Natural Cycles Backend Challenge 3

Technologies

- NodeJS
- ExpressJS
- Cloud deployment

Expected output

Please provide the solution for this challenge as a link to a github or bitbucket repository created by you.

Please provide a meaningful “readme.md” with the instruction of how to run the project. If you provide a link to a live online demo - it’s a plus! If we experience problems following your setup instruction and running the demo locally - it’s a minus (regardless of how good the solution itself is). So, “packaging” the result of your work is an essential part of the work.

If you work on a solution and commit your progress to github “as you go” (as you would in a real project) - that’s a plus! We want to see your commit granularity and comments.

Take as much time as you need. We welcome both junior and experience developers in our team. If you’re junior and need time to learn the required technologies - go ahead and learn. But in the end we judge the result equally.

The task

We ask you to create a small CRUD service with REST API and a simple UI.

It should allow to Create, Read (list), Update and Delete Accounts. Account entity should have just 2 fields: id (string) and email (string).

ID should be somewhat cryptographically globally unique. Email should be valid email, no invalid entries are allowed in the DB. It’s enough to have validation just on the Backend (not in the Frontend), since it’s a Backend position.

For the DB we suggest to use some online-hosted DB, so you can easily deploy your service to the cloud to show us. We recommend either Firebase realtime DB or mlab.com.

To deploy your code to the cloud feel free to use any provider that suits you. We recommend now.sh as the simplest one.

UI can be as simple as possible, no CSS or styling is needed (cause it's not a Frontend position), it should "just work". It should show the list of all Accounts in the DB (no paging required). Should have "Create" button where you specify ID and email and can Insert a record in the DB. ID should be autogenerated and unique, email should be valid and also unique.

There should be "Delete" button to delete Account from DB. "Update" button should allow to change email of the Account (not ID, it's autogenerated and immutable).

It should be possible to run the service on a local development machine and to test the deployed version in the cloud.

Please describe in "readme.md" what would you do differently if this service should be used in a production environment, what would you change in your current implementation. Is it production-ready? What would make it production-ready?

Please use prettier to format you code (we use it internally to not have discussions or conflicts around how to format and style the code, this will help us to read your code easier), thanks. Please use these settings as a `prettier.config.js`:

```
module.exports = {
  printWidth: 80,
  tabWidth: 2,
  useTabs: false,
  semi: false,
  singleQuote: true,
  trailingComma: 'all',
  bracketSpacing: true,
}
```

Bonus tasks

Bonus tasks are optional. Feel free to do them if you want to impress us;)

Add auth layer to the service, so admin user needs to be authenticated to view/edit/etc the rows. So, there is a Login button to log in initially, and a Logout button. You can use e.g Firebase Auth for that.

Use Typescript (cause we use it everywhere internally).

Create Unit tests (we're using Jest for simple tests, but feel free to use any tool/framework).

Setup a Continuos Integration platform (e.g CircleCI - they have a free plan for 1 container) that will deploy your service every time you make a git push.

What is important

- How well your solution solves the problem.
- How "testable" is your code, are your tests doing the job.
- The architecture of your solution, the overall picture.
- How well packaged your solution is, is it easy to install and run.

Good luck! Show us that you're good and we'll be more than happy to welcome you to our team!