EPAM Practice - Designing a REST API

Description:
Students and professors schedule activities, receive notifications and reminders. API exposes CRUD operations for endpoints to trigger alerts.

Functional Requirements:
- CRUD(create/read/update/delete) courses, activities, schedules per user, notifications and webhooks (3rd-party callbacks).
- Generate alerts automatically by schedule rules.
- Users can acknowledge or dismiss alerts.
- Filtering, sorting, and pagination across list endpoints.
- HATEOAS links for workflow navigation.

Non-Functional Requirements:
- Security: OAuth2.0 Bearer JWT, role-based authorizatoin.
- Performance: cursor pagination, selective field expansion.
- Reliability: idempotent POST requests, optimistic concurrency with If-Match.
- Observability: X-Request-Id correlation, structured error format.
- Time: All datetimes ISO-8601 UTC.

Entities:
- User: id, email, name, role (student|instructor|admin), timezone, createdAt, updatedAt
- Course: id, code, title, ownerUserId, createdAt, updatedAt
- Activity: id, courseId, title, description, dueAt, recurrence, reminders, status, createdBy, createdAt, updatedAt
- Schedule: id, userId, active, createdAt, updatedAt
- Alert: id, activityId, userId, scheduledAt, sentAt, status, delivery, retries, metadata, createdAt, updatedAt

Conventions:
- Auth: Authorization: Bearer JWT
- Tracing: optional X-Request-Id
- Idempotency: Idempotency-Key
- Concurrency: If-Match: ETag on PUT/PATCH/DELETE
- Pagination: cursor style
- Filtering: filter[field]=value, ranges: filter[dueAt][gte]
- Sorting: sort=field
- Media: application/json

API Descriptions:
- **User**

    - List all users: GET /users
    - Get a user: GET /users/{id}
    - Creaet a user: CREATE /users. Body: email, name, role, timezone. Idempotent
    - Update a user: PUT /users/{id}
    - Delete a user: DELETE /users/{id}
- **Course**
    - List all courses: GET /courses
    - Get a course: GET /courses/{id}
    - Create a course: CREATE /users/ Body: code, title, ownerUserId
    - Update a course: PUT /users/{id}
    - Delete a course: DELETE /users/{id}

Status Codes:

- 200 OK – successful retrieval or updatel.
- 201 Created – resource created and location header to new resource.
- 204 No Content – successful operation with no body,
- 400 Bad Request – invalid input.
- 401 Unauthorized – missing or invalid token - not authorized.
- 403 Forbidden – not enough priveliges
- 404 Not Found – resource is missing in the server.
- 500 Internal Server Error – unexpected error from the server side.