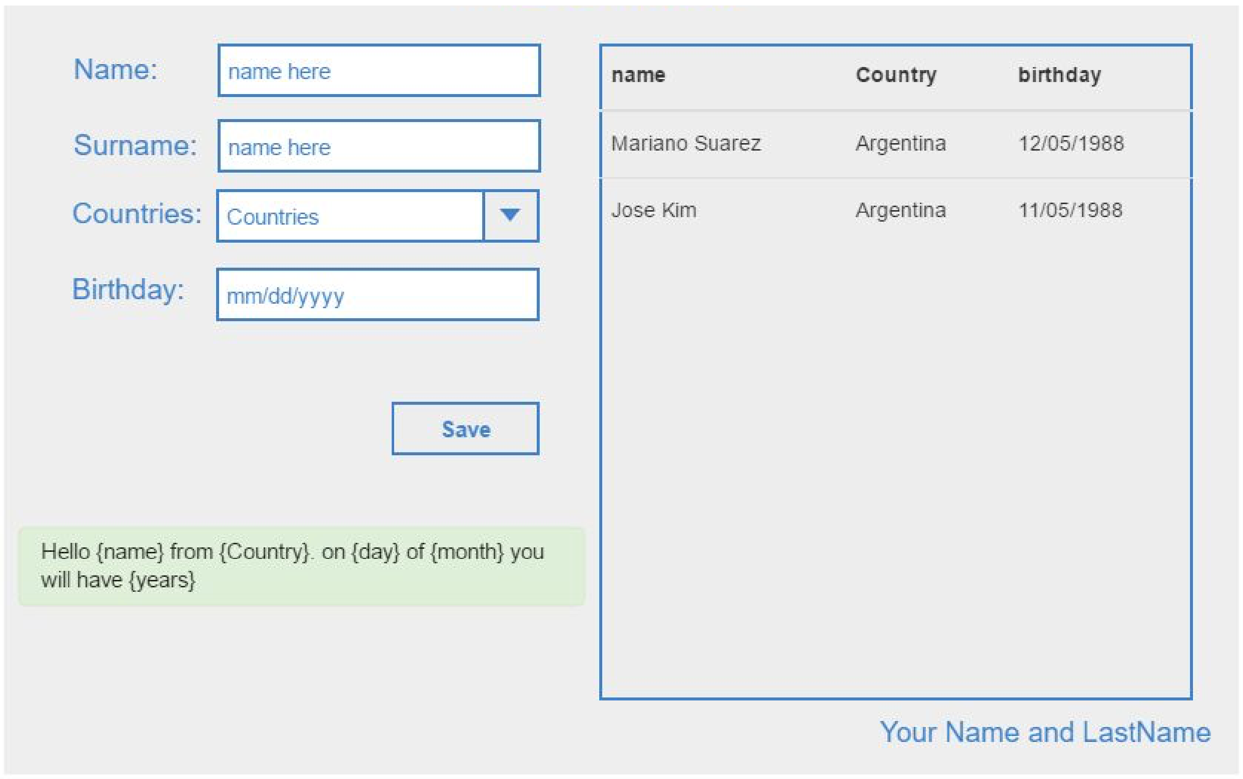
MOSANO

React.js & Node.js Challenge

Introduction:

Develop a web platform with the given interface that allows the users to register their full name, country, and birthday. You are given 3 options both for the backend and for the frontend, and you should pick one for each according to what you are more comfortable with.

Design and Functionality:



Hello {name} from {Country}. on {day} of {month} you will have {years}

name	Country	birthday
Mariano Suarez	Argentina	12/05/1988
Jose Kim	Argentina	11/05/1988

Create a container with components that bring up a form with:

- **Name**
- **Surname**
- **Country (dropdown of countries)**
- **Year of birth**
- **Save button**

Once the form is saved, it should show a message that refers to the next birthday as such:

Hello** {name} **from** {country}**. on** {day} **of** {month} **you will be** {years} **old!

Also has to show a list with all the entries made.

Development

BACKEND - API

For the country list, use one of the following options:

Option A (for fullstack):
Code a basic REST API using NodeJs and Express (express is optional, any framework will do), that serves a static list of countries (just one or two, no need to work the data), similar to (note that it just needs to contain a couple fields):
https://restcountries.eu/rest/v2/all

Option B (for frontend only, if you are applying for a frontend only position):
Use the following endpoint to fetch the list:
https://restcountries.eu/rest/v2/all

Option C (for fullstack):
Code a fully featured Rest API using NodeJs, Express, and storing a dynamic list of countries on a MongoDB database.. Take into account that:

- The create, update, and delete methods should be protected with any sort of authentication system (basic auth, token, static password will do, etc)
- The listing should be public
- You just need one or two, no need to work the data

Extra points if:

- Use a MongoDB ORM (ex: mongoose)
- Use Typescript
- Use GraphQL instead of REST (only for fullstack)
- Include a basic unit test (please don't code tests)

- Use ES6/7/next (or the new cool name that people will call it 2 months after writing this document)

FRONTEND - APP

Option A:

- Create a React app with a container with components where you can enter the data and when you save, the greeting is shown. (no persistent data or previous entries needed, just store on container state)
- The resources can be retrieved by a simple fetch/http request on container lifecycle

Option B:

- Create a React app that uses a state management solution with a container with components where you can enter the data and when you save it should trigger a mechanism within the state management engine.
- Coding Standards and ES6 / ES7 are required

Option C:

- Create a React app that uses a state management solution with a container with components where you can enter the data and when you save it should trigger a mechanism within the state management engine.
- It must use Observables or Promises, and async/await features
- The old entries must be available in *http://<app>/revisited,* and it should be protected route through an HoC (high order component) with an auth key (that can be hardcoded in the code) or other reusable pattern
- Extra points for supporting i18n, i.e. the app has to change the language for EN (english) and PT (portuguese)

Extra points for all options if:

- When you click on one of the previous visitors, redraw the legend of the greeting.
- You use a solid strategy for form management and form validation
- Use Graphql instead of REST API (Applicable if you did it on backend)
- Standard Coding and ESLint is used
- Use Typescript
- Use React Hooks
- You code in (or integrate) animations (spinners, loaders, etc) on data fetching, saving form, and greeting changes
- You use code generators to automate some tasks such as graphql code generation, etc.