МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

**ОТЧЕТ**

По лабораторной работе №3

**Авторизация**

Выполнили студенты

группы 6301-010302D

Астафьева А.В.

и 6304-010302D

Серебряков В.С.

САМАРА 2025

**ОПИСАНИЕ АУТЕНТИФИКАЦИИ**

Эндпоинты:

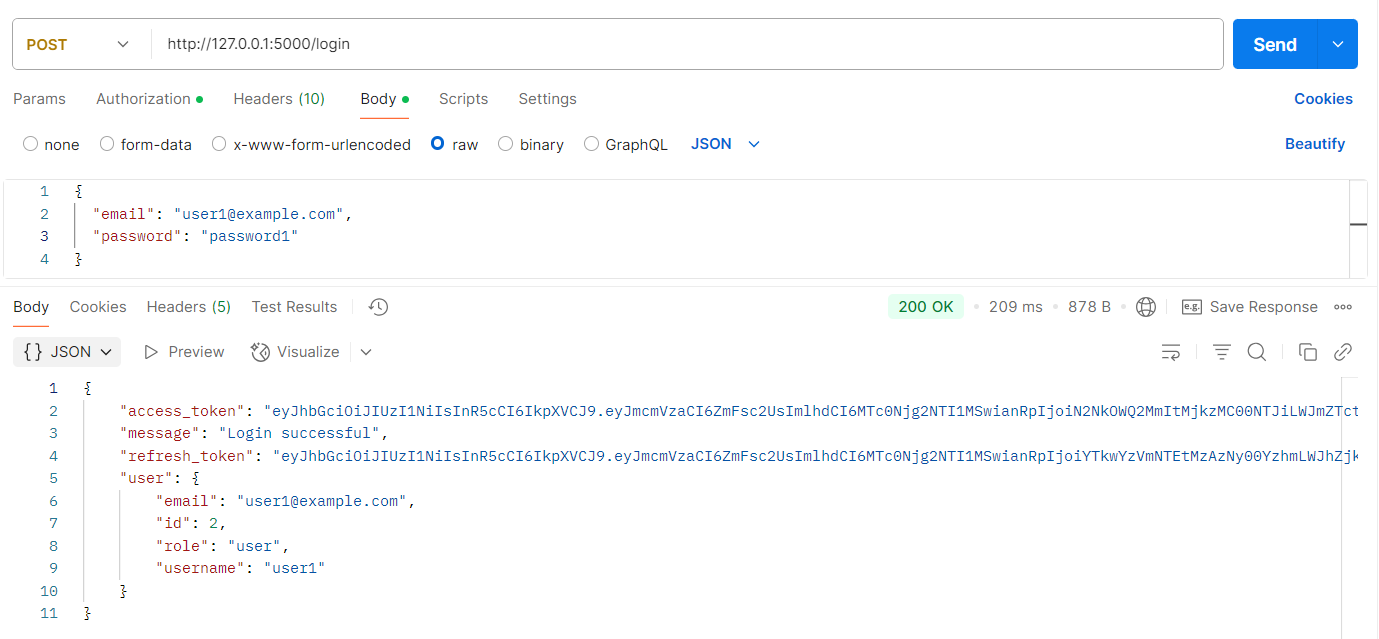
* Вход

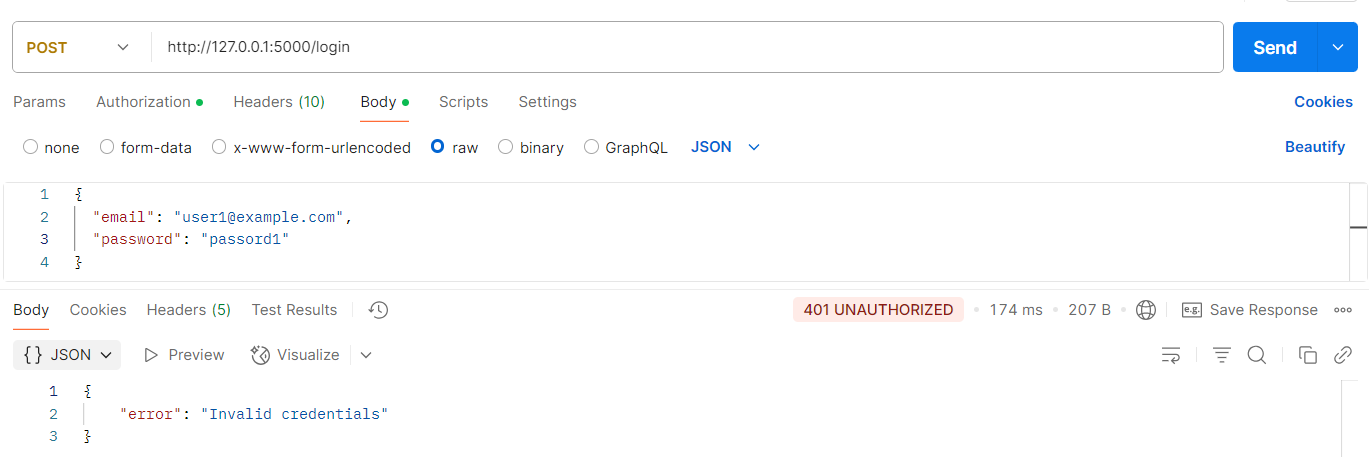
1. @bp.route('/login', methods**=**['POST'])
2. **def** login():
3. data **=** request.get\_json()
4. **if** **not** data **or** 'email' **not** **in** data **or** 'password' **not** **in** data:
5. **return** jsonify({"error": "Email and password required"}), 400
7. user **=** Users.query.filter\_by(email**=**data['email']).first()
8. **if** **not** user **or** **not** user.check\_password(data['password']):
9. **return** jsonify({"error": "Invalid credentials"}), 401
11. # Создаем токены с дополнительными claims (включая роль)
12. additional\_claims **=** {"role": user.role}
13. access\_token **=** create\_access\_token(
14. identity**=**str(user.id),
15. additional\_claims**=**additional\_claims
16. )
17. refresh\_token **=** create\_refresh\_token(
18. identity**=**str(user.id),
19. additional\_claims**=**additional\_claims
20. )
22. **return** jsonify({
23. "message": "Login successful",
24. "access\_token": access\_token,
25. "refresh\_token": refresh\_token,
26. "user": {
27. "id": user.id,
28. "username": user.username,
29. "email": user.email,
30. "role": user.role  # Добавляем роль в ответ
31. }
32. }), 200

* Выход
* @bp.route('/logout', methods**=**['POST'])
* @jwt\_required()
* **def** logout():
* jti **=** get\_jwt()["jti"]
* now **=** datetime.now(timezone.utc)
* db.session.add(TokenBlocklist(jti**=**jti, created\_at**=**now))
* db.session.commit()
* **return** jsonify({"message": "Successfully logged out"}), 200
* Регистрация
* @bp.route('/register', methods**=**['POST'])
* **def** register():
* **import** re  # Импортируем модуль для работы с регулярными выражениями
* data **=** request.get\_json()
* required\_fields **=** ['email', 'password', 'username']
* **if** **not** data **or** **not** all(key **in** data **for** key **in** required\_fields):
* **return** jsonify({"error": "Missing required fields"}), 400
* # Валидация email
* email\_regex **=** r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
* **if** **not** re.match(email\_regex, data['email']):
* **return** jsonify({"error": "Invalid email format"}), 400
* **if** Users.query.filter\_by(email**=**data['email']).first():
* **return** jsonify({"error": "Email already registered"}), 400
* **if** Users.query.filter\_by(username**=**data['username']).first():
* **return** jsonify({"error": "Username already taken"}), 400
* **try**:
* # Создаем пользователя с ролью по умолчанию 'user'
* user **=** Users(
* email**=**data['email'],
* username**=**data['username'],
* role**=**data.get('role', 'user')  # Можно задать роль при регистрации, если нужно
* )
* user.set\_password(data['password'])
* db.session.add(user)
* db.session.commit()
* # Создаем токены с дополнительными claims (включая роль)
* additional\_claims **=** {"role": user.role}
* access\_token **=** create\_access\_token(
* identity**=**str(user.id),
* additional\_claims**=**additional\_claims
* )
* refresh\_token **=** create\_refresh\_token(
* identity**=**str(user.id),
* additional\_claims**=**additional\_claims
* )
* **return** jsonify({
* "message": "User registered successfully",
* "access\_token": access\_token,
* "refresh\_token": refresh\_token,
* "user": {
* "id": user.id,
* "username": user.username,
* "email": user.email,
* "role": user.role  # Добавляем роль в ответ
* }
* }), 201
* **except** Exception as e:
* db.session.rollback()
* **return** jsonify({"error": str(e)}), 500
* Верификация токена
* @bp.route('/verify', methods**=**['GET'])
* @jwt\_required()
* **def** verify\_token():
* **try**:
* # Получаем ID из токена (теперь будет работать корректно)
* user\_id **=** get\_jwt\_identity()
* user **=** Users.query.get(user\_id)
* **if** **not** user:
* **return** jsonify({"error": "User not found"}), 404
* **return** jsonify({
* "message": "Token is valid",
* "user": {
* "id": user.id,
* "username": user.username,
* "email": user.email
* }
* }), 200
* **except** Exception as e:
* **return** jsonify({"error": "Token verification failed", "details": str(e)}), 401
* Смена пароля/имени/почты
* @bp.route('/update-account', methods**=**['POST'])
* @jwt\_required()
* **def** update\_account():
* current\_user\_id **=** get\_jwt\_identity()
* data **=** request.get\_json()
* # Базовые проверки
* **if** **not** data:
* **return** jsonify({"error": "No data provided"}), 400
* user **=** Users.query.get(current\_user\_id)
* **if** **not** user:
* **return** jsonify({"error": "User not found"}), 404
* # Изменение пароля (требуется старый пароль)
* **if** 'new\_password' **in** data:
* **if** 'old\_password' **not** **in** data:
* **return** jsonify({"error": "Old password is required to set new password"}), 400
* **if** **not** user.check\_password(data['old\_password']):
* **return** jsonify({"error": "Invalid old password"}), 401
* **if** len(data['new\_password']) < 8:
* **return** jsonify({"error": "New password must be at least 8 characters"}), 400
* user.set\_password(data['new\_password'])
* # Изменение email
* **if** 'email' **in** data:
* new\_email **=** data['email'].strip()
* **if** **not** new\_email:
* **return** jsonify({"error": "Email cannot be empty"}), 400
* **if** new\_email !**=** user.email **and** Users.query.filter\_by(email**=**new\_email).first():
* **return** jsonify({"error": "Email already in use"}), 400
* user.email **=** new\_email
* # Изменение username
* **if** 'username' **in** data:
* new\_username **=** data['username'].strip()
* **if** **not** new\_username:
* **return** jsonify({"error": "Username cannot be empty"}), 400
* **if** new\_username !**=** user.username **and** Users.query.filter\_by(username**=**new\_username).first():
* **return** jsonify({"error": "Username already taken"}), 400
* user.username **=** new\_username
* # Если нет изменений
* **if** **not** any(field **in** data **for** field **in** ['new\_password', 'email', 'username']):
* **return** jsonify({"error": "No fields to update provided"}), 400
* **try**:
* db.session.commit()
* **return** jsonify({
* "message": "Account updated successfully",
* "user": {
* "id": user.id,
* "username": user.username,
* "email": user.email
* }
* }), 200
* **except** Exception as e:
* db.session.rollback()
* **return** jsonify({"error": str(e)}), 500

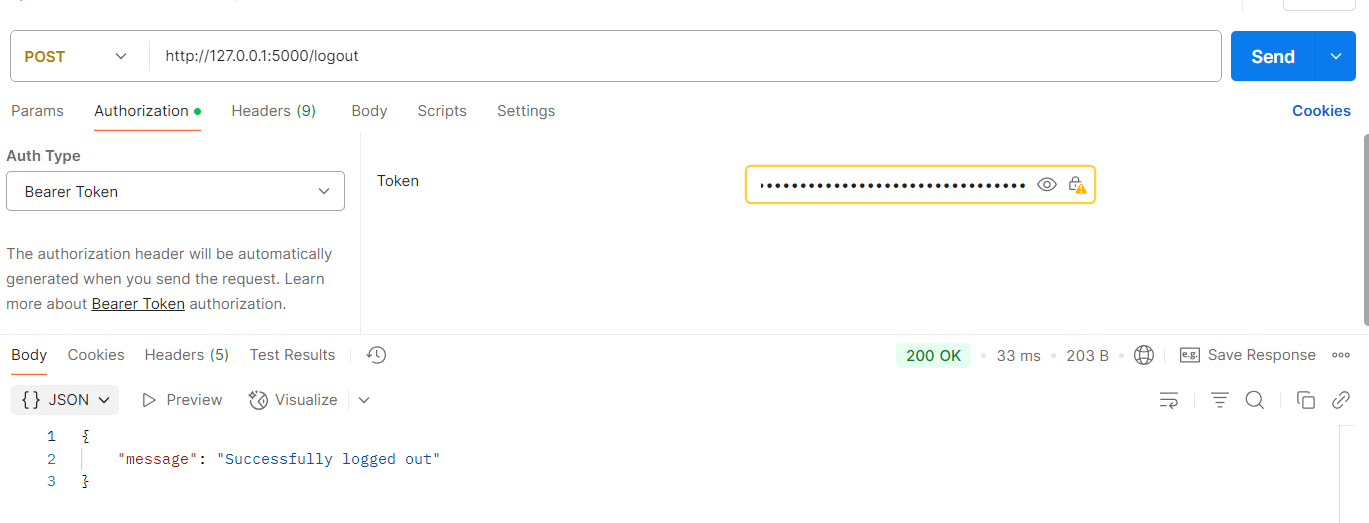
**ПРИМЕРЫ ЗАПРОСОВ И ИХ СТАТУСОВ**

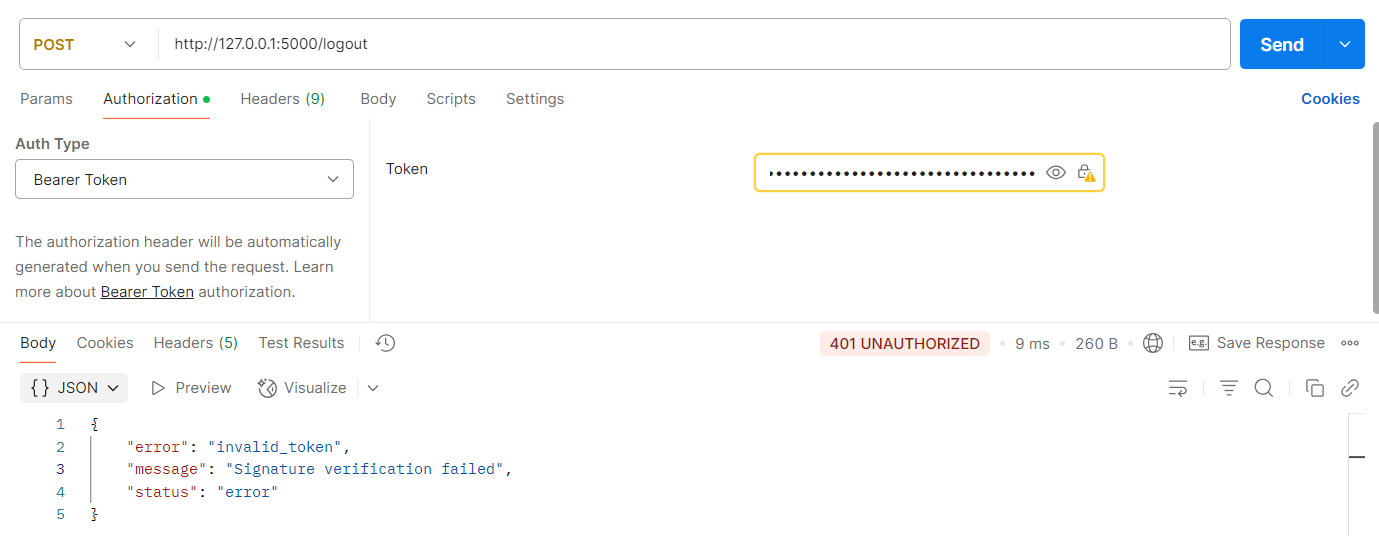
* Вход



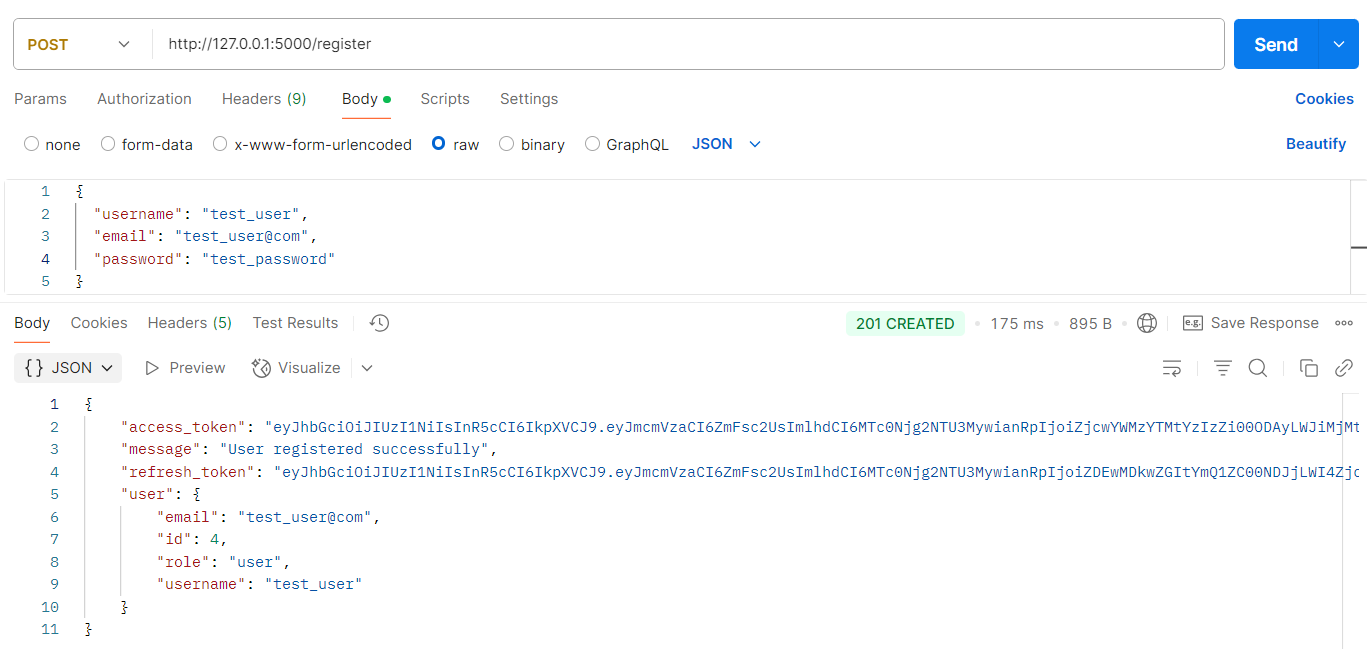


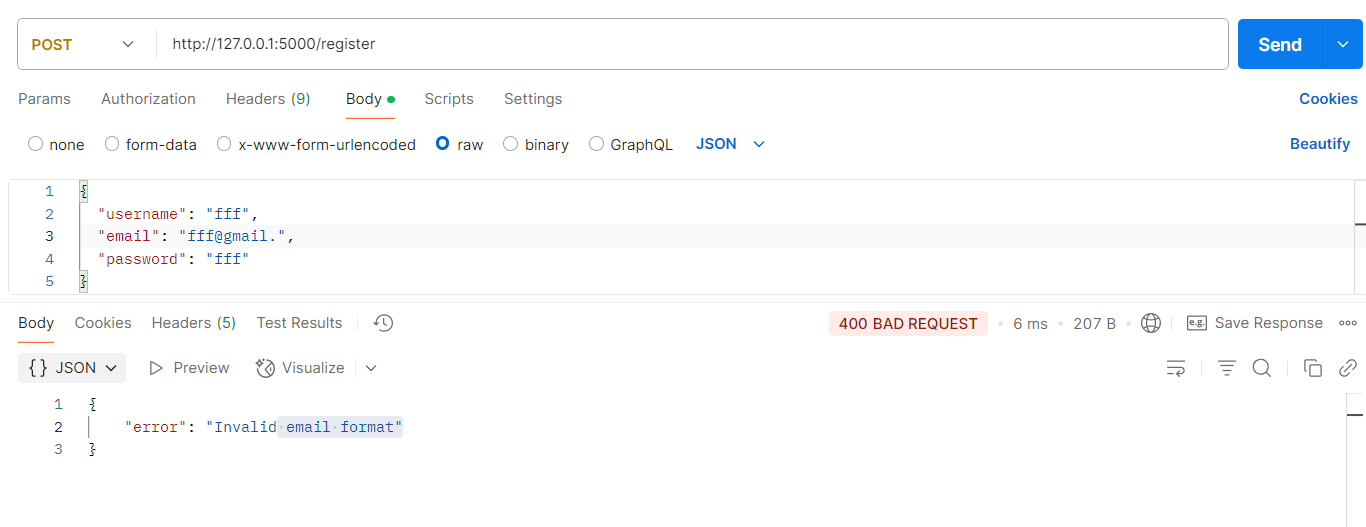
* Выход



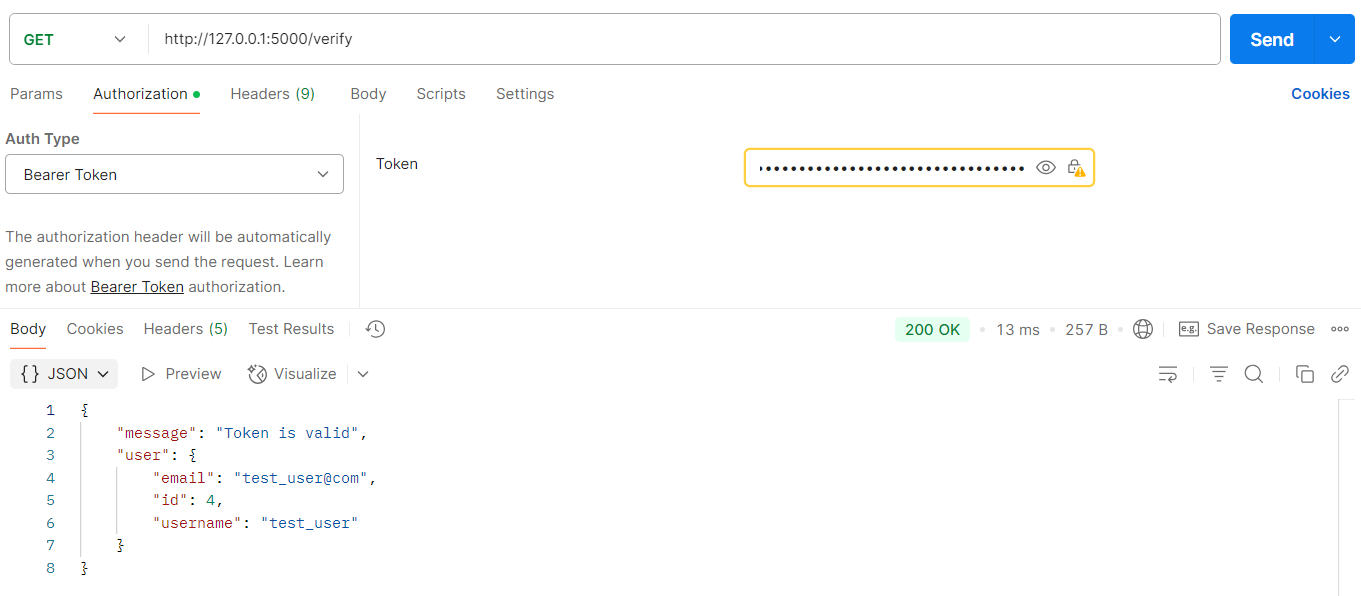


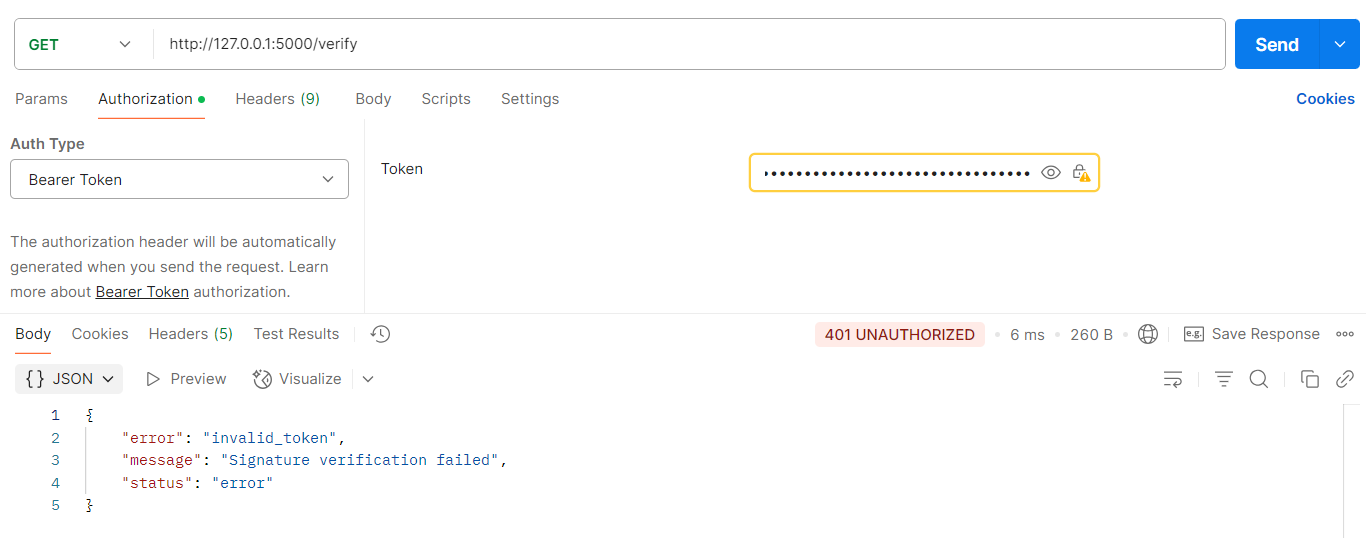
* Регистрация





* Верификация токена





* Смена пароля/имени/почты

