Senior Backend Developer Technical Assignment

Create and deploy a simple calendar REST api.

Application development

Create a simple REST api for calendar management application.

Calendars

A `Calendar` entity defines a single calendar. It has at least the following fields:

- `id` - unique identifier, generated on calendar creation
- `name` - calendar name, can be changed later

Add other fields if needed.

Endpoints:

- `POST /calendars` - create a new calendar, with a `name`.
- `GET /calendars/:calendarId` - get single calendar details by `id`.
- `PUT /calendars/:calendarId` - update single calendar details (ex. `name`).
- `DELETE /calendars/:calendarId` - delete single calendar by `id`, should also delete all related calendar entries.

Calendar Entries

A `CalendarEntry` entity defines an event on the calendar. It has (at least) the following fields:

- `id` - unique identifier, generated on event creation.
- `title` - event title, can be changed later.
- `start` - DateTime value, defines when event starts.
- `duration` - event duration. Use the duration type/format you consider fits best.

Add other fields if needed.

Endpoints:

- `POST /calendars/:calendarId/entries` - create new `CalendarEntry`, with a `name`, `start` and `duration`
    - It Should throw an error if event overlaps with other event
    - Add an option to force-create overlapping events
- `GET /calendars/:calendarId/entries` - list calendar entries, with *required* `start`/`end` datetime filters.
- `PUT /calendars/:calendarId/entries/:entryId` - update single calendar entry details (ex. `name`, `start`, `duration`)
    - It Should throw an error if event overlaps with other event
    - Add an option to force-create overlapping events
- `DELETE /calendars/:calendarId/entries/:entryId` - delete single calendar entry by `id`;

Requirements

- Include input validation.
- Use a database of your choice (Sql or NoSql).
- Use a framework of your choice (ex. express, fastify, ...)

4. Bonus Tasks (Optional but Encouraged)

- Implement recurring events support
    - A recurring event can repeat every `x` days, weeks or months
    - Event recurrence can (optionally) stop after a given number of occurrences or at a point in time (DateTime).
    - If no stop criteria is given - must be repeated indefinitely
    - Update and Delete operations over a recurring event may be applied to single occurrence of starting from given occurrence and to all future occurrences
    - Adapt API endpoints (add missing data)
- Implement authentication (e.g., JWT) for the API.
    - In this case calendars (and entries) have an owner user, all endpoints should be adapted.
- Use caching (e.g., Redis) to optimize GET requests (ex. list calendar entries).
- Add API documentation

Infrastructure & Deployment

Containerize and deploy the application.

Requirements:

- Write a `Dockerfile` to containerize the application.
- Create a `docker-compose.yml` for local development (including the database). Start local development with a single command.
- Use Kubernetes (or a lightweight alternative like Minikube or K3s) for deployment.
- Include load balancing for the API.
- For a cloud platform deployment (AWS, GCP, Azure) - managed Kubernetes, LB and DB are ok (but not required).
- Automate deployment using Infrastructure as Code (IaC) with a tool such as Terraform, CloudFormation, or Pulumi.

4. Bonus Tasks (Optional but Encouraged)

- Use a cloud platform (AWS, GCP, Azure) for deployment.
- Setup CI/CD pipeline for build and deploy
- Configure basic monitoring/logging (e.g., Prometheus, Grafana, or ELK stack).

Evaluation

Push the result to a git repository (or multiple repos).

Add README with instructions on how to start local development and how to deploy the app.

Based on your time constraints - you can reduce functionality to save time, but the delivered solution must be complete (cover application development AND infrastructure setup).