Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем і технологій

Лабораторна робота №2

з дисципліни «Основи WEB-технологій»

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

Виконав:
студент групи ІА-11
Воробей Антон Олегович

Перевірив:
Альбрехт Й. О.

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 authToken в заголовку Authorization. Сервер отримує токен, верифікує його і авторизує користувача (або ні в іншому випадку).

Посилання на вихідний код проекта: https://github.com/tonujet/WEB-basics/tree/main/Lab2

Хід роботи

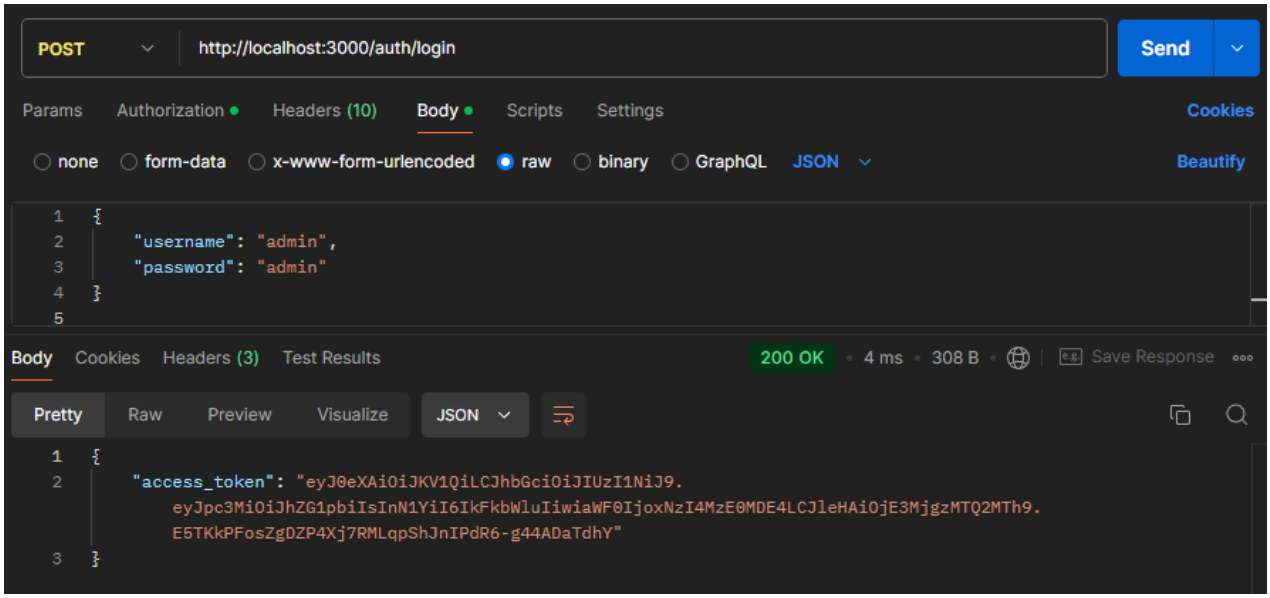
Для виконання даної лабораторної роботи я скористаюся мовою Rust

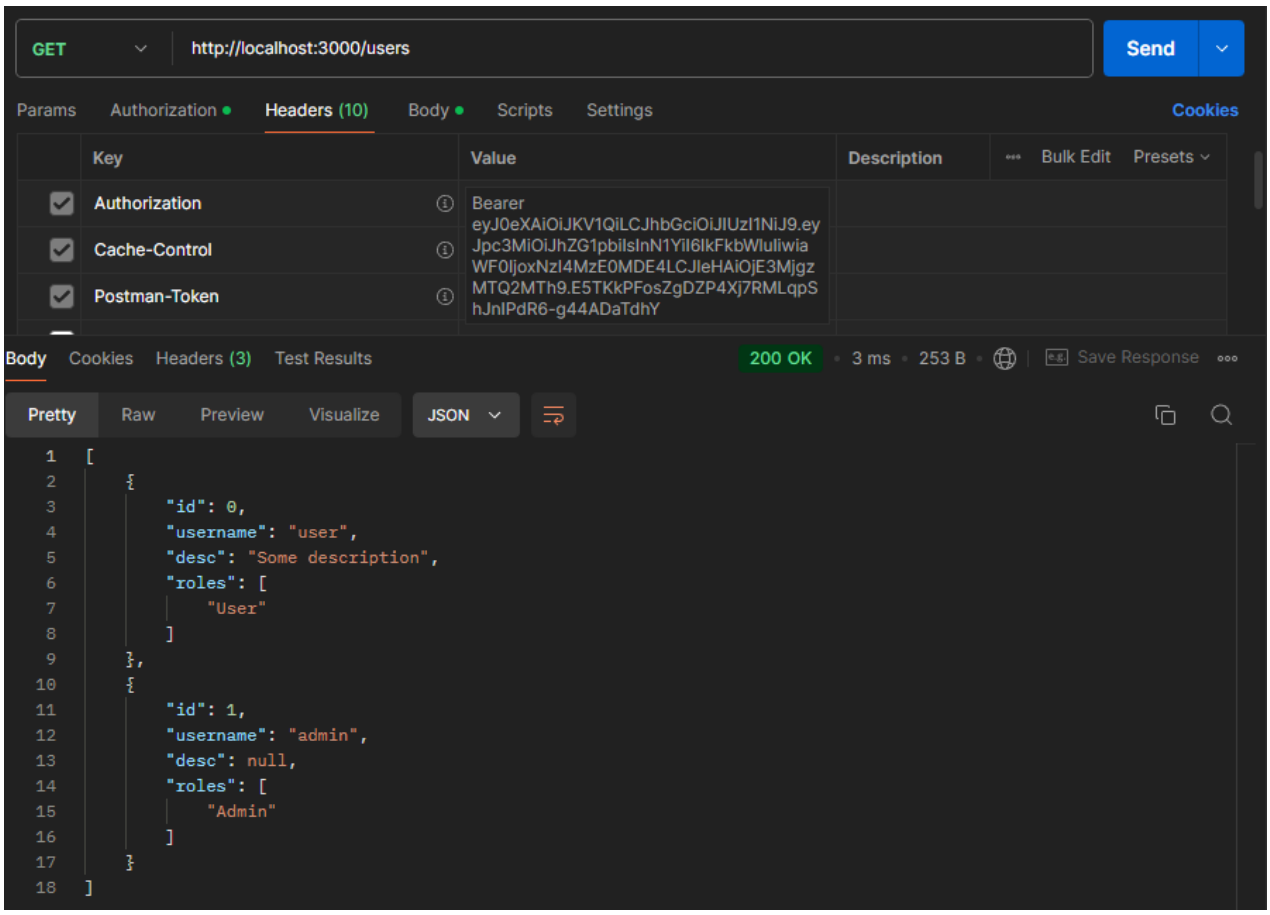
Для утворення JWT токена та його валідації використовуються такий код

```rust
#[derive(Debug, Serialize, Deserialize)]
pub struct Claims {
    pub iss: String,
    pub sub: String,
    pub iat: i64,
    pub exp: i64,
}

pub fn make_jwt(
    issuer: String,
    subject: String,
    duration: Duration,
    secret: &[u8],
) -> AuthResult<String> {
    let mut header = Header::new(Algorithm::HS256);
    header.typ = Some("JWT".to_string());

    let now = Local::now();
    let iat = now.timestamp();
    let exp = (now + duration).timestamp();
    let claims = Claims {
        iss: issuer,
        sub: subject,
        iat,
        exp,
    };

    Ok(encode(&header, &claims, &EncodingKey::from_secret(secret))?)
}

pub fn validate_jwt(token: &str, secret: &[u8]) -> AuthResult<Claims> {
    let validation = Validation::default();
    let token = decode::<Claims>(token, &DecodingKey::from_secret(secret),
```

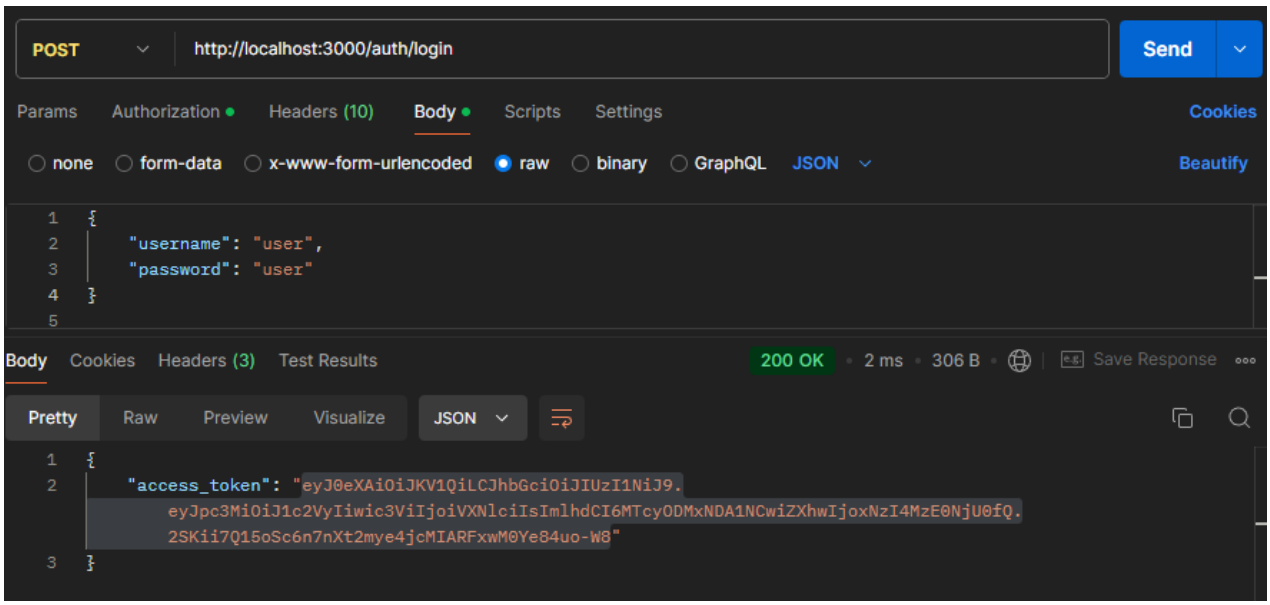
```
&validation)?;
    Ok(token.claims)
}
```

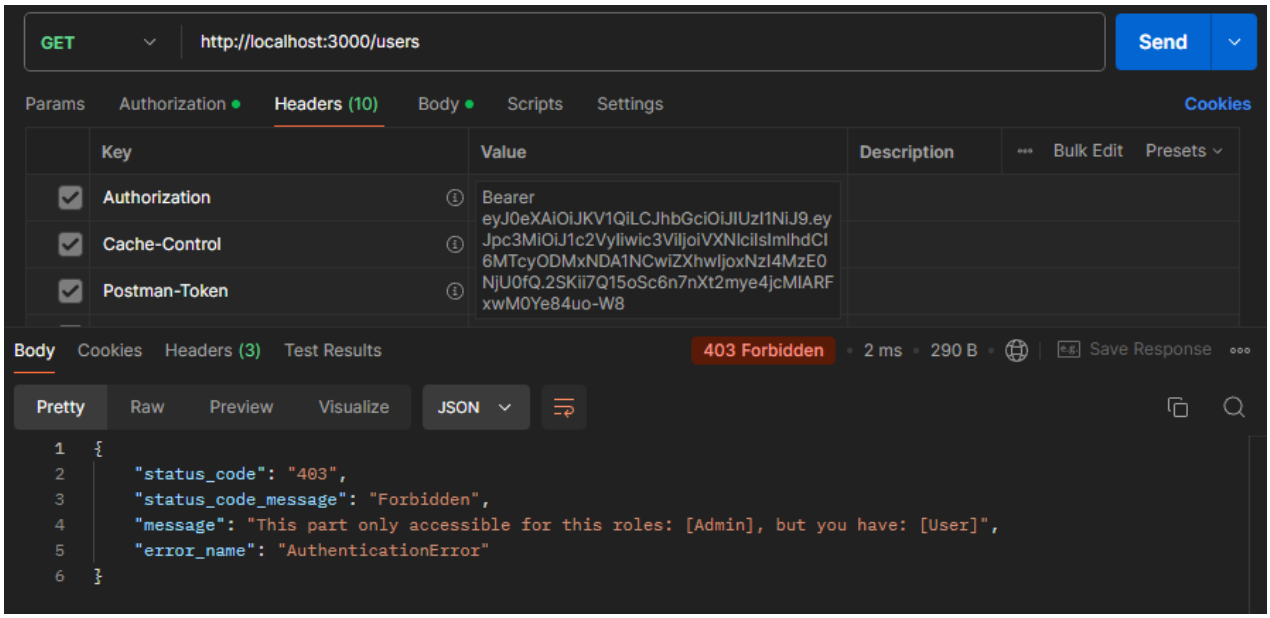
Отримаємо JWT для адміна та спробуємо прочитати інформацію для усіх користувачів

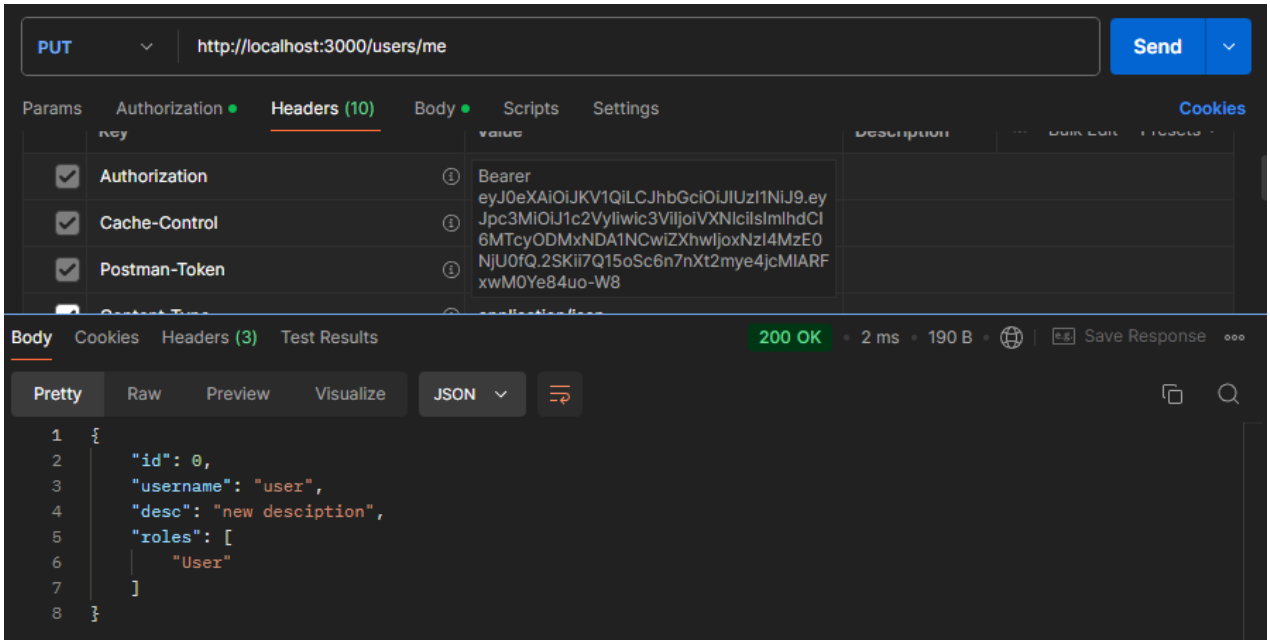


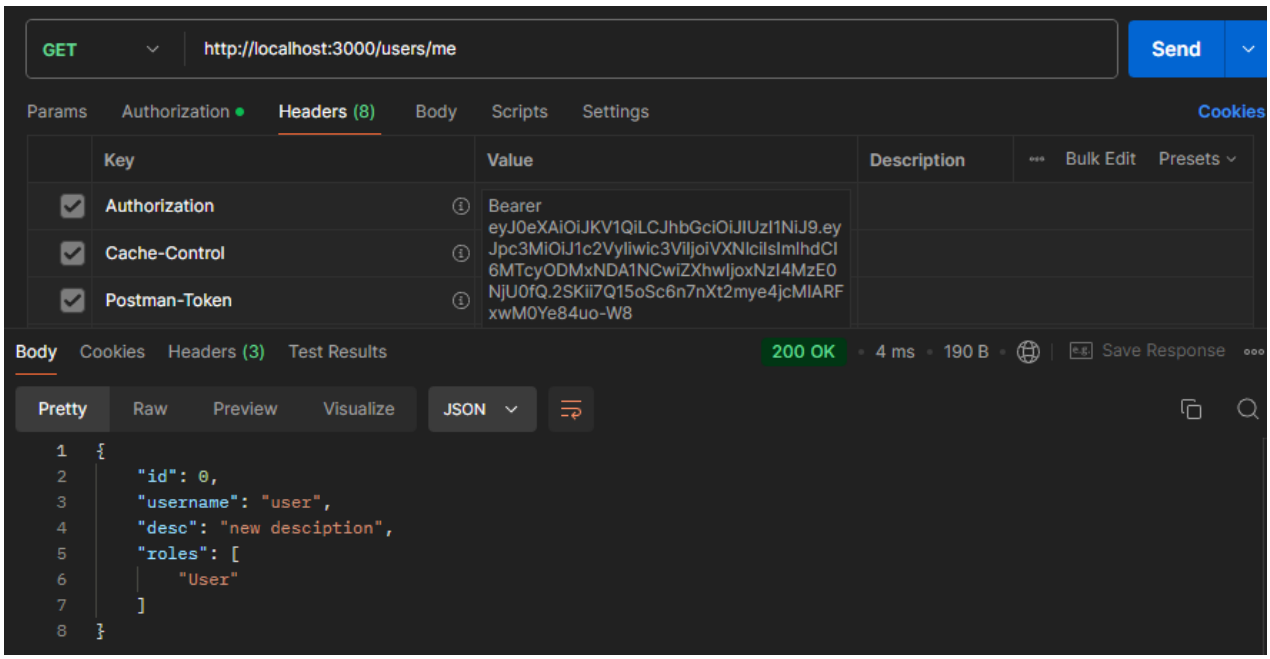


Отримаємо JWT для користувача та спробуємо прочитати інформацію про всіх користувачів(отримаємо помилку) та спробуємо змінити власний опис


POST
http://localhost:3000/auth/login
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "user",
"password": "user"
}
Body
Cookies
Headers (3)
Test Results
200 OK
2 ms
306 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJpc3MiOiJ1c2VyIiwic3ViIjoiVXNlciIsImlhdCI6MTcyODMxNDA1NCwiZXhwIjoxNzI4MzE0NjU0fQ.
2SKii7Q15oSc6n7nXt2mye4jcMIARFxwM0Ye84uo-W8"
}


GET
http://localhost:3000/users
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
Key
Value
Description
Bulk Edit
Presets
Authorization
Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
Jpc3MiOiJ1c2VyIiwic3ViIjoiVXNlciIsImlhdCI
6MTcyODMxNDA1NCwiZXhwIjoxNzI4MzE0
NjU0fQ.2SKii7Q15oSc6n7nXt2mye4jcMIARF
xwM0Ye84uo-W8
Cache-Control
Postman-Token
Body
Cookies
Headers (3)
Test Results
403 Forbidden
2 ms
290 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status_code": "403",
"status_code_message": "Forbidden",
"message": "This part only accessible for this roles: [Admin], but you have: [User]",
"error_name": "AuthenticationError"
}


PUT
http://localhost:3000/users/me
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
Authorization
Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
Jpc3MiOiJ1c2VyIiwic3ViIjoiVXNlciIsImlhdCI
6MTcyODMxNDA1NCwiZXhwIjoxNzI4MzE0
NjU0fQ.2SKii7Q15oSc6n7nXt2mye4jcMIARF
xwM0Ye84uo-W8
Cache-Control
Postman-Token
Body
Cookies
Headers (3)
Test Results
200 OK
2 ms
190 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 0,
"username": "user",
"desc": "new desciption",
"roles": [
"User"
]
}



Висновок: На цій лабораторні роботі я навчився використовувати JWT та імплементував свою власну реалізацію застосування цієї технології