TEST

Bazire, during our interview, we briefly evoked Spring being be a good candidate to replace Rails.

You triggered my curiosity. This test is about demonstrating Spring capabilities. Follows a simple backend specification. It's a relatively simple CRUD backend api,JSON friendly, with a few tables, and a couple business rules regarding the expected output.

The specification is quite verbose - although not fully completed (mainly because I believe there is no need to go into even more details)

You shall use all the tools/libraries you wish.

I wrote some pretty complete specifications, although, you don't have to follow them to the letter, as long as you keep the same functionalities (CRUD) and the same JSON output for the GET requests.

Also I provided a sqlite database that already has the tables and some seeded data for you to play with. Of course you're free to build your own from scratch on whatever DB engine you see fit.

When you are done, I hope we can do a 15/20 minutes code review.

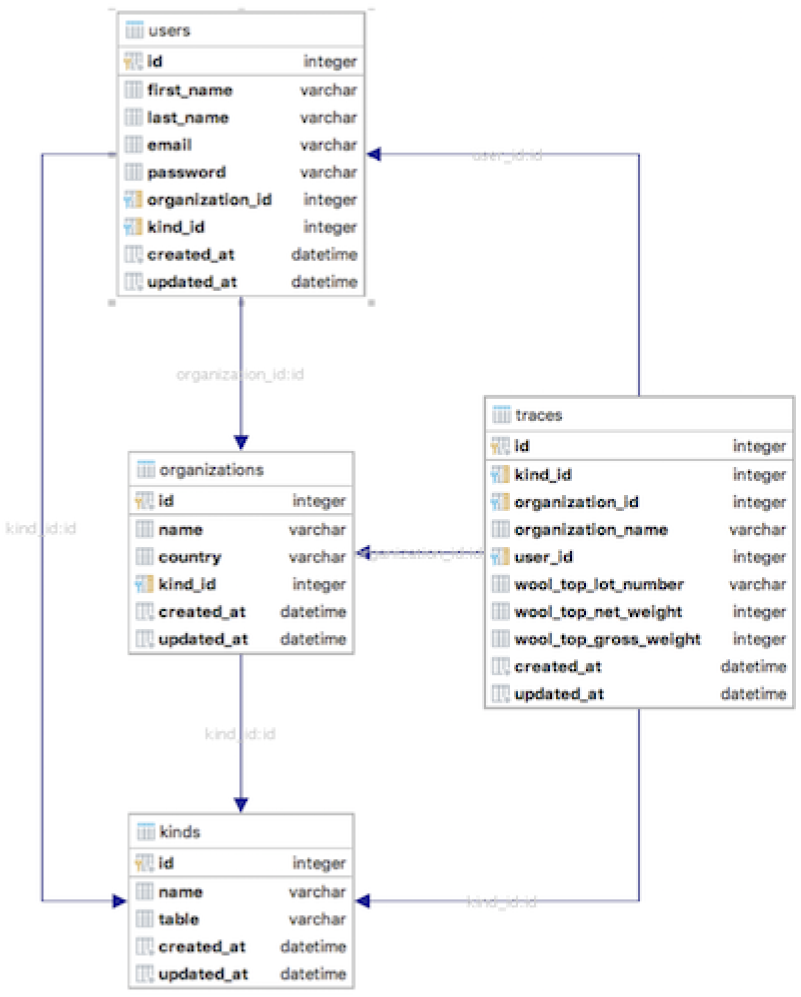
1. Backend specifications (provided)

1.1. Tables :

1.1.1 Table names :

- Organizations
- Users
- Kind
- Traces

1.1.2. Database Schema



1.1.3. Table relations

An organization belongs to a kind. An organization kind must have a "table" value equal to "Organization".

A user belongs to a kind. A user's kind must have a "table" value equal to "User"

A trace belongs to a kind. A trace kind must have a "table" value equal to "Trace"

A user belongs to an organization. An organization has many users. An organization has many traces.

A kind may have many users, organizations or traces, depending on the value of the "table" field (ie: "User","Trace" or "Organization")

1.2. Endpoints

Endpoints	Prefix	URL	description
kinds	GET	/kinds(.:format)	get all kinds
	POST	/kinds(.:format)	create a new kind
kind	GET	/kinds/:id(.:format)	get a kind by id
	PATCH	/kinds/:id(.:format)	update a kind by id
	PUT	/kinds/:id(.:format)	replace a kind's data by id
	DELETE	/kinds/:id(.:format)	delete a kind, by id
users	GET	/users(.:format)	get all users
	POST	/users(.:format)	create a new user
user	GET	/users/:id(.:format)	get a user by id
	PATCH	/users/:id(.:format)	update a user by id
	PUT	/users/:id(.:format)	replace a user data, by id.
	DELETE	/users/:id(.:format)	delete a user, by id
traces	GET	/traces(.:format)	get all traces
	POST	/traces(.:format)	create a new trace
trace	GET	/traces/:id(.:format)	get a trace by id
	PATCH	/traces/:id(.:format)	update a trace, by id
	PUT	/traces/:id(.:format)	replace a trace, by id
	DELETE	/traces/:id(.:format)	delete a trace, by id
organizations	GET	/organizations(.:format)	get all organizations
	POST	/organizations(.:format)	create a new organization
organization	GET	/organizations/:id(.:format)	get an organization, by id
	PATCH	/organizations/:id(.:format)	update an organization, by id
	PUT	/organizations/:id(.:format)	replace an organization, by id
	DELETE	/organizations/:id(.:format)	delete an organization, by id

1.2.1 kinds

Endpoints	Prefix	URL	description
kinds	GET	/kinds(.:format)	get all kinds
	POST	/kinds(.:format)	create a new kind

Business rules :

none

Examples :

- **GET** (get all kinds):
- *example :* `curl http://localhost:3000/kinds` *returns a JSON array containing all kinds, ordered by id :*

```
[
{"id":1,"name":"COMBINGMILL","table":"organization"},
{"id":2,"name":"SPINNINGMILL","table":"organization"},
{"id":3,"name":"ADMIN","table":"organization"},
{"id":4,"name":"CM-TR010","table":"trace"},
{"id":5,"name":"CM-TR020","table":"trace"},
{"id":6,"name":"SM-TR010","table":"trace"},
{"id":7,"name":"SM-TR020","table":"trace"},
{"id":8,"name":"ADMIN","table":"user"},
{"id":9,"name":"USER","table":"user"}
]
```

- **POST** (create a new kind) :

  example : `curl http://localhost:3000/kinds/ --request POST --header "Content-Type:application/json" -d '{"name":"DISABLED","table":"user"}'` *returns the newly created kind item as JSON :*

  ```
  {
    "id":10,
    "name":"DISABLED",
    "table":"user",
    "created_at":"2018-10-14T16:31:42.311Z",
    "updated_at":"2018-10-14T16:31:42.311Z"
  }
  ```

1.2.2. Kind

Endpoints	Prefix	URL	description
kind	GET	/kinds/:id(.:format)	get a kind by id
	PATCH	/kinds/:id(.:format)	update a kind by id
	PUT	/kinds/:id(.:format)	replace a kind's data by id
	DELETE	/kinds/:id(.:format)	delete a kind, by id

Business rules :

none

Examples :

- **GET** (get a kind by id)

  example : `curl 'http://localhost:3000/kinds/1'`*returns the kind wich id's is 1 as a JSON :*

  ```
  {
    "id":1,
    "name":"COMBINGMILL",
    "table":"organization",
    "created_at":"2018-09-28T06:27:03.945Z",
    "updated_at":"2018-09-28T06:27:03.945Z"
  }
  ```

- **PATCH**

  example : `curl http://locaLhost:3000/kinds/1 --request PATCH --header "Content-Type:application/json" -d '{"name":"COMBINGMill"}'` *returns the newly created kind item as JSON :*

  ```
  {
    "id":1,
    "name":"CombingMill",
    "table":"organization",
    "created_at":"2018-10-14T16:31:42.311Z",
    "updated_at":"2018-10-14T17:41:42.311Z"
  }
  ```

- **PUT**

- **DELETE**

1.2.3. Users

Endpoints	Prefix	URL	description
users	GET	/users(.:format)	get all users
	POST	/users(.:format)	create a new user

Buisiness rules :

A user's organization and kind are embedded in the JSON responses (cf. examples)

Examples :

- **GET** (all users) :
- *example :* `curl http://localhost:3000/users` *returns a JSON array :*

```
[
  {
    "id":1,"first_name":"User","last_name":"comb1",
    "email":"user@comb1.fr","password":"passpass",
    "kind":{
      "name":"USER"
     },
     "organization":{
       "id":1,"name":"Combing Mill  1","country":"FR",
       "kind":{
         "name":"COMBINGMILL"
       }
     }
  },
  {
    "id":2,"first_name":"User","last_name":"spin1",
    "email":"user@spin1.es","password":"passpass",
    "kind":{
      "name":"USER"
    },
    "organization":{
      "id":2,"name":"Spinnning Mill 1","country":"ES",
      "kind":{
        "name":"SPINNINGMILL"
      }
    }
  }
]
```

- **POST** (create a new user)

1.2.4. User

Endpoints	Prefix	URL	description
user	GET	/users/:id(.:format)	get a user by id
	PATCH	/users/:id(.:format)	update a user by id
	PUT	/users/:id(.:format)	replace a user data, by id.
	DELETE	/users/:id(.:format)	delete a user, by id

Buisiness rules :

A user organization and his kind are always embedded in the responses (cf. examples)

Examples :

- **GET** (a user by id) :

  example : `curl http://localhost:3000/users/1` *returns a JSON*:

  ```
  {
    "id":1,"first_name":"User","last_name":"comb1",
    "email":"user@comb1.fr","password":"passpass",
    "kind":{
      "name":"USER"
     },
     "organization":{
       "id":1,"name":"Combing Mill  1","country":"FR",
       "kind":{
         "name":"COMBINGMILL"
       }
     }
  }
  ```

- **PATCH**
- **PUT**
- **DELETE**

1.2.5. Traces

Endpoints	Prefix	URL	description
traces	GET	/traces(.:format)	get all traces
	POST	/traces(.:format)	create a new trace

Business rules

- A user whose kind is ADMIN has access to all traces
- All other users only have access to the traces belonging to the organization they belong to.

Examples :

- **GET** (all traces)

  example : `curl http://localhost:3000/traces -H "uid: 1"` *returns the* `traces` *belonging to the organization of the uid (user id = 1) in the form of the following JSON :*

```
[
  {
    "id":1,"kind_id":4,"organization_id":1,
    "organization_name":"Combing Mill 1","user_id":1,
    "wool_top_lot_number":"WTL_01","wool_top_net_weight":1000,
    "wool_top_gross_weight":1010
  },
  {
    "id":2,"kind_id":5,"organization_id":1,
    "organization_name":"Combing Mill 1",
    "user_id":3,"wool_top_lot_number":"WTL_01",
    "wool_top_net_weight":1000,"wool_top_gross_weight":1010
  }
]
```

- **POST** :

1.2.7. Trace

Endpoints	Prefix	URL	description
trace	GET	/traces/:id(.:format)	get a trace by id
	PATCH	/traces/:id(.:format)	update a trace, by id
	PUT	/traces/:id(.:format)	replace a trace, by id
	DELETE	/traces/:id(.:format)	delete a trace, by id

Business rules :

- A user who's kind is ADMIN has access to all traces
- All other users only have access to the traces belonging to the organization theyxxx belong to.

Examples :

- **GET** (a trace by id)

  `curl 'http://localhost:3000/traces/1'` *returns the following JSON :*

  ```
  {
    "id":1,
    "kind_id":4,
    "organization_id":1,
    "organization_name":"Combing Mill 1",
    "user_id":1,
    "wool_top_lot_number":"WTL_01",
    "wool_top_net_weight":1000,
    "wool_top_gross_weight":1010,
    "created_at":"2018-09-28T12:34:50.831Z",
    "updated_at":"2018-09-28T12:34:50.831Z"
  }
  ```

1.2.8. Organizations

Endpoints	Prefix	URL	description
organizations	GET	/organizations(.:format)	get all organizations
	POST	/organizations(.:format)	create a new organization

Business rules :

none

Examples :

- **GET**

  `curl http://localhost:3000/organizations` returns

```
[

  { "id":1,"name":"Combing Mill  1","country":"FR",
    "kind":{"name":"COMBINGMILL"}
  },

  { "id":2,"name":"Spinnning Mill 1","country":"ES",
    "kind":{"name":"SPINNINGMILL"}
  },

  {
    "id":3,"name":"crystalchain","country":"FR",
    "kind":{"name":"ADMIN"}
  }
]
```

- **POST**

1.2.8. Organization

Endpoints	Prefix	URL	description
organization	GET	/organizations/:id(.:format)	get an organization, by id
	PATCH	/organizations/:id(.:format)	update an organization, by id
	PUT	/organizations/:id(.:format)	replace an organization, by id
	DELETE	/organizations/:id(.:format)	delete an organization, by id

Business rules :

None specified

Examples :

- **GET**

  `curl http://localhost:3000/organizations/1`

  ```
  {
    "id":1,"name":"Combing Mill  1","country":"FR",
    "kind":{"name":"COMBINGMILL"}
  }
  ```

- **PATCH**
- **PUT**
- **DELETE**