи включали:

1. Підключення до бази даних MongoDB: було створено та налаштовано підключення до MongoDB за допомогою Mongoose, що забезпечило можливість зберігання та доступу до інформації про користувачів.

2. Створення моделей User та Role: було налаштовано структуру даних для користувачів із збереженням паролів у зашифрованому вигляді та використанням ролей у вигляді масиву рядків. Це дозволило зручно керувати привілеями користувачів.

3. Реалізація маршруту реєстрації та входу: було створено маршрути для реєстрації та входу користувачів, які включають захист пароля (хешування) та видачу JWT токена для аутентифікації.

4. Впровадження механізму авторизації за допомогою ролей: було налаштовано middleware для перевірки ролей користувачів, що дозволяє обмежити доступ до певних маршрутів, зокрема, лише для адміністратора.

5. Забезпечення безпеки: паролі зберігаються у зашифрованому вигляді, а для обміну даними між клієнтом і сервером використовується JWT, що дозволяє безпечно ідентифікувати користувачів.

У результаті було створено базову систему аутентифікації та авторизації, яка може використовуватися як основа для розширення функціональності і реалізації складніших механізмів керування доступом в більш масштабних додатках.