We'd like to get a feel for how you approach problems, how you think, and how you design your code.
Please complete the exercise below in Typescript and upload your code to a github repo!
Thank you and have fun!

Stories:

1. A company has a name and address
2. A company can have multiple job vacancies
3. A company has many users
4. A vacancy has a title, description, expiredAt (datetime)
5. A user has a name, username, password
6. A user belongs to one company only
7. A user can have two types of roles: user and admin
8. A user with an admin role can view, create, edit, and delete vacancies
9. A user without an admin role can view job vacancies only
10. A user has to login first before doing any operation

Requirements:

Create a set of back-end apis to manage users and vacancies (No need to implement front-end UI).

1. Use Node.js + typescript + mongodb
2. Use Nest.js (recommend) or koa
3. Use Microservices (note: consider how to separate microservices)
4. Use BFF (backend for front-end) to create front-end related api
5. Use REST API for the db service and use GraphQL for the BFF service. For GraphQL, you can use Apollo
6. Unit tests are required
7. Dockerize your microservices
8. Upload your project to github and write a proper readme.

Seed Data:

You don't need to implement apis to create companies and users (but you need to implement login). Instead, just focus on the vacancy apis and architecture design. We will provide the seed data (company and user) as follows:

Company:

_id	name	address

ObjectId("5e5df7fc6953acd3dc50fe8f")	PredictiveHire	15 Newton St

Users:

_id	companyId	name	username	password	role
ObjectId("5e5df7f450571fb3aecdcf21")	ObjectId("5e5df7fc6953acd3dc50fe8f")	Bob Markle	bob	bob	user
ObjectId("5e5df7f450571fb3aecdcf21")	ObjectId("5e5df7fc6953acd3dc50fe8f")	Mark Smith	mark	mark	admin