Budget Pro Project

Symfony 4 API REST Project

__Bank Pro__ is a finance management api application for everyone. A __User__ is linked to a __Subscription__ and can have zero or infinite number of credit __Cards__. At the registration, the __User__ is in the __obligation__ to inform the selected __Subscription__ (created by an __Admin__)

Models

User
id, firstname (optional), lastname (optional), email (required and uniq), apiKey (generated and uniq), createdAt (Datetime generated), address (optional), country (optional), subscription (User is linked to one Subscription), cards (User can have many cards (optional))

Subscription
id, name (required), slogan (required), url (optional), users (Subscription have many Users)

Card
id, name (required), creditCardType (required), creditCardNumber (required), currencyCode (required, can be for example : EUR, USD, GBP...), value (required, between 0 and 100 000), user (Card is linked to one User)

Rules

TIPS : Resources to CRUD [User, Subscription, Card]

As ANONYMOUS :
- Create a User (and optionally cards at the same time)
- Get One or All Subscriptions
- Get One or All User (but with another serialization : User#**firstname**, User#**email**, Card#**name** and full Subscription informations

As ROLE_USER :
- Get my User information (profile, with full serialization information)
- Edit your User firstname, lastname, address, country, cards, subscription
- GetOne, GetAll, Edit, Create or Delete one of your Cards

As ROLE_ADMIN (*same as ROLE_USER, plus*) :
- User
  - List all Users; Get one User, Edit any User; Delete any User (*with his Cards*)
- Subscription
  - Create, GetOne, GetAll, Edit, Delete
  - Do not allow to delete a Subscription if there is at least one User linked. Make sure there is an error.
- Card
  - Create, GetOne, GetAll, Edit, Delete

You will use :

- Fixtures on every resources possible (with Faker random data library)
- Symfony Command to create an Admin User
- Symfony Command to display the number of Card by User email
- Symfony Validation (@Assert) on every resources possible
- Error Management (with ValidatorInterface or ConstraintViolationListInterface)
- Correct HTTP Status Code (201 Created, 204 No Content, 400 Bad Request, 403 Forbidden…)
- Swagger API Doc (working as possible)
- Tests with PHPUnit (for every multiple successful cases and errors cases)
- Your code will be hosted on GitHub, with Travis CI connected to run your PHPUnit tests

Nota Bene : Code must be hosted on GitHub. Comment your code! Even though it's not a complete feature. Please document your README.md file. Use your common sense and knowledge to prioritise. Good luck !