MonkeyShelter - My Approach

The Monkey module was an engaging and time-sensitive task, with a completion window of three days. This required focused effort to ensure all necessary endpoints were implemented thoroughly, including handling edge cases and potential inconsistencies.

Endpoints & Logic

/api/monkeys POST

This endpoint uses IMonkeyService from the contracts project and the repository implementation to fetch the monkey from a database. This POST endpoint is authenticated with the ShelterId JWT key, since, as per the requirement, “*Shelter managers may only manage monkeys in their own shelter.*”

Monkey admission is logged here with the Admissions table, referencing monkeyId

/api/monkeys DELETE

This endpoint creates a departure in the database, while simultaneously removing the monkey from the shelter, after verifying the conditions meet. This is authenticated as well, to follow the business rule.

Both of these endpoints might fail if called concurrently from multiple devices, since the counter occurs just before the monkey is added. Another approach could be here the use of transactions or stored procedures that will internally check the business logic.

/api/monkeys PATCH

This endpoint updates the weight based on the provided Id. This is also authenticated.

Veterinarian Check-up

This part of business logic is ingrained in the Domain model and is visible when querying GET /api/monkeys
This way we can find out when are the upcoming vet checks and schedule them timely.

Reports

/GET /api/reports

These two endpoints in the *ReportsController* get monkeys' information based on species and dates. Smart Invalidation Cache is used here.

The Frontend

With my limited experience in Angular, I tried to make a working solution, using Bootstrap on HTML and I've separated each of the endpoints as components. I realize that this might not be the most optimal way, but with the limited time I had, this is the option I chose.