Booking API

Hello!

This exercise isn't supposed to bring you any discomfort nor any unnecessary stress. The goal here is to create an opportunity for us to discuss technical decisions and real use cases.

You will be given one or two days to accomplish this so that you can manage your own workload and life (to be pre-defined). We value simplicity, pragmatism and the discussion around the reasoning behind your decisions. Please, don't feel the need to put a lot of hours into this exercise.

Context

This exercise is based on a real project but a little (a lot) simplified. On this project Users can be Booked by Agents so they are available to perform specific tasks (which don't matter for the sake of this exercise)

The goal is to build a Node API which will serve two different applications. However, these two applications will share the same database, authorization logic (roles) and some common endpoints.

Feel free to ask questions, communicate and make any assumption you find necessary as long as you can justify it. Don't worry if you can't do everything. What's more important is the decisions, time management, and the process itself rather than just the outcome of this exercise.

Requirements

- Node.js
    - Needs to support es6 modules
        - Eg: *await* and *async* premises
        - Eg: *import* instead of *require*
- App will serve two distinct APIs
    - Business Api
    - Client Api
    - Some endpoints will be shared
- Api will share same Database
    - Database should be relational (MSSQL)
    - However, if you face difficulties setting it up, feel free to default to any other DB
- Simplified Authentication
    - However the API should expect an header "X-Agent-Id" which contains a specific agent ID, so we can identify who is using this endpoint
- Authorization

    - We expect this api to have different authorization levels.
    - This section should be easily customizable so that we can create more roles anytime and associate them with authorized endpoints
    - For now let's consider just two roles for each Agent:
        - *Admin* - can do everything
        - *Regular* - Can read
- Specs
    - Choose a test stack for this API
    - Implement just some tests that you find more important
    - Consider any other important test strategy for our late discussion, but you don't need to implement it
- Database - feel free to adapt anything you might need
    - User
        - name
        - email
        - has_many bookings
    - Booking
        - belongs to User
        - belongs to Agent (who performed the Booking)
        - start_at
        - finish_at
    - Agent
        - name
        - email
        - has_many bookings (booked_by)
        - has_many users

API Specification

Client API

Controller	***Method***	**Endpoint**	**Purpose**
Scheduler	*GET*	/scheduler?week=weekdate	Returns all bookings and user data for that specific week - REGULAR, ADMIN
Booking	*POST*	/booking	Create a booking for a user which belongs to the current agent - ADMIN
Booking	*DELETE*	/booking/:id	Delete booking - ADMIN

Business API

Controller	***Method***	**Endpoint**	**Purpose**
Scheduler	*GET*	/scheduler?week=weekdate	No need to be implemented. Leave just a scaffold and return 200 OK

Common API endpoints

Controller	***Method***	**Endpoint**	**Purpose**
User	*GET*	/users	Return all users for that Agent - REGULAR, ADMIN
Agent	*GET*	/agents	Return all agents - ADMIN

Deliverables

- Github link (or another platform) with code - id**eally do regular commits and push them along the day**
- Any documentation you find useful (can be on readme or whatever fits you best)
- Quick Q&A at a pre-scheduled time