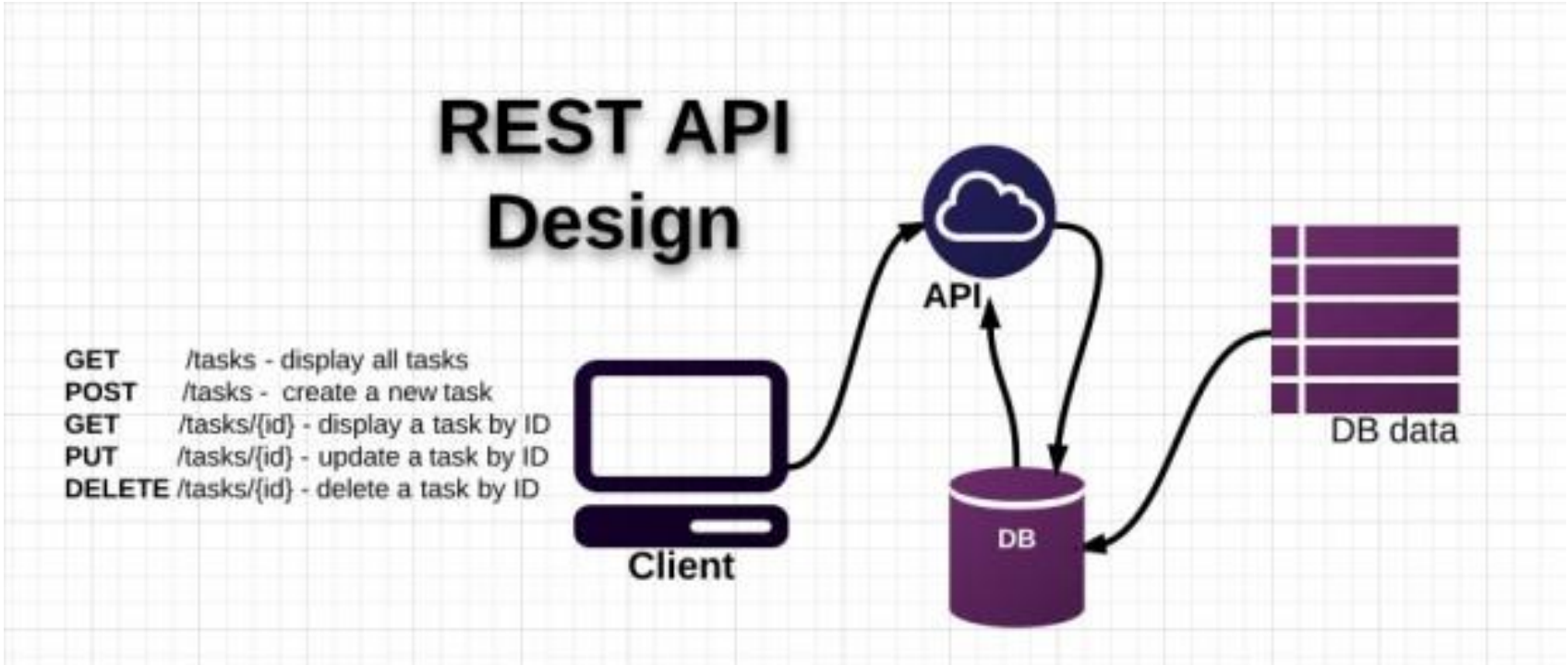
Defenda seus consumidores
Use testes de contrato!

Marcelo Serpa

Olá, sou Marcelo Serpa :)

- Desenvolvedor de software na Ilegra
- Formando em ADS na FTEC
- Apaixonado por desenvolvimento de software
- Além de programação, gosto de card games
- https://github.com/marceloserpa
- https://twitter.com/_marceloserpa

O que são APIs?



Vantagens

- Plugar vários consumidores
- Regra centralizada
- Omnichannel
- Simplicidade de integração
- Desenvolvimento paralelizado

Exemplo

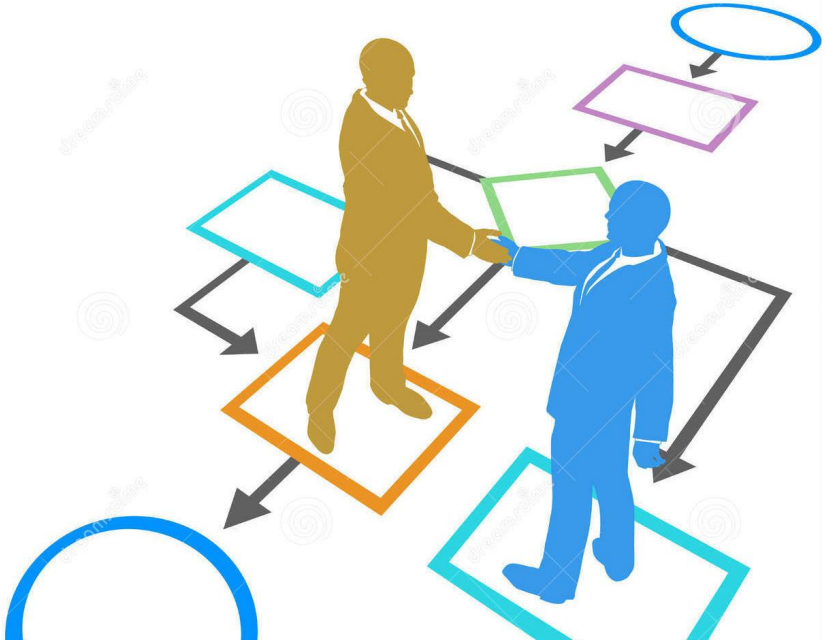
GET - http://localhost:3000/note/v1/notes


```
[
  {
    "title": "Prova de redes",
    "description": "A prova será na sala 305"
  },
  {
    "title": "Aula de Ingles",
    "description": "Tentar conversação livre"
  },
  {
    "title": "Ler o livro X",
    "description": "continuar no capitulo 6"
  }
]
```

O que é contrato?

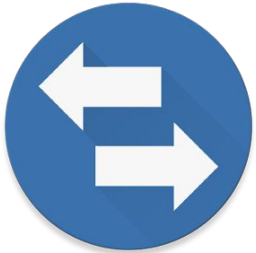
Componentes do contrato

- url
- método http
- body (request, response)
- tipos
- headers
- formato
- exceptions
- validações



Mas contratos podem quebrar...

Exemplo 1 - refatorando



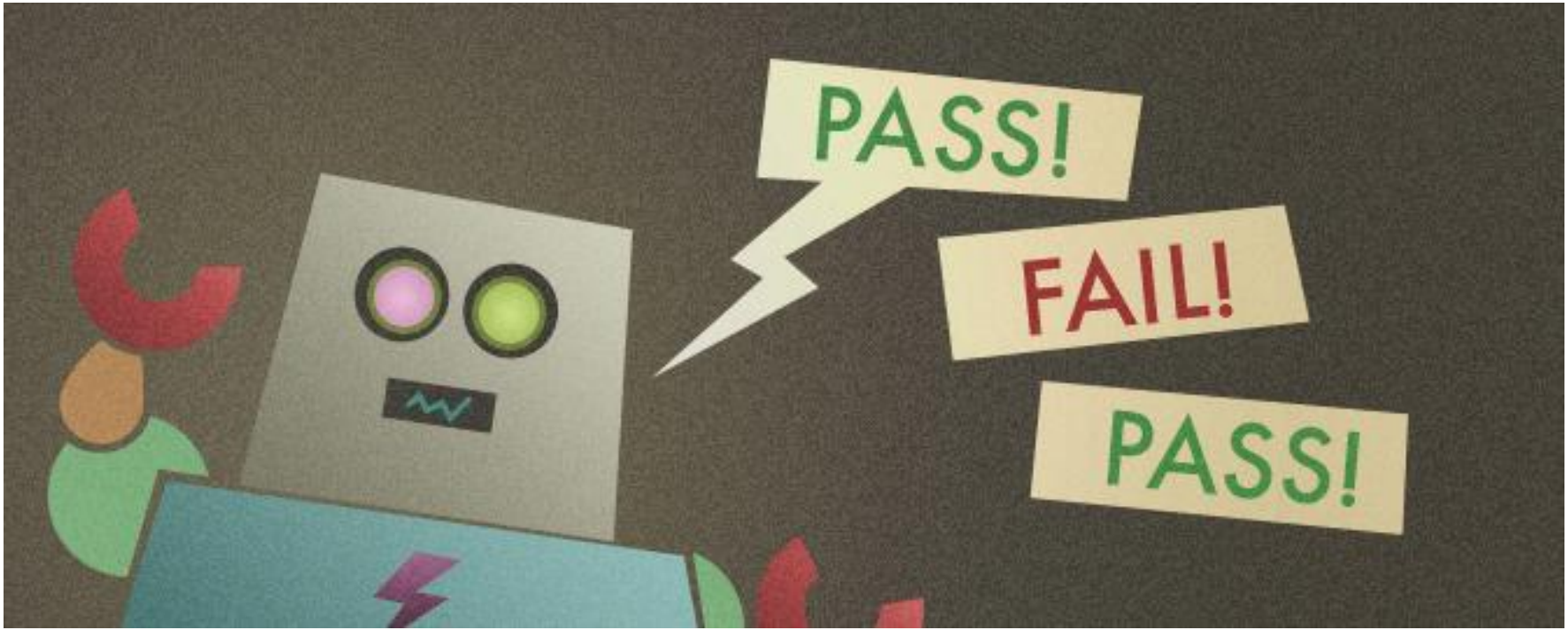
Exemplo 2 - mudança de formato

```
{
    “dataNascimento”: “1990-12-19”
}
```

```
{
    “dataNascimento”: “19/12/1990”
}
```

Como podemos garantir que o contrato não foi quebrado?

Testes de contrato



Minha aplicação de exemplo - Stack

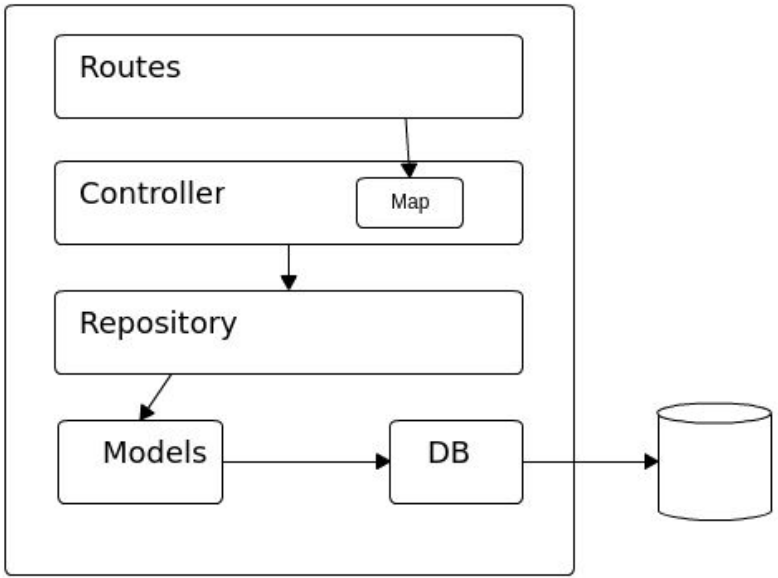
Aplicação:

- Node
- Mongo
- Express
- Express validator
- Mongoose

Testes:

- Mocha
- Chai
- Supertest
- Mockery

Minha aplicação de exemplo - estrutura



Configurando a aplicação

```
1  var express = require("express");
2  var expressValidator = require('express-validator');
3  var bodyParser= require('body-parser');
4
5  var routes = require('./routes');
6
7  var app = express();
8  app.use(bodyParser.json());
9  app.use(bodyParser.urlencoded({extended: true}))
10 app.use(expressValidator());
11 app.use("/", routes);
12
13 module.exports = app;
```

Definindo rotas

```
1  var express = require('express');
2  var router = express.Router();
3
4  var Note = require('../models/note_model');
5  var NoteRepository = require('../repository/notes_repository')(Note);
6  var NoteController = require('../controllers/notes_controller')(NoteRepository);
7
8  router.route('/note/v1/notes/')
9      .post(NoteController.post)
10     .get(NoteController.get);
11
12 module.exports = router
```

Manipulando a request e aplicando validações

```
4  var NoteController = (NoteRepository) => {
5      return {
6          post: (req, res) => {
7              req.assert('title', 'required').notEmpty();
8              req.assert('description', 'required').notEmpty();
9              req.getValidationResult().then(saveNote);
10
11             function saveNote(result){
12                 if (result.isEmpty()) {
13                     NoteRepository.save(req.body)
14                         .then(document => res.send("Ok"))
15                         .catch(err => console.log(err));
16                 } else {
17                     res.status(400).send(result.array());
18                 }
19             };
20         },
```

Repository

```
2  var NoteRepository = (NoteSchema) => {
3      return {
4          save: (newNote) => {
5              const note = new NoteSchema(newNote);
6              return note.save();
7          },
8          findAll: () => {
9              return NoteSchema.find().sort({created_at: -1});
10         }
11     }
12 };
13
14 module.exports = NoteRepository;
```

Definindo Schema

```
2  var mongoose = require("../db");
3
4  var Schema = mongoose.Schema;
5
6  var NoteSchema = new Schema({
7    title: String,
8    description: String,
9    created_at: { type: Date, default: Date.now }
10 });
11
12 const Note = mongoose.model('Note', NoteSchema);
13
14 module.exports = Note;
```

Configurando a conexão com o Mongo

```
2  var mongoose = require("mongoose");
3
4  mongoose.connect('mongodb://localhost:27017/notes_app');
5
6  mongoose.Promise = global.Promise;
7
8  module.exports = mongoose;
```

Testando status HTTP em caso de sucesso

```
describe('POST /note/v1/notes/', () => {
  it('success case', done => {
    var app = require('../src/server');

    const note = {title: 'title ok', description: 'description ok'}
    request(app)
      .post('/note/v1/notes/')
      .send(note)
      .expect(200, done);
  });
```

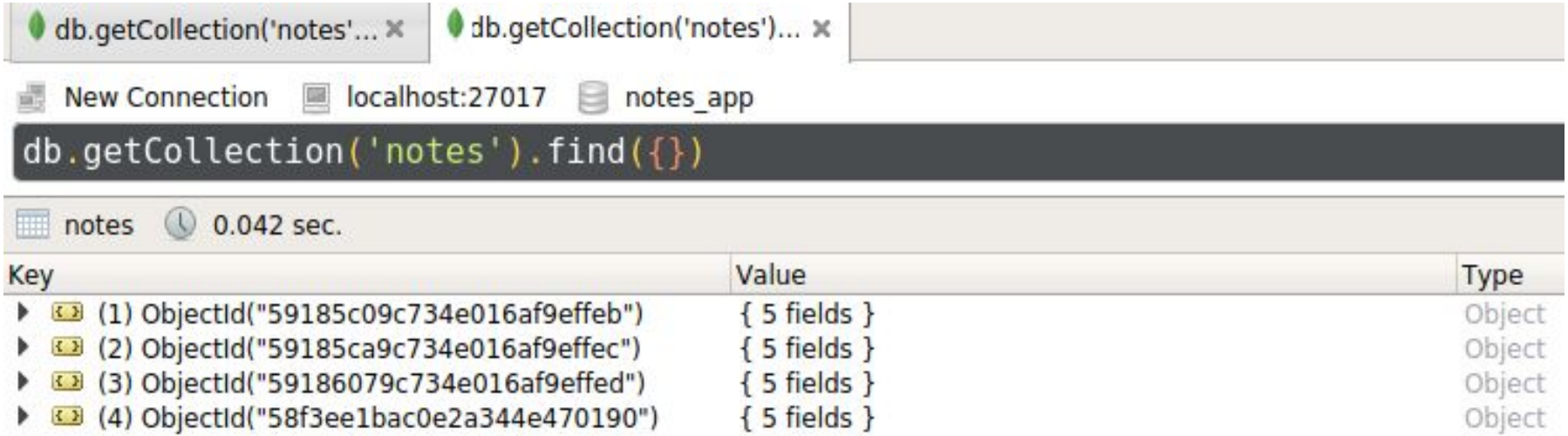
Testando status HTTP em caso de erro

```
it('title is required field', done => {
  var app = require('../src/server');

  const note = {description: 'description ok'}
  request(app)
    .post('/note/v1/notes/')
    .send(note)
    .expect(res => {
      res.body.lenght = 1;
      res.body[0].param = 'title';
      res.body[0].msg = 'required';
    })
    .expect(400, done);
});
```

```
Notes
  GET /note/v1/notes/
    ✓ respond with success (932ms)
  POST /note/v1/notes/
    ✓ success case (45ms)
    ✓ title is required field
    ✓ description is required field
```

Problema: está conectado no banco real

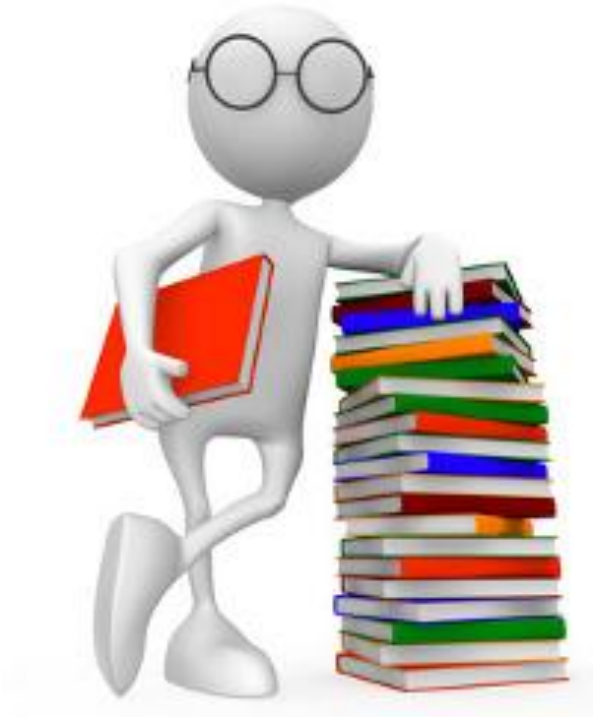


Mockery

```
8       before('init mocks', () => {
9           var mock = (NoteSchema) => {
10              return {
11                  save: (newNote) => Promise.resolve(),
12                  findAll: () => {
13                    return Promise.resolve([  {
14                      "_id": "58f3ee1bac0e2a344e470190",
15                      "title": "title test",
16                      "description": "lorem",
17                      "__v": 0,
18                      "created_at": "2017-05-30T00:00:00.000Z"
19                    }])
20                  }
21              }
22          }
23          mockery.registerMock('../repository/notes_repository', mock);
24          mockery.enable({
25              warnOnUnregistered: false
26          });
27      })
```

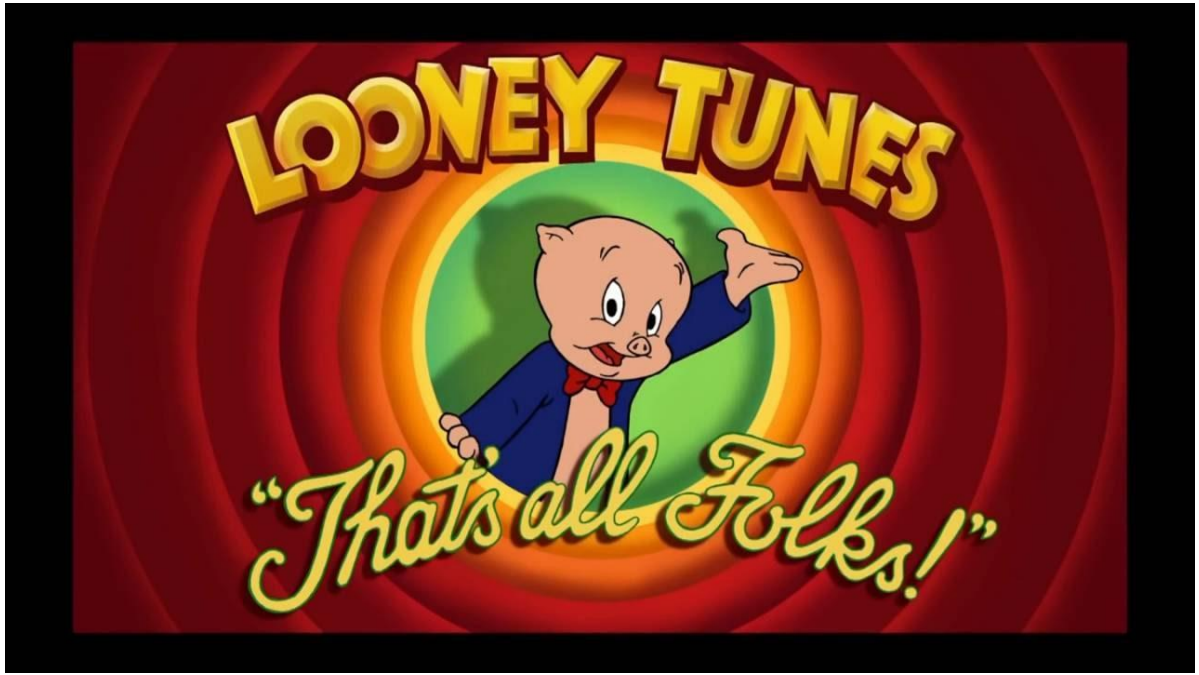
Outros tópicos importantes:

- Versionamento de contrato
- Contract-first
- Anti-corruption layer
- Testes com containers
- Proxyquire



Concluindo...

- Boa noite de sono
- Segurança
- Feedback imediato
- Contract-first
- Documentar sua API
- Conhecer sua API



https://github.com/marceloserpa/contract_test_nodejs

https://twitter.com/_marceloserpa