Documentation Employee track app

APIs:

POST:

1. ***/employee***

Is used to add employees to a company.

Request contains a body that should be provided for the success of adding operation.

URL: http://****:3000/employee;
Method: ***POST***;
Type of response: ***String****;*
Response value for success : ***"Employee is added!"***;
Validation rules:

*field ***age*** : value > ***18***;
*field ***diploma*** : must have at least one value;

Response for validation(unsuccessful) : **"*Employee must be at least 18 years old or more to be added!*" or "*Employee must have at least one diploma to be added!*"**;
Request body:

{
"first_name": <string>,
"last_name": <string>,
"age": <number>,
"telephone": <string>,
"diploma": <Vec<string>>
}

**Vec* = *vector* same meaning as list;
Request body: example is provided on fig 1;

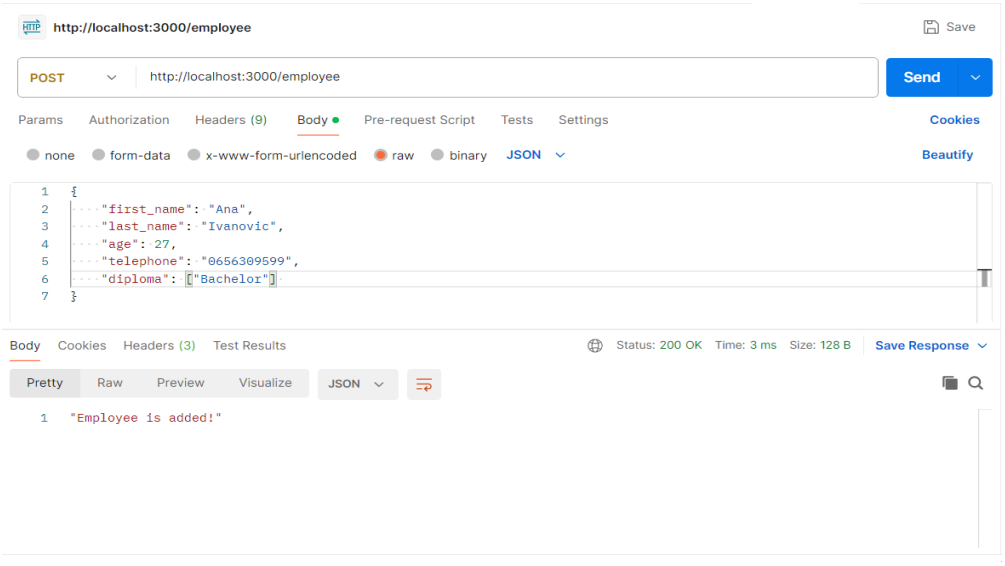

Fig 1. Request body for adding employee

After adding employees, the state of our employee base can be seen on fig 2.

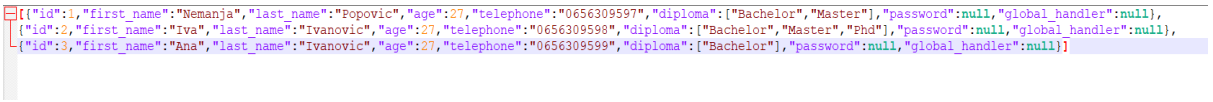

Fig 2. employee base

For this request we have validations and in that validation we check two fields, ***age*** and ***diploma***.

Example of incorrect value for field ***age*** can be seen on fig 3 .

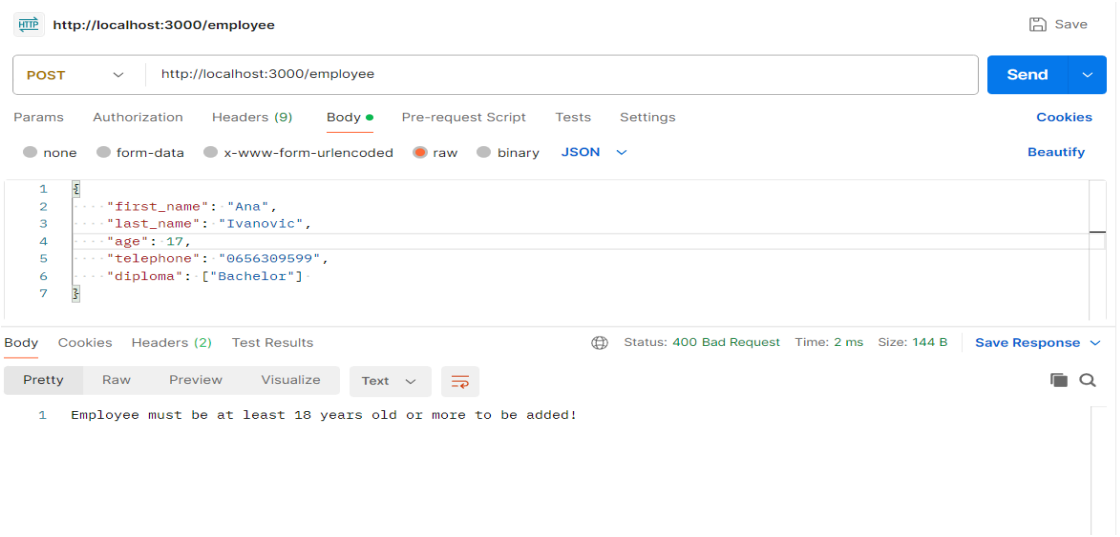

Fig 3. Incorrect value for age

Example of incorrect value for field ***diploma*** can be seen on fig 4.

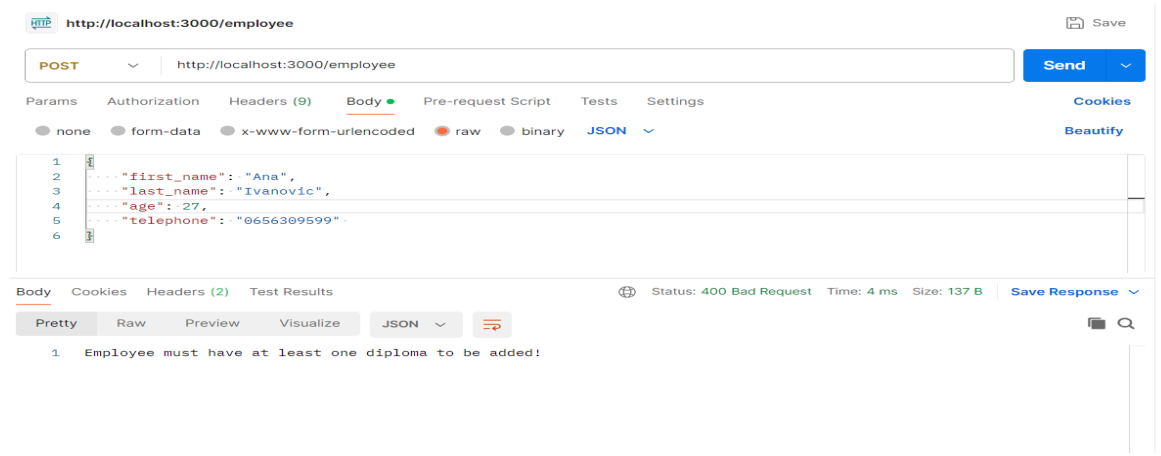

Fig 4 Incorecct value for diploma

GET:

2. ***/notonborded***

Is used to get information about employees who are not onboarded.

Request does not contain any parameters that should be provided for execution.

URL: http://****:3000/notonboarded;
Method: ***GET***;
Type of response: ***Vec<i64>***;
**Vec* = *vector* same meaning as list;
RESPONSE: is provided as a vector of integers(fig 5);

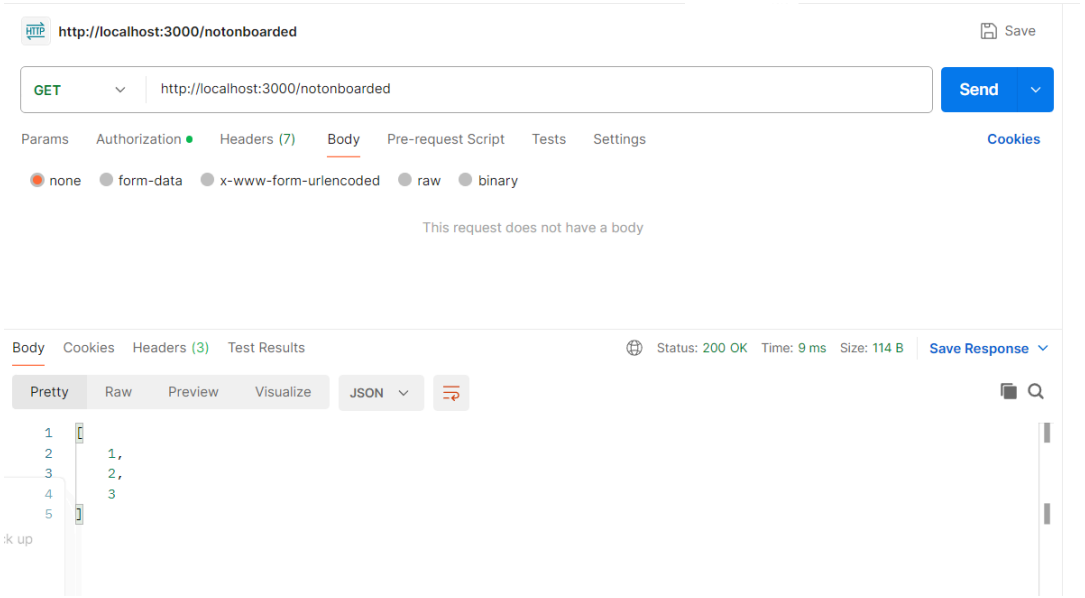

Fig 5. employees to be onboarded

PATCH:

3. ***/onboarding/:id***

Is used to set fields ***password*** and ***global_handling*** on request of an IT person.

For executing this request it is needed to provide employee ***id*** as a *path* variable.

URL: http://****:3000/onboarding/:id;
Method: ***PATCH***;
Type of response: ***String***;
Path variable: *"id": <number>*;

Example of successful request on fig 6.

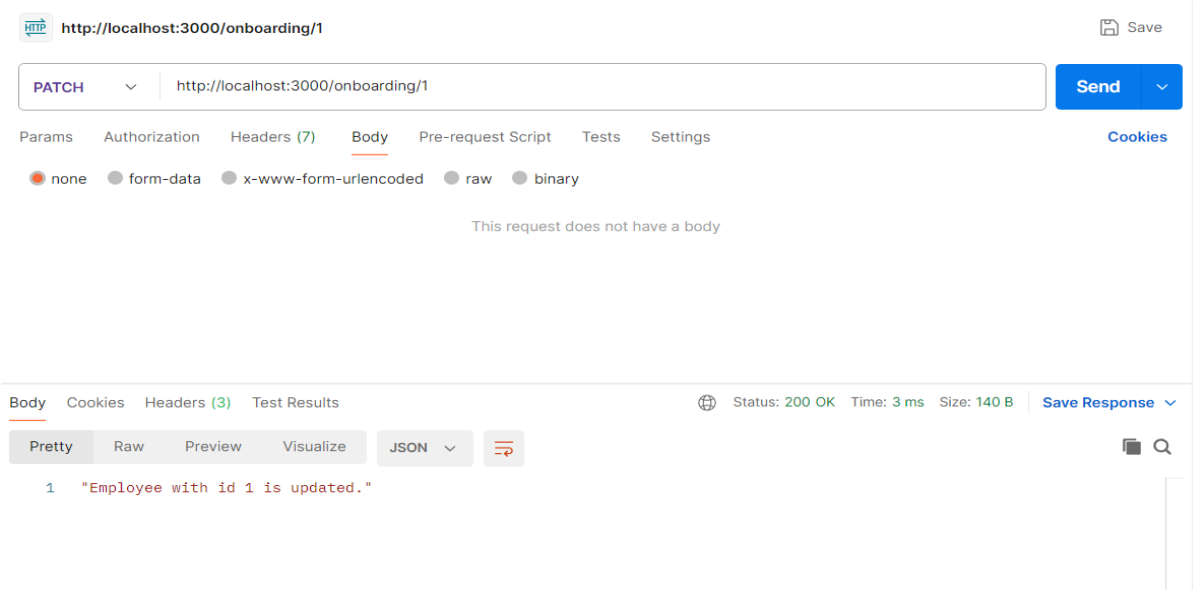

Fig 6. patch successful

Result of this request with the state in the employee base can be seen on fig 7.

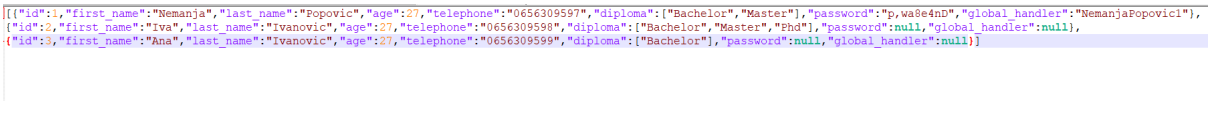

Fig 7 employee base state after patch

If there is no user with ***id*** response is "***There is no employee with {id}***" (fig 8).

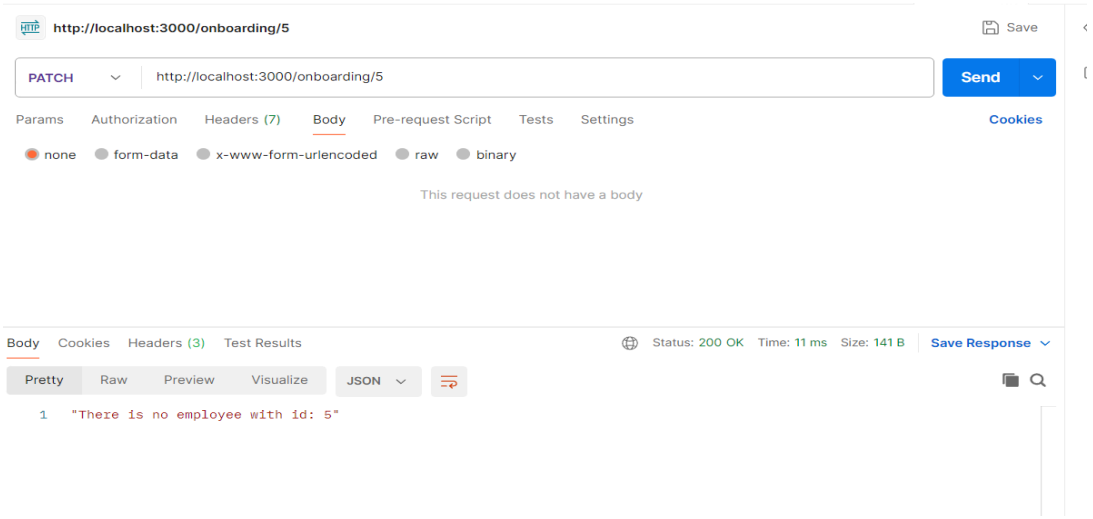

Fig 8 wrong id for patch

GET:

4. ***/details***

Is used to get information about employees after they are onboarded. This request has authentication so for this request it is important to provide the password of that employee.

URL: http://****:3000/details;
Method: ***GET***;
Type of response: ***Json<Employee>***;
Employee:
{
"first_name": <string>,
"last_name": <string>,
"age": <number>,
"telephone": <string>,
"diploma": <Vec<string>>,
"password": <string>,
"global_handler": <string>

}
Authentication parameter: ***"password": <string>;***

Type of authentication: ***Basic authentication;***

Response if authentication is not selected can be seen on fig 9.

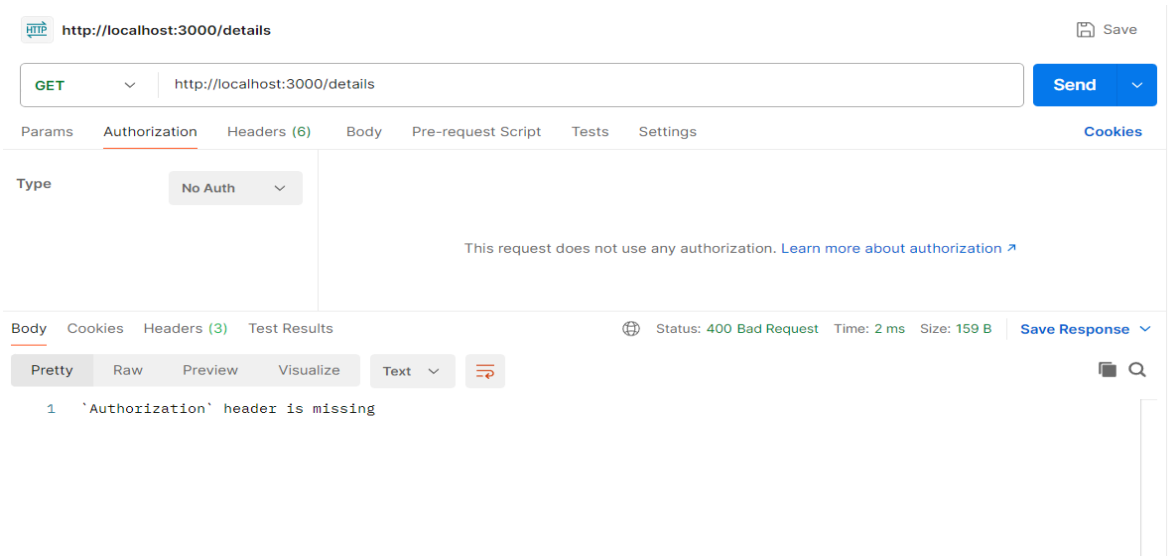

Fig 9. no authentication

Response if value for password is not id of employee can be seen on fig 10.

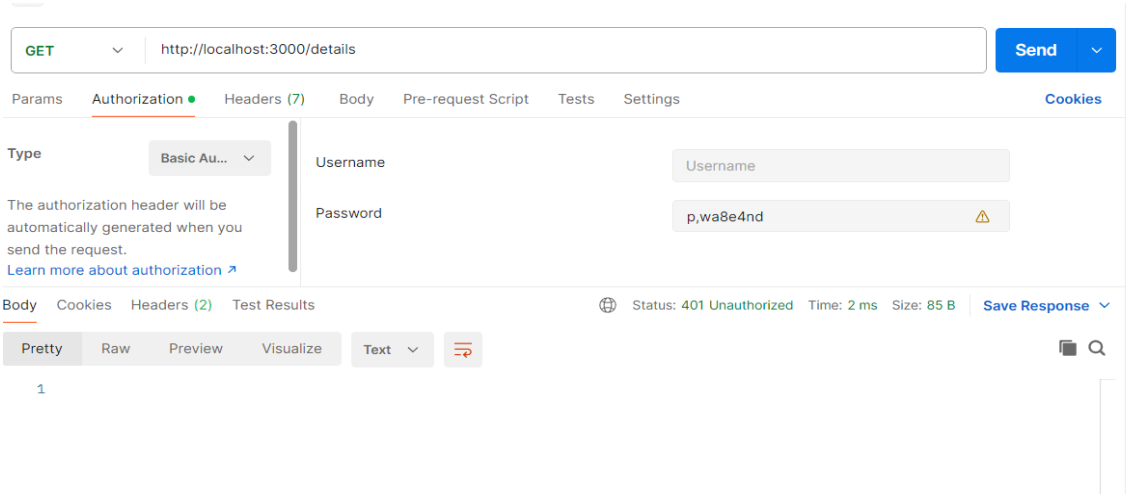

Fig 10. wrong id

Response if value for password is id of employee can be seen on fig 11.

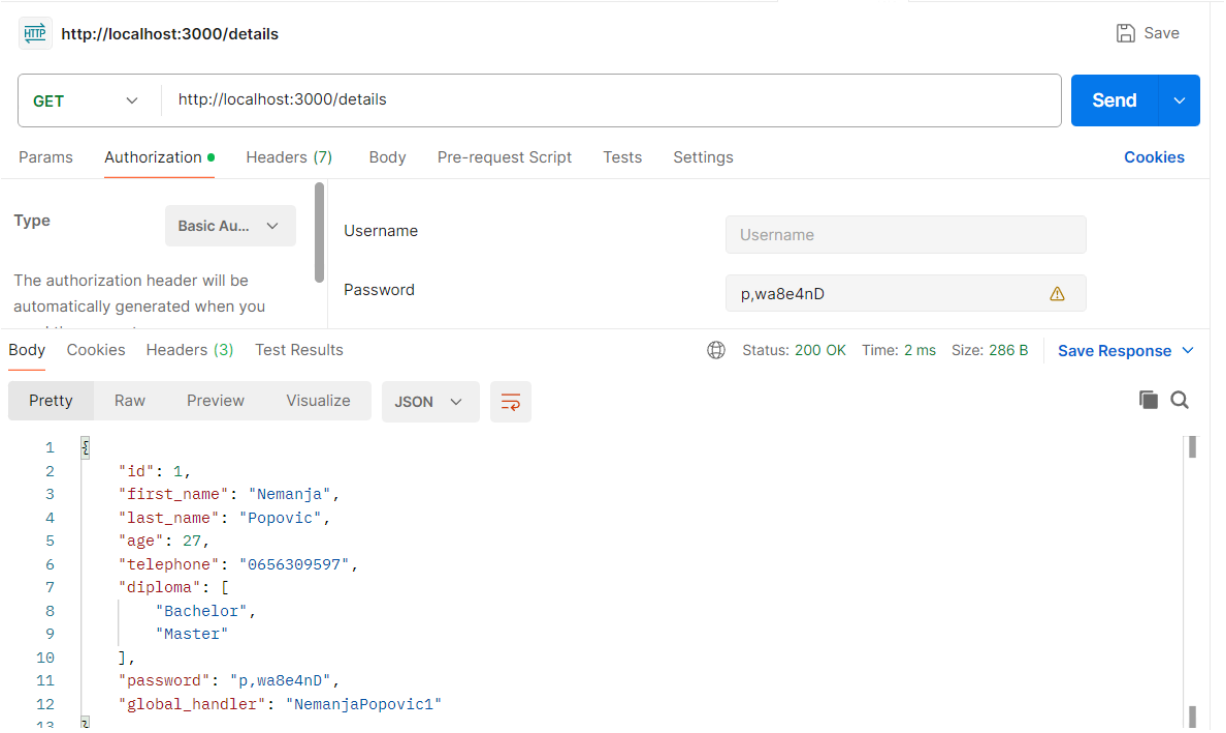

Fig 11 corect id

Some improvements/thoughts:

Instead of using a file as an employee base, there should be an implementation of a database so to get employees. In the specification of the task it is told not to add more dependencies, so in the given package example there is no library for the database and because of that I used a file.
If there is a database we could write queries and get data easily. In this example I pull all the data and then operate with them. So if there is an enormous amount of data this process can slow the program.

When we talk about negative cases there is room for improvements in terms of messages or status that need to be provided.

Also maybe to add more validation rules to get valid data.
About password there is no encoding and decoding, data is stored as plain text so that needs to be improved.