Quinto dia

- [x] ~~Adicionando validação~~

Abrir o Backend no Code

```
npm install celebrate
```

abrir arquivo Routes.js

```
const{ celebrate, Segments, Joi } = require('celebrate')

    ----


    Validar aqui:


    routes.post('/ongs', celebrate({
       [Segments.BODY]: Joi.object().keys({
           name: Joi.string().required(),
           email: Joi.string().email(),
           whatsapp: Joi.string().required().min(10).max(11),
           city: Joi.string().required(),
           uf: Joi.string().required().length(2)
       })
    }),ongController.create)



    routes.get('/profile', celebrate({
        [Segments.HEADERS]: Joi.object({
            authorization: Joi.string().required()
        }).unknown(),
    }),profileController.index)


    routes.get('/incidents', celebrate({
        [Segments.QUERY]: Joi.object().keys({
            page: Joi.number(),
        })
    }),incidentController.index)
```

ir na pasta database, arquivo index:

```
const { errors } = require('celebrate')

    ----


    app.use(errors())
```

- [ ] Adicionando testes
    - [x] ~~Por que fazer testes?~~

Porque uma alteração pode afetar várias outras partes da aplicação

- [x] ~~TDD(Test-Driven Development)~~

Quando você cria o teste antes mesmo de terminar a funcionalidade, e desenvolve para que a aplicação passe no teste

- [x] ~~Configurando Jest~~

Pasta Backend

```
npm install jest -D

    npx jest --init
```

Criar pasta tests dentro do backend e dentro dela as pastas Unit e Integration

- [x] ~~Tipos de testes~~

Unitário - testa uma parte isolada

Integração - "esbarra" em vários caminhos, testando como um todo

Na pasta SRC, criar uma pasta Utils

Dentro dela, criar aquivo genereteUniqueId.js

```
const crypto = require('crypto')

    module.exports = function generateUniqueId(){
        return crypto.randomBytes(4).toString('HEX')
    }
```

E no OngController, importar esse arquivo e implementar essa function

```
const generateUniqueId = require('../utils/generateUniqueId')

    ---

    const id = generateUniqueId()
```

Dentro ta pasta Unit em Tests

Criar arquivo generateUniqueId.spec.js

```
const generateUniqueId =  require("../../src/utils/generateUniqueId")

    describe('Generate Unique ID', ()=> {
        it('should generate an unique ID', () =>{
            const id = generateUniqueId()

            expect(id).toHaveLength(8)
```

```
    })
})
```

E no terminal

```
npm test
```

- [x] ~~Configurando banco de testes~~

Abrir knexfile.js

Criar mais um bloco para testes (Esse é cópia do development, só alterado nome e nome do db)

```
test: {
      client: 'sqlite3',
      connection: {
        filename: './src/database/test.sqlite'
      },
      migrations: {
        directory: './src/database/migrations'
      },
      useNullAsDefault: true,
    },
```

dentro da backend

```
npm install cross-env
```

no arquivo package.json

```
{

    "name": "backend",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "scripts": {
      "start": "nodemon src/index.js",
      "test": "cross-env NODE_ENV=test jest"
    }
```

Na pasta de Migrations/connection.js

```
const knex = require('knex')
    const configuration = require('../../knexfile')

    const config = process.env.NODE_ENV === 'test' ? configuration.test : configuration.development

    const connection = knex(config)
```

```
module.exports = connection;
```

Na pasta tests/integration criar arquivo ong.spec.js

```
describe('ONG', () => {
        it('should be able to create a new ONG', () => {

        })
    })
```

- [x] ~~Instalando supertest~~

```
npm install supertest -D
```

dentro da pasta SRC, criar arquivo server.js

Trocar o nome do arquivo index.js por app.js

no arquivo package.json trocar de index para server

```
{
    "name": "backend",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "scripts": {
      "start": "nodemon src/server.js",  // <----- HERE
      "test": "cross-env NODE_ENV=test jest"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "dependencies": {
      "celebrate": "^12.0.1",
      "cors": "^2.8.5",
      "cross-env": "^7.0.2",
      "express": "^4.17.1",
      "knex": "^0.20.13",
      "sqlite3": "^4.1.1"
    },
    "devDependencies": {
      "nodemon": "^2.0.2",
      "jest": "^25.2.3",
      "supertest": "^4.0.2"
    }
  }
```

No arquivo app.js

Trocar

```
app.listen(3333);

    PARA

    module.exports = app
```

Dentro do arquivo server.js

```
const app = require('./app')

    app.listen(3333)
```

No arquivo ong.spec.js

```
const request = require('supertest')
    const app = require('../../src/app')
    const connection = require('../../src/database/connection')

    describe('ONG', () => {

        beforeEach(async ()=> {
            await connection.migrate.latest()
        })

        afterAll(async()=>{
            await connection.destroy()
        })

        it('should be able to create a new ONG', async () => {
            const response = await request(app)
            .post('/ongs')
            .send({
                name: "APAD2",
                email: "contato@ong.com",
                whatsapp: "4700000000",
                city: "sao paulo",
                uf: "SP"
            })

            expect(response.body).toHaveProperty('id')
            expect(response.body.id).toHaveLength(8)

        })
    })
```

Para executar o teste é só fazer o npm test

Para criar um teste que tenha o header de autorização:

```
it('should be able to create a new ONG', async () => {
            const response = await request(app)

            .set('Authorization', 'a4b3c1af') // <------ HERE

            .post('/ongs')
            .send({
                name: "APAD2",
                email: "contato@ong.com",
                whatsapp: "4700000000",
                city: "sao paulo",
                uf: "SP"
            })

            expect(response.body).toHaveProperty('id')
            expect(response.body.id).toHaveLength(8)

        })
```

- [x] ~~Testando rota de autenticação~~

- [x] ~~Deploy~~
  - [x] ~~Alternativas~~

  (NODE) Para mostrar para amigos - Heroku

  (NODE) Comercial - DigitalOcean

  (WEB) App pequeno, especifico pra FRONT END - Netlify

  - [x] ~~Qual devo escolher~~

- [x] ~~Estudos daqui pra frente~~
  - [x] ~~Padrões de código: ESLint, Prettier~~
  - [x] ~~Autenticação JWT -> Segurança de login (exemplo)~~
  - [x] ~~Styled Componentes~~

- [x] ~~Dicas para aproveitar melhor~~
  - [x] ~~GitHub~~
  - [x] ~~LinkdIn~~