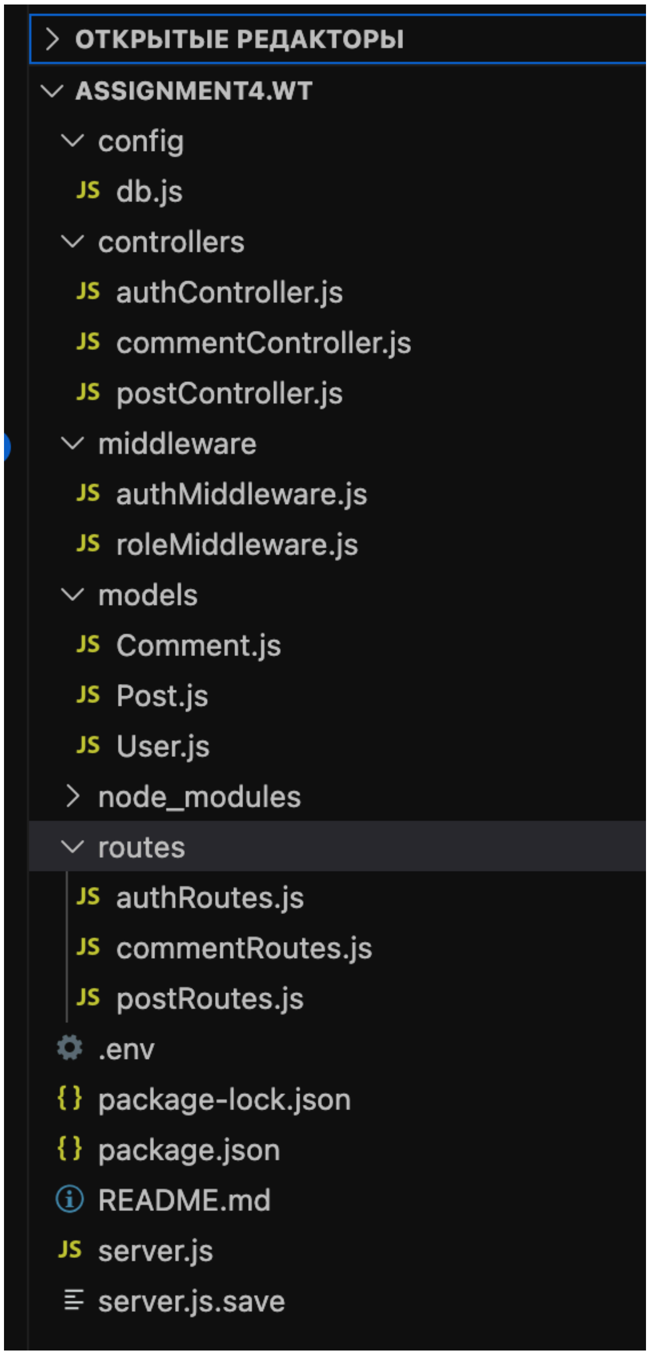
Aktoty Omar
SE - 2432

Assignment 4

1 Refined Project Architecture (MVC)

Code is cleanly separated into Models, Routes, Controllers, Middleware

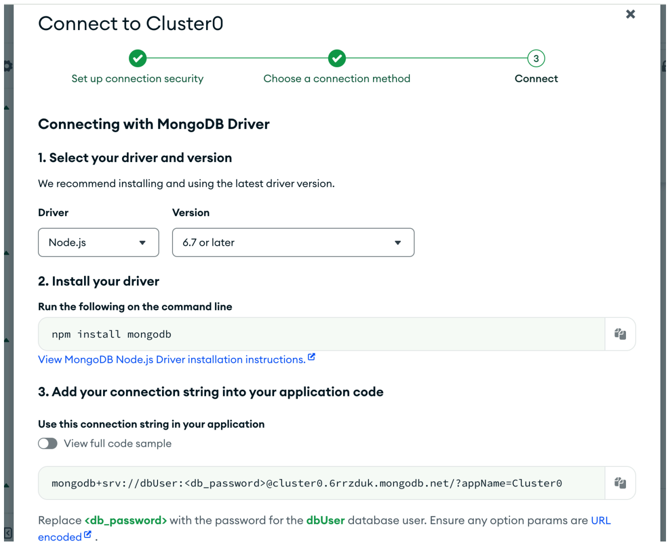
```
The project follows the Model-View-Controller (MVC) architectural pattern.

Models are responsible for defining the database structure using Mongoose schemas.
Routes define the available API endpoints.
Controllers contain the business logic and database operations.
Middleware is used for authentication, authorization, and request validation.

The server.js file is kept minimal and is only responsible for initializing the
server, connecting to the database, and registering routes.

This structure improves code readability, maintainability, and scalability.
```



```
> ОТКРЫТЫЕ РЕДАКТОРЫ
∨ ASSIGNMENT4.WT
  ∨ config
      db.js
  ∨ controllers
      authController.js
      commentController.js
      postController.js
  ∨ middleware
      authMiddleware.js
      roleMiddleware.js
  ∨ models
      Comment.js
      Post.js
      User.js
  > node_modules
  ∨ routes
      authRoutes.js
      commentRoutes.js
      postRoutes.js
    .env
    package-lock.json
    package.json
    README.md
    server.js
    server.js.save
```

```
The application uses three main models.

User model:
The User model contains email, password, and role fields.
Passwords are hashed using bcrypt before being stored in the database.
The role field is used for role-based access control and can be either "user" or
"admin".
```

```
Post model:
The Post model represents the primary object of the application.
It contains a title and content.

Comment model:
The Comment model represents the secondary object.
Each comment contains text and a reference to a Post using MongoDB ObjectId.
This creates a relationship between posts and comments.
```


Connect to Cluster0

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver: Node.js
Version: 6.7 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

View MongoDB Node.js Driver installation instructions.

3. Add your connection string into your application code

Use this connection string in your application

View full code sample

```
mongodb+srv://dbUser:<db_password>@cluster0.6rrzduk.mongodb.net/?appName=Cluster0
```

Replace **<db_password>** with the password for the **dbUser** database user. Ensure any option params are URL encoded .

2 Multi-Object CRUD (Primary + Secondary)
At least two related objects with full CRUD

Primary Object — `Post`

- Create → `POST /api/posts` (admin)
- Read → `GET /api/posts` (public)
- Update → `PUT /api/posts/:id` (admin)
- Delete → `DELETE /api/posts/:id` (admin)

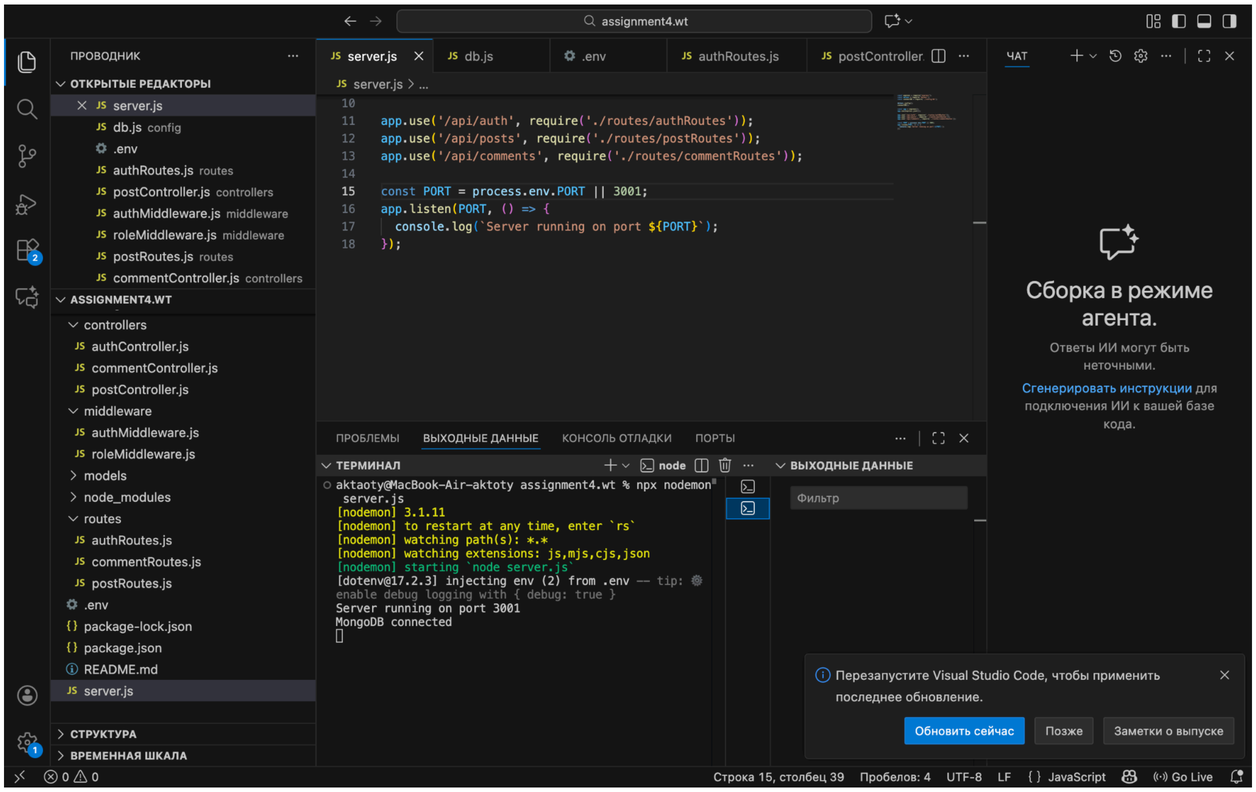
Secondary Object — `Comment`

- Create → `POST /api/comments` (admin)
- Read → `GET /api/comments` (public)
- Update → `PUT /api/comments/:id` (admin)
- Delete → `DELETE /api/comments/:id` (admin)

3 Authentication & RBAC

```
Role-based access control is implemented using middleware.

There are two roles in the system:

* user
* admin

Public access:
GET (read) requests are publicly accessible to all users.

Protected access:
POST, PUT, and DELETE requests require authentication.

Admin-only access:
Only users with the "admin" role can create, update, or delete posts and comments.

If a regular user attempts to access admin-only routes, the server responds with a 403
Forbidden error.
This behavior was verified and tested using Postman.
```

```
Authentication is implemented using JSON Web Tokens (JWT).

Users can register and log in using email and password.
After successful login, the server returns a JWT token.
This token must be included in the Authorization header when accessing protected
routes.

Passwords are never stored in plain text.
They are hashed using bcrypt to ensure security.
```



```
app.use('/api/auth', require('./routes/authRoutes'));
app.use('/api/posts', require('./routes/postRoutes'));
app.use('/api/comments', require('./routes/commentRoutes'));

const PORT = process.env.PORT || 3001;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

```
aktaoty@MacBook-Air-aktoty assignment4.wt % npx nodemon
 server.js
[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
[dotenv@17.2.3] injecting env (2) from .env -- tip: ⚙
enable debug logging with { debug: true }
Server running on port 3001
MongoDB connected
```

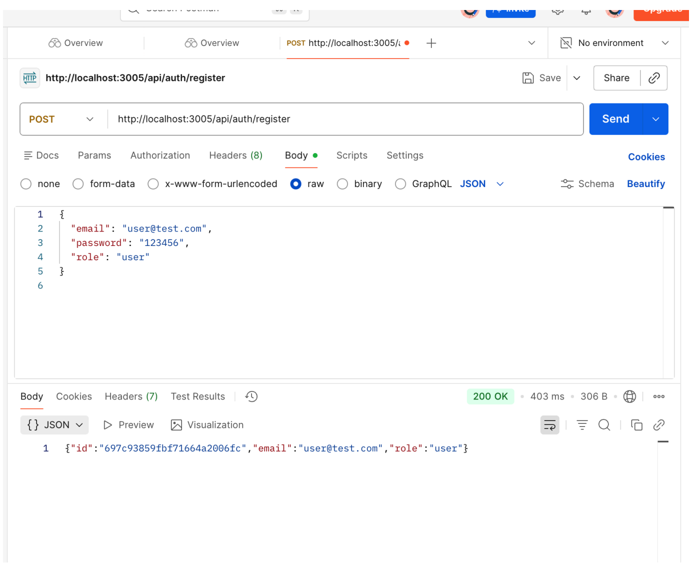
Перезапустите Visual Studio Code, чтобы применить последнее обновление.

Server is running

4Postman Collection

```
Both Post and Comment objects support full CRUD operations.

Posts:

* Create post (admin only)
* Read posts (public)
* Update post (admin only)
* Delete post (admin only)

Comments:

* Create comment (admin only)
* Read comments (public)
* Update comment (admin only)
* Delete comment (admin only)

Comments are linked to posts through ObjectId references, ensuring relational behavior
within MongoDB.
```

```
API TESTING WITH POSTMAN

All API endpoints were tested using Postman.

The Postman collection includes:

* User registration
* User login and JWT generation
* Failed requests from users without admin role (403 Forbidden)
* Successful requests from admin users (200 OK)
* CRUD operations for posts and comments

This confirms that authentication and role-based access control are working correctly.
```



http://localhost:3005/api/auth/register

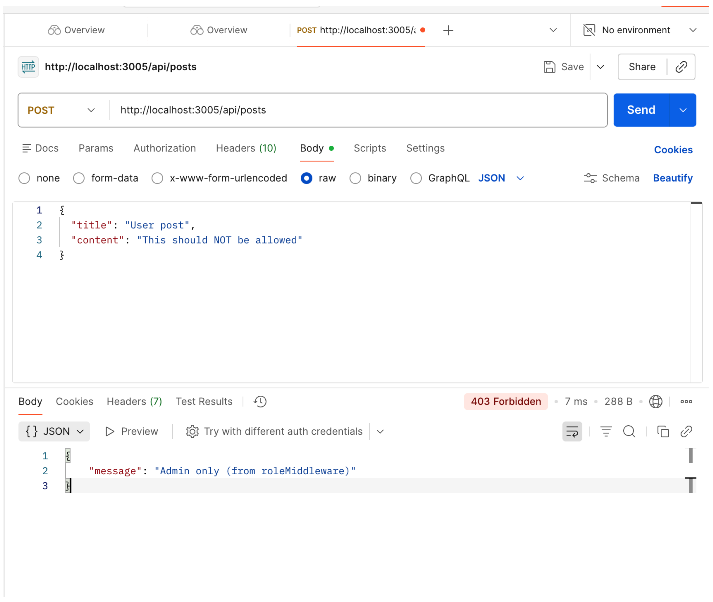
POST http://localhost:3005/api/auth/register

```
{
"email": "user@test.com",
"password": "123456",
"role": "user"
}
```

200 OK • 403 ms • 306 B

```
{"id":"697c93859fbf71664a2006fc","email":"user@test.com","role":"user"}
```

Post



http://localhost:3005/api/posts

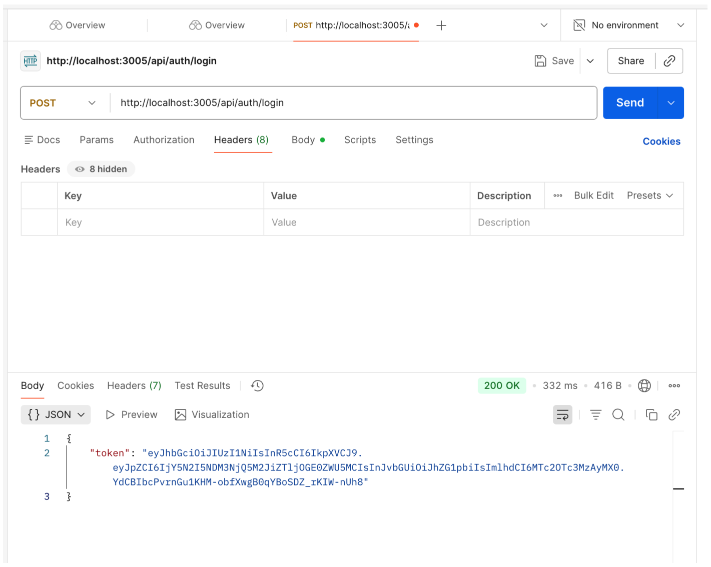
POST http://localhost:3005/api/posts

```
{
"title": "User post",
"content": "This should NOT be allowed"
}
```

403 Forbidden • 7 ms • 288 B

```
{
  "message": "Admin only (from roleMiddleware)"
}
```

Correct answer because Only users with the "admin" role can access POST / PUT / DELETE endpoints, and i prove it
When a regular user attempts to create a post, the request is rejected with a 403 status by the role-based middleware, which enforces admin-only access.”



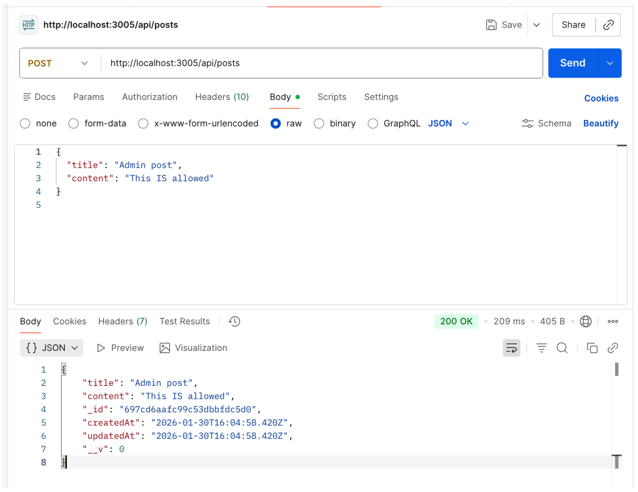
POST http://localhost:3005/api/auth/login

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6IjY5N2I5NDM3NjQ5M2JiZTljOGE0ZWU5MCIsInJvbGUiOiJhZG1pbiIsImlhdCI6MTc2OTc3MzAyMX0.
        YdCBIbcPvrnGu1KHM-obfXwgB0qYBoSDZ_rKIW-nUh8"
}
```

ADMIN LOGIN

Admin token

```
{
    "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5N2I5NDM3NjQ5M2JiZTljOGE0ZWU5MCIsInJv
bGUiOiJhZG1pbiIsImlhdCI6MTc2OTc3MzI5NH0.gvbCJzdl8mkcoQbnBI0lW2tDrhR87cwGbKxJFG6pidc"
}
```

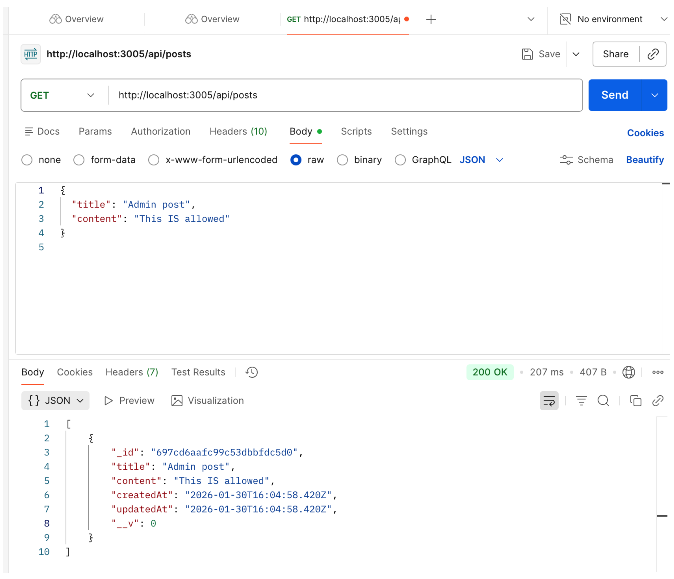


POST http://localhost:3005/api/posts

```
{
  "title": "Admin post",
  "content": "This IS allowed"
}
```

```
{
    "title": "Admin post",
    "content": "This IS allowed",
    "_id": "697cd6aafc99c53dbbfdc5d0",
    "createdAt": "2026-01-30T16:04:58.420Z",
    "updatedAt": "2026-01-30T16:04:58.420Z",
    "__v": 0
}
```

ADMIN create POST



Overview | Overview | GET http://localhost:3005/a | No environment

http://localhost:3005/api/posts

GET http://localhost:3005/api/posts — Send

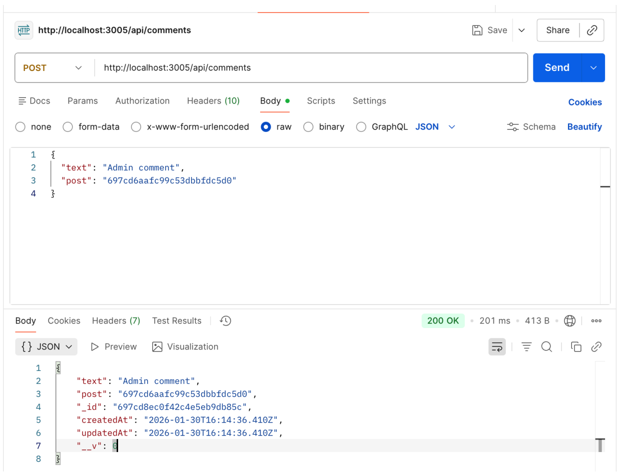
Body: raw JSON

```
{
  "title": "Admin post",
  "content": "This IS allowed"
}
```

200 OK · 207 ms · 407 B

```
[
    {
        "_id": "697cd6aafc99c53dbbfdc5d0",
        "title": "Admin post",
        "content": "This IS allowed",
        "createdAt": "2026-01-30T16:04:58.420Z",
        "updatedAt": "2026-01-30T16:04:58.420Z",
        "__v": 0
    }
]
```

GET POSTS (ПУБЛИЧНЫЙ ДОСТУП)



http://localhost:3005/api/comments

POST http://localhost:3005/api/comments — Send

Body: raw JSON

```
{
  "text": "Admin comment",
  "post": "697cd6aafc99c53dbbfdc5d0"
}
```

200 OK · 201 ms · 413 B

```
{
    "text": "Admin comment",
    "post": "697cd6aafc99c53dbbfdc5d0",
    "_id": "697cd8ec0f42c4e5eb9db85c",
    "createdAt": "2026-01-30T16:14:36.410Z",
    "updatedAt": "2026-01-30T16:14:36.410Z",
    "__v": 0
}
```

- MVC структура
- User / Admin
- JWT
- RBAC (403 для user, 200 для admin)
- Two related objects (Post ↔ Comment)

- CRUD
- Postman tests