Take Home Challenge: Backend Engineer with Infrastructure

> We've designed the following task for you. Impress us with your ideas and your obsession for simplicity.

The Challenge

As a member of the Prisma Data team you will work on the core components of the data infrastructure of our customers. You will work with a wide range of databases, design sophisticated and backwards compatible APIs while making sure everything will also work at high scale. To get an idea of your skills and your approach to coding, we'd like you to complete the task below and answer the questions to it. Ah, and before you start, please read all the below carefully first.

Task

- Estimated time commitment: 3 - 6 hours

- Build a microservice that stores & retrieves `blog` entities and their related `post` entities.
- Implementation should be simple without looking to initially serve high levels of traffic.
    - We expect high levels of growth in time so leave room for the microserve to scale
- Populate your readme with answers to the questions below
- You will have an opportunity to discuss your implementation as the interview process continues

Bonus

The CTO of the startup told you that you are free to choose which database you want to use. He asked you to pick something that you are familiar with. He is sure that they will in the long run switch to a different database that is best suited for their most common query patterns. He therefore asked you to design the application in a way that swapping out the database is relatively easy.

It is fine if you don't manage to implement the bonus task. We would love to hear though how you would tackle this challenge.

Data Model

The business gave you the following rough guidelines for the data model:

- A `blog` has the following fields:
    - `name`: required String
    - `slug`: required String that is unique across all Blogs
    - `posts`: A blog may have many posts.
- A `post` has the following fields:
    - `title`: optional String
    - `content`: required String
    - `viewCount`: required Integer with a default value of 0
    - `blog`: A post is related to exactly one Post.

Functional Requirements

- It must be possible to create a new `blog` entity while simultaneously creating one or more `post` entities that are linked to the new `blog` entity in a single request.
- It must also be possible to create a new `post` entity and connect it to an existing `blog` entity.
- It must be possible to query a `blog` entity by any unique criteria. In this query it must optionally be possible to get the related `post` entities as well. So the client is in control of whether the related `post` entities should be returned or not.
- Please do not use Prisma for this. We would like to see how you implement on your own.

Questions

- What were some of the reasons you chose the technology stack that you did?
- What were some of the trade-offs you made when building this application? Why
  were these acceptable trade-offs?
- Given more time, what improvements or optimizations would you want to add? When would you add them?
- What would you need to do to make this application scale to hundreds of thousands of users?
- How would you change the architecture to allow for models that are stored in different databases? E.g. posts are stored in Cassandra and blogs are stored in Postgres.
- How would you deploy the Architecture you designed in a production ready way?

Important: For us, the how and the why are important. Sending us your results somewhat incomplete is totally fine.

Submission

Please submit your assignment via email to:

- hogarth@prisma.io
- rensburg@prisma.io

If you are having any trouble, please contact rensburg@prisma.io

We wish you good luck and are really curious to see your ideas!