Europace

Europace
Coding Challenge:
Token-Based Todo App (2 Microservices)

Welcome!
This small challenge is designed to help us understand how you approach API design, authentication logic, and service boundaries. It’s intentionally scoped to be completed in 4–6 hours.

🎯 Objective

Build two Spring Boot microservices that work together to support a basic Token-authenticated todo application.
One service handles user registration and login, the other manages user-specific todo notes.

💬 Before you start…

- **Please read the full task description before jumping in** – it will save you time.
- **Keep it simple and focused** – this isn’t about building a production-grade app.
- We care more about your **thinking, structure, and clarity** than about completeness.
- And most importantly: have fun building it!

📦 Tech Stack Requirements

- **Java or Kotlin** (your choice)
- **Spring Boot (**Use start.spring.io to get started)
- Mono-repo setup (both services in one repo)
- Containerized via Docker (use `docker-compose` or similar)
- Basic documentation:
  - How to run it locally
  - Example API requests (e.g. `curl` or Postman)

🧱 Services Overview

1️⃣ User Service (Authentication)

Handles user registration and login.

Endpoints:

- POST /register: Create new user with username + password
- POST /login: Authenticate user and return a **token**
- POST /token: Verify given token

Expected behavior:

- The token should include at least data for identifying the user
- Passwords can be stored in plain text (no need for hashing for this task)

2️⃣ Todo Service

Manages todos for authenticated users.

Endpoints:

- POST /todos: Add a todo item for the authenticated user
- GET /todos: Retrieve all todos for the authenticated user

Authentication Requirements:

- Requests must include a token in the Authorization header
- Verify token validity on each request

Responses:

- 401 Unauthorized: Token is missing or invalid
- 403 Forbidden: Token is valid but the user is trying to access data they shouldn’t
- 200 OK: Return todos for the user

🛠️ Bonus (Optional)

If you still have time or want to impress:

- Store data in an in-memory DB like H2
- Add basic unit or integration tests

📤 Submission

Please submit your solution as a **public Git repository** (e.g. GitHub, GitLab, Bitbucket).

✅ Your repo should contain:

- Working code for both services
- A README.md with:
  - How to start the app locally
  - Example API calls (e.g. using curl, Postman, or Swagger)
- Container setup or clear run instructions

📩 Just send us the **link to your repository** when you're done.

You're welcome to set the repository to private after we've reviewed it.