Curso Mongo, Express, Angular e Node - Primeira Aplicação do ZERO!

Leonardo Leitão

Versão 1.0, 10/05/2017

Índice

Sumário

Apostila do curso de Mongo, Express, Angular e Node - Primeira Aplicação do ZERO! da Cod3r.

https://www.cod3r.com.br

1. Introdução

1.1. Bem-vindo

1.2. Visão Geral do Curso

1.3. Conhecendo o Projeto

1.4. Assine o Nosso Canal

Olá amigos, convido vocês a conhecer o canal da COD3R no Youtube.

Visite o canal: https://www.youtube.com/aulasdeprogramacao

1.5. Repositório do Curso

Os arquivos do curso estão disponíveis no GitHub.

Repositório do curso: https://github.com/CursosCod3r/mean-primeira-app

1.6. Instalação MongoDB (Windows)

1.7. Instalação Node (Windows)

2. Mongo

2.1. Mongo - Visão Geral

2.2. Mongo - Exercício 01: Comandos Básicos

Abra o terminal e execute o seguinte comando:

Listagem 1 - *Inicializar o servidor do Mongo*

```
mongod
```

Para iniciar o `console do Mongo` abra um novo terminal.

Listagem 2 - *Iniciar o console do Mongo*

```
mongo
```

Listagem 3 - *Listar todas as databases*

```
show dbs
```

Listagem 4 - *Usar database db_finance*

```
use db_finance
```

Listagem 5 - *Verificar o database atual*

```
db
```

Listagem 6 - *Listar todas as databases*

```
show dbs
```

Listagem 7 - *Criar Collection billingCycles*

```
db.createCollection('billingCycles')
```

Listagem 8 - *Listar todas as databases*

```
show dbs
```

Listagem 9 - *Listar as Collections*

```
show collections
```

Listagem 10 - *Criar Collection billingcycles*

```
db.createCollection('billingcycles')
```

Listagem 11 - *Listar as Collections*

```
show collections
```

Listagem 12 - *Remover a Collection billingCycles*

```
db.billingCycles.drop()
```

Listagem 13 - *Listar as Collections*

```
show collections
```

Listagem 14 - *Remover a Collection billingcycles*

```
db.billingcycles.drop()
```

Listagem 15 - *Listar as Collections*

```
show collections
```

Listagem 16 - *Listar todas as databases*

```
show dbs
```

2.3. Mongo - Exercício 02: Inserções

Listagem 17 - *Limpar o console*

```
cls
```

Listagem 18 - *Listar todas as databases*

```
show dbs
```

Listagem 19 - *Usar database db_finance*

```
use db_finance
```

Listagem 20 - *Listar as Collections*

```
show collections
```

Listagem 21 - *Inserção na Collection billingcycles*

```
db.billingcycles.insert({name:"Janeiro/17", month: 1, year: 2017})
```

Listagem 22 - *Listar as Collections*

```
show collections
```

Listagem 23 - *Listar todas as databases*

```
show dbs
```

Listagem 24 - *Salvar na Collection billingcycles*

```
db.billingcycles.save({name:"Fevereiro/17", month: 2, year: 2017})
```

Listagem 25 - *Inserção na Collection billingcycles*

```
db.billingcycles.insert({
name:"Março/17",
month: 3,
year: 2017,
credits: [
{name: "Salário", value: 5000}
],
debts: [
{name: "Luz", value: 100, status: "PAGO"}
{name: "Telefone", value: 100, status: "PENDENTE"}
]
})
```

2.4. Mongo - Exercício 03: Consultas

Listagem 26 - *Limpar o console*

```
cls
```

Listagem 27 - *Usar database db_finance*

```
use db_finance
```

Listagem 28 - *Listar todos os registros da Collection*

```
db.billingcycles.find()
```

Listagem 29 - *Listar os registros da Collection com uma melhor leitura*

```
db.billingcycles.find().pretty()
```

Listagem 30 - *Listar apenas um registro da Collection*

```
db.billingcycles.findOne()
```

Listagem 31 - *Listar apenas um registro da Collection que contém mês 2*

```
db.billingcycles.findOne({month: 2})
```

Listagem 32 - *Listar apenas um registro da Collection que contém mês 1 ou mês 2 com uma melhor leitura*

```
db.billingcycles.findOne({$or: [{month: 1}, {month: 2}]}).pretty()
```

Listagem 33 - *Listar os registros da Collection com o atributo credits com uma melhor leitura*

```
db.billingcycles.find({credits:{$exists:true}}).pretty()
```

Listagem 34 - *Listar os registros da Collection que contém ano 2017*

```
db.billingcycles.findOne({year: 2017})
```

Listagem 35 - *Listar os registros da Collection menos o primeiro registro*

```
db.billingcycles.findOne({year: 2017}).skip(1)
```

Listagem 36 - *Listar apenas um registros da Collection menos o primeiro registro*

```
db.billingcycles.findOne({year: 2017}).skip(1).limit(1)
```

2.5. Mongo - Exercício 04: Agregação

Listagem 37 - *Criar fluxo do pipeline para somatória de credits e debts*

```
db.billingcycles.aggregate([{
$project:{credit:{$sum: "$credits.value"},debt:{$sum: "$debts.value"}}
},{
$group:{_id:null,credit:{$sum: "$credit"},debt:{$sum: "$debt"}}
}])
```

Listagem 38 - *Listar apenas um registro da Collection que contém mês 3*

```
db.billingcycles.findOne({month: 3})
```

2.6. Mongo - Exercício 05: Atualização

Listagem 39 - *Limpar o console*

```
cls
```

Listagem 40 - *Atualizar o registro da collection*

```
db.billingcycles.update(
{$and:[{month: 1},{year:2017}]},
{$set:{credits:[{name:"Salário",value:5000}]}}
)
```

Listagem 41 - *Listar apenas um registro da Collection*

```
db.billingcycles.findOne()
```

Listagem 42 - *Listar os registros da Collection com atributo credits e retornar o atributo name com uma melhor leitura*

```
db.billingcycles.find({credits:{$exists:true}}, {_id:0, name: 1}).pretty()
```

2.7. Mongo - Exercício 06: Contador E Remoções

Listagem 43 - *Limpar o console*

```
cls
```

Listagem 44 - *Contar quantos registros tem a Collection*

```
db.billingcycles.count()
```

Listagem 45 - *Remover os registros da Collection que contém mês 2*

```
db.billingcycles.remove({month: 2})
```

Listagem 46 - *Contar quantos registros tem a Collection*

```
db.billingcycles.count()
```

Listagem 47 - *Remover apenas um registros da Collection que contém ano 2017*

```
db.billingcycles.remove({year: 2017}, 1)
```

Listagem 48 - *Contar quantos registros tem a Collection*

```
db.billingcycles.count()
```

Listagem 49 - *Remover a database db_finance*

```
db.dropDatabase()
```

Listagem 50 - *Listar todas as databases*

```
show dbs
-----
```

3. Node

3.1. Node - Exercício 01: Javascript Básico

Abra o terminal e dentro da pasta `Desktop` execute o seguinte comando:

Listagem 51 - *Criar pasta FundamentosMEAN*

`Desktop/FundamentosMEAN`

```
mkdir FundamentosMEAN && cd FundamentosMEAN
```

Listagem 52 - *Criar pasta node*

`FundamentosMEAN/node`

```
mkdir node && cd node
```

Listagem 53 - *Abrir o Atom*

```
atom .
```

Listagem 54 - *Criar arquivo ex01.js*

`FundamentosMEAN/node/ex01.js`

```
const ola = () => console.log('Ola node!')
setInterval(ola, 1000)
```

Abra o terminal e dentro da pasta `FundamentosMEAN/node` execute os seguintes comandos:

Listagem 55 - *Executar arquivo ex01.js*

```
node ex01
```

Listagem 56 - *Parar de executar o arquivo ex01.js*

```
ctrl + c
```

Existe um plugin do Atom chamado `atom-runner` para executar seus arquivos.

3.2. Node - Exercício 02: Sistema de Módulos

Listagem 57 - *Criar arquivo ex02_utils.js*

FundamentosMEAN/node/ex02_utils.js

```
function upper(text) {
    return text.toUpperCase()
}

module.exports = { upper }
```

Listagem 58 - *Criar arquivo ex02_teste.js*

FundamentosMEAN/node/ex02_teste.js

```
const utils = require('./ex02_utils')
console.log(utils.upper('show de bola!'))
```

Se você instalou o plugin `atom-runner` no Atom, basta está no documento e apertar `alt + r` para executar o arquivo `ex02_teste.js`.

3.3. Node - Exercício 03: Singleton

Listagem 59 - *Criar arquivo ex03_singleton.js*

FundamentosMEAN/node/ex03_singleton.js

```
let numero = 1

function exibirProximo() {
    console.log(numero++)
}

module.exports = { exibirProximo }
```

Listagem 60 - *Criar arquivo ex03_teste.js*

FundamentosMEAN/node/ex03_teste.js

```
const s1 = require('./ex03_singleton')
const s2 = require('./ex03_singleton')

s1.exibirProximo()
s2.exibirProximo()
s1.exibirProximo()
s2.exibirProximo()
```

No Atom aperte `alt + r` para executar o arquivo `ex03_teste.js`.

3.4. Node - Exercício 04: Objeto Global

Listagem 61 - *Criar arquivo ex04_global.js*

FundamentosMEAN/node/ex04_global.js

```
const PI = 3.14
console.log(global.PI)

global.obj = { name: 'Estou no global!' }
```

No Atom aperte `alt + r` para executar o arquivo `ex04_global.js`.

Listagem 62 - *Criar arquivo ex04_teste.js*

FundamentosMEAN/node/ex04_teste.js

```
require('./ex04_global')

console.log(global.obj.name)
console.log(obj.name)
```

No Atom aperte `alt + r` para executar o arquivo `ex04_teste.js`.

3.5. Node - Exercício 05: This

Listagem 63 - *Criar arquivo ex05_module.js*

FundamentosMEAN/node/ex05_module.js

```
console.log(global === this)
console.log(module === this)
console.log(module.exports === this)

this.digaOi = function () {
    console.log('Oi!!!')
}
```

Listagem 64 - *Criar arquivo ex05_teste.js*

FundamentosMEAN/node/ex05_teste.js

```
const modulo = require('./ex05_module')

modulo.digaOi()
```

No Atom aperte `alt + r` para executar o arquivo `ex05_teste.js`.

3.6. Node - Exercício 06: Módulo Externo (Lodash)

Listagem 65 - *Criar arquivo ex06_lodash.js*

FundamentosMEAN/node/ex06_lodash.js

```
const _ = require('lodash')

const alunos = [{
    nome: 'Joao',
    nota: 7.6
}, {
    nome: 'Maria',
    nota: 8.6
}, {
    nome: 'Pedro',
    nota: 8.1
}]

const media = _.sumBy(alunos, 'nota') / alunos.length
console.log(media)
```

Abra o terminal e dentro da pasta `FundamentosMEAN/node` execute os seguintes comandos:

Listagem 66 - *Criar arquivo package.json*

FundamentosMEAN/node

```
npm init -y
```

Listagem 67 - *Instalando a dependência lodash*

```
npm i lodash --save
```

Listagem 68 - *Criar arquivo .gitignore*

FundamentosMEAN/node

```
node_modules
```

No Atom aperte `alt + r` para executar o arquivo `ex06_lodash.js`.

3.7. Node - Exercício 07: Passagem de Parâmetros

Listagem 69 - *Criar arquivo ex07_param.js*

FundamentosMEAN/node/ex07_param.js

```
module.exports = function(param) {
    console.log(`O param informado foi ${param}`)
}
```

Listagem 70 - *Criar arquivo ex07_teste.js*

FundamentosMEAN/node/ex07_teste.js

```
const moduloComoParam = require('./ex07_param')
moduloComoParam('param1')
```

No Atom aperte `alt + r` para executar o arquivo `ex07_teste.js `.

3.8. Node - Exercício 08: Process (ARGV)

No Atom aperte `alt + r` para executar o arquivo `ex08_process.js`.

Listagem 71 - *Criar arquivo ex08_process.js*

FundamentosMEAN/node/ex08_process.js

```
function temParam(param) {
    return process.argv.indexOf(param) !== -1
}

if(temParam('--producao')){
    console.log('Atenção total!')
} else {
    console.log('Tranquilo!!!')
}
```

Abra o terminal e dentro da pasta `FundamentosMEAN/node` execute os seguintes comandos:

Listagem 72 - *Executar arquivo ex08_process.js*

```
node ex08_process
```

Listagem 73 - *Executar arquivo ex08_process.js*

```
node ex08_process --prducao
```

3.9. Node - Exercício 09: Process (STDIN/STDOUT)

Listagem 74 - *Criar arquivo ex09_process.js*

FundamentosMEAN/node/ex09_process.js

```
process.stdout.write('Está gostando do curso? ')
process.stdin.on('data', function(data) {
    process.stdout.write(`Sua resposta foi ${data.toString()}Obrigado!\n`)
    process.exit()
})
```

No Atom aperte `alt + r` para executar o arquivo `ex09_process.js`.

3.10. Node - Exercício 10: Módulo FS

Listagem 75 - *Criar arquivo ex10_fs.js*

FundamentosMEAN/node/ex10_fs.js

```
const fs = require('fs')
const files = fs.readdirSync(__dirname)

files.forEach(f => console.lgo(f))
```

No Atom aperte `alt + r` para executar o arquivo `ex10_fs.js`.

3.11. Node - Exercício 11: Módulo HTTP

Listagem 76 - *Criar arquivo ex11_http.js*

FundamentosMEAN/node/ex11_http.js

```
const http = require('http')
const server = http.createServer(function(req, res) {
    res.writeHead(200, {"Content-Type": "text/html"})
    res.end('<h1>Acho que é melhor usar o Express, não?</h1>')
})

const porta = 3456
server.listen(porta, function() {
    console.log(`Escutando a ${porta}`)
})
```

No Atom aperte `alt + r` para executar o arquivo `ex11_http.js`.

4. Express

4.1. Express - Visão Geral

4.2. Express - Exercício 01: Configuração e Mapeando uma Rota

Abra o terminal e dentro da pasta do projeto FundamentosMEAN execute o seguinte comando:

Listagem 77 - *Criar pasta express*

FundamentosMEAN/express

```
mkdir express && cd express
```

Listagem 78 - *Criar arquivo package.json*

FundamentosMEAN/express/package.json

```
npm init
```

Durante a criação do package.json mude o nome do projeto para `exercicios_express` e coloque seu nome como autor e as demais configurações deixe padrão.

Listagem 79 - *Instalando a dependência*

```
npm i --save express
```

Listagem 80 - *Abrir o Atom*

```
atom .
```

Listagem 81 - *Criar arquivo ex01.js*

FundamentosMEAN/express/ex01.js

```
const express = require('express')
const server = express()

server.get('/', function(req, res){
    res.send('<h1>Index</h1>')
})

server.all('/teste', function(req, res){
    res.send('<h1>Teste!</h1>')
})

server.get(/api/, function(req, res){
    res.send('<h1>API!</h1>')
})

server.listen(3000, () => console.log('Executando...'))
```

No Atom aperte `alt + r` para executar o arquivo `ex01.js`.

Outro plugin que foi instalado no Atom foi o `browser-plus` para visualizar as páginas e para executar o browser aperte `ctrl + shift + p` que será exibido um campo e você digita `Browser Plus Open` e clique nele. No browser digite a url `http:localhost:3000` que será exibida a página index.

4.3. Express - Exercício 02: Cadeia de Middlewares

Listagem 82 - *Criar arquivo ex02.js*

FundamentosMEAN/express/ex02.js

```
const express = require('express')
const server = express()

server.get('/', function(req, res, next){
    console.log('Inicio...')
    next()
    console.log('Fim...')
})

server.get('/', function(req, res){
    console.log('Resposta...')
    res.send('<h1>Olá Express</h1>')
})

server.listen(3000, () => console.log('Executando...'))
```

No Atom aperte `alt + r` para executar o arquivo `ex02.js`.

4.4. Express - Exercício 03: Método USE

Listagem 83 - *Criar arquivo ex03.js*

FundamentosMEAN/express/ex03.js

```
const express = require('express')
const server = express()

server.use(function(req, res, next){
    console.log('Inicio...')
    next()
    console.log('Fim...')
})

server.use(function(req, res){
    console.log('Resposta...')
    res.send('<h1>API!</h1>')
})

server.listen(3000, () => console.log('Executando...'))
```

No Atom aperte `alt + r` para executar o arquivo `ex03.js`.

4.5. Express - Exercício 04: Método Route

Listagem 84 - *Criar arquivo ex04.js*

FundamentosMEAN/express/ex04.js

```
const express = require('express')
const server = express()

server.route('/clientes')
    .get((req, res) => res.send('Lista de Clientes'))
    .post((req, res) => res.send('Novo Cliente'))
    .put((req, res) => res.send('Altera Cliente'))
    .delete((req, res) => res.send('Remove Cliente'))

server.listen(3000, () => console.log('Executando...'))
```

4.6. Express - Exercício 05: Express Router

Listagem 85 - *Criar arquivo ex05_routes.js*

FundamentosMEAN/express/ex05_routes.js

```
const express = require('express')
const router = express.Router()

router.use((req, res, next) => {
    const init = Date.now()
    next()
    console.log(`Tempo = ${Date.now() - init} ms.`)
})

router.get('/produtos/:id', (req, res) => {
    res.json({id: req.params.id, name: 'Caneta'})
})

router.get('/clientes/:id/:name', (req, res) => {
    res.json({id: req.params.id, name: req.params.name})
})

module.exports = router
```

Listagem 86 - *Criar arquivo ex05.js*

FundamentosMEAN/express/ex05.js

```
const express = require('express')
const server = express()
const router = require('./ex05_routes')

server.use('/api', router)

server.listen(3000, () => console.log('Executando...'))
```

No Atom aperte `alt + r` para executar o arquivo `ex05.js`.

4.7. Express - Exercício 06: Express e Router são Singletons?

Listagem 87 - *Criar arquivo ex06.js*

FundamentosMEAN/express/ex06.js

```
const express1 = require('express')
const express2 = require('express')
console.log(express1 === express2)

const server1 = express1()
const server2 = express1()
console.log(server1 === server2)

const router1 = express1.Router()
const router2 = express1.Router()
console.log(router1 === router2)
```

No Atom aperte `alt + r` para executar o arquivo `ex06.js`.

5. Angular

5.1. Angular - Instalando via NPM

Abra o terminal e dentro da pasta do projeto FundamentosMEAN execute o seguinte comando:

Listagem 88 - *Criar pasta angular1*

FundamentosMEAN/angular1

```
mkdir angular1 && cd angular1
```

Listagem 89 - *Criar arquivo package.json*

FundamentosMEAN/angular1/package.json

```
npm init
```

Durante a criação do package.json mude o nome do projeto para `exercicios_angular` e coloque seu nome como autor e as demais configurações deixe padrão.

Listagem 90 - *Instalando a dependência*

```
npm i angular --save
```

Listagem 91 - *Abrir o Atom*

```
atom .
```

Listagem 92 - *Criar arquivo .gitignore*

FundamentosMEAN/angular1/.gitignore

```
node_modules
```

5.2. Angular - Exercício 01: Configurando uma Página com Angular

Listagem 93 - *Criar arquivo ex01.html*

FundamentosMEAN/angular1/ex01.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
    </script>
  </head>
  <body>
    {{ 1 + 1 }}
  </body>
</html>
```

5.3. Angular - Exercício 02: Binding

Listagem 94 - *Criar arquivo ex02.html*

FundamentosMEAN/angular1/ex02.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
    </script>
  </head>
  <body>
    <h1>{{ value }}</h1>
    <input ng-model="value">
  </body>
</html>
```

5.4. Angular - Exercício 03: Controller

Listagem 95 - *Criar arquivo ex03.html*

FundamentosMEAN/angular1/ex03.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
      angular.module('app').controller('MeuController', [
        '$scope',
        function($scope){
          $scope.value = 10
          $scope.inc = function() {
            $scope.value++
          }
        }
      ])
    </script>
  </head>
  <body>
    <div ng-controller="MeuController">
      <h1>{{ value }}</h1>
      <input ng-model="value"/>
      <button ng-click="inc()">Inc</button>
    </div>
  </body>
</html>
```

5.5. Angular - Exercício 04: Controller As

Listagem 96 - *Criar arquivo ex04.html*

FundamentosMEAN/angular1/ex04.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
      angular.module('app').controller('MeuController', [
        function() {
          const self = this
          self.value = 10
          self.inc = function () {
            self.value++
          }
        }
      ])
    </script>
  </head>
  <body>
    <div ng-controller="MeuController as ctrl">
      <h1>{{ ctrl.value }}</h1>
      <input ng-model="ctrl.value"/>
      <button ng-click="ctrl.inc()">Inc</button>
    </div>
  </body>
</html>
```

5.6. Angular - Exercício 05: Filter

Listagem 97 - *Criar arquivo ex05.html*

FundamentosMEAN/angular1/ex05.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
      angular.module('app').filter('upper', function() {
        return function (input) {
          return input.toUpperCase()
        }
      })
      angular.module('app').controller('MeuController', [
        function(){
          const self = this
          self.value = "texto"
        }
      ])
    </script>
  </head>
  <body>
    <div ng-controller="MeuController as ctrl">
      <h1>{{ ctrl.value | upper }}</h1>
      <input ng-model="ctrl.value"/>
    </div>
  </body>
</html>
```

5.7. Angular - Exercício 06: Factory

Listagem 98 - *Criar arquivo ex06.html*

FundamentosMEAN/angular1/ex06.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
      angular.module('app').factory('utils', function() {
        function upper(input) {
          return input.toUpperCase()
        }
        return{upper}
      })
      angular.module('app').controller('MeuController', [
        'utils',
        function (utils) {
          const self = this
          self.value = utils.upper("texto")
        }
      ])
    </script>
  </head>
  <body>
    <div ng-controller="MeuController as ctrl">
        <h1>{{ ctrl.value }}</h1>
    </div>
  </body>
</html>
```

5.8. Angular - Exercício 07: Service

Listagem 99 - *Criar arquivo ex07.html*

FundamentosMEAN/angular1/ex07.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
      angular.module('app').service('utils', function() {
        this.upper = function(input) {
          return input.toUpperCase()
        }
      })
      angular.module('app').controller('MeuController', [
        'utils',
        function (utils) {
          const self = this
          self.value = utils.upper("texto")
        }
      ])
    </script>
  </head>
  <body>
    <div ng-controller="MeuController as ctrl">
        <h1>{{ ctrl.value }}</h1>
    </div>
  </body>
</html>
```

5.9. Angular - Exercício 08: Directive

Listagem 100 - *Criar arquivo ex08.html*

FundamentosMEAN/angular1/ex08.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
      angular.module('app', [])
      angular.module('app').directive('meuRelogio', function() {
        return {
          restrict: 'AE',
          template: `<h2>${new Date}</h2>`
        }
      })
    </script>
  </head>
  <body>
    <h1>Atributo</h1>
    <div meu-relogio></div>
    <hr>
    <h1>Elemento</h1>
    <meu-relogio></meu-relogio>
  </body>
</html>
```

5.10. Angular - Exercício 09: Component

Listagem 101 - *Criar arquivo ex09.html*

FundamentosMEAN/angular1/ex09.html

```
<!DOCTYPE html>
<html ng-app='app'>
  <head>
    <meta charset="utf-8">
    <title>Fundamentos de Angular</title>
    <script type="text/javascript" src="node_modules/angular/angular.min.js"
></script>
    <script type="text/javascript">
        angular.module('app', [])
        angular.module('app').component('field', {
            bindings: {
                id: '@',
                label: '@',
                model: '='
            },
            template: `
                <label for="{{ $ctrl.id }}">{{ $ctrl.label }}</label>
                <input id="{{ $ctrl.id }}" ng-model='$ctrl.model'/>
            `
        })
        angular.module('app').controller('MeuController', function() {
            const self = this
            self.name = 'Anônimo'
        })
    </script>
  </head>
  <body>
     <div ng-controller="MeuController as ctrl">
        <h1>{{ ctrl.name }}</h1>
        <field id="nome" label="Nome" model="ctrl.name"></field>
    </div>
  </body>
</html>
```

6. Backend - Configurações Iniciais

6.1. Versão Inicial do Projeto Backend

Abra o terminal e dentro da pasta Desktop execute o seguinte comando:

Listagem 102 - *Criar pasta CursoFramesWeb*

Desktop/CursoFramesWeb

```
mkdir CursoFramesWeb && cd CursoFramesWeb
```

Listagem 103 - *Criar pasta backend*

CursoFramesWeb/backend

```
mkdir backend && cd backend
```

Listagem 104 - *Criar arquivo package.json*

CursoFramesWeb/backend/package.json

```
npm init
```

Durante a criação do package.json mude o entry point para `loader.js` e coloque seu nome como autor e as demais configurações deixe padrão.

Listagem 105 - *Instalando as dependências*

```
npm i express body-parser mongoose node-restful mongoose-paginate lodash express-
query-int pm2 --save
```

Listagem 106 - *Instalando a dependência*

```
npm i nodemon --save-dev
```

Listagem 107 - *Abrir o Atom*

```
atom .
```

Listagem 108 - *Alterar arquivo package.json*

CursoFramesWeb/backend/package.json *(aprox. linha 7)*

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "loader.js",
  "scripts": {
    "dev": "nodemon",
    "production": "pm2 start loader.js --name backend"
  },
  "author": "Leonardo Leitão",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.17.1",
    "express": "^4.15.2",
    "express-query-int": "^1.0.1",
    "lodash": "^4.17.4",
    "mongoose": "^4.9.2",
    "mongoose-paginate": "^5.0.3",
    "node-restful": "^0.2.6",
    "pm2": "^2.4.2"
  },
  "devDependencies": {
    "nodemon": "^1.11.0"
  }
}
```

6.2. Adicionando o .gitignore ao Projeto

Listagem 109 - *Criar arquivo .gitignore*

CursoFramesWeb/backend/.gitignore

```
node_modules
*.log
```

6.3. Implementando o Servidor com Express

Listagem 110 - *Criar arquivo loader.js*

CursoFramesWeb/backend/loader.js

```
const port = 3003

const bodyParser = require('body-parser')
const express = require('express')
const server = express()

server.use(bodyParser.urlencoded({ extended: true }))
server.use(bodyParser.json())

server.listen(port, function() {
  console.log(`BACKEND is running on port ${port}.`);
})
```

Listagem 111 - *Executar a aplicação*

```
npm run dev
```

6.4. Criando a Conexão com MongoDB

Criar uma pasta chamada `config` em `backend/config`.

O conteúdo do arquivo `loader.js` foi movido para o arquivo `server.js`.

Listagem 112 - *Criar arquivo server.js*

CursoFramesWeb/backend/config/server.js

```
const port = 3003

const bodyParser = require('body-parser')
const express = require('express')
const server = express()

server.use(bodyParser.urlencoded({ extended: true }))
server.use(bodyParser.json())

server.listen(port, function() {
  console.log(`BACKEND is running on port ${port}.`);
})
```

Listagem 113 - *Criar arquivo database.js*

CursoFramesWeb/backend/config/database.js

```
const mongoose = require('mongoose')
module.exports = mongoose.connect('mongodb://localhost/db_finance')
```

Listagem 114 - *Alterar arquivo loader.js*

CursoFramesWeb/backend/loader.js *(na linha 1)*

```
require('./config/server')
require('./config/database')
```

Abra um novo terminal para executar o servidor do Mongo.

Listagem 115 - *Executar o servidor Mongo*

```
mongod
```

7. Backend - Ciclo de Pagamento API

7.1. Mapeamento ODM do Objeto Ciclo de Pagamentos

Criar uma pasta chamada `api` e dentro dela criar uma pasta chamada `billingCycle` em `backend/api/billingCycle`

Listagem 116 - *Criar arquivo billingCycle.js*

`CursoFramesWeb/backend/api/billingCycle/billingCycle.js`

```
const restful = require('node-restful')
const mongoose = restful.mongoose

const creditSchema = new mongoose.Schema({
  name: { type: String, required: true },
  value: { type: Number, min: 0, required: true }
})

const debtSchema = new mongoose.Schema({
  name: { type: String, required: true },
  value: { type: Number, min: 0, required: true },
  status: { type: String, required: false, uppercase: true,
    enum: ['PAGO', 'PENDENTE', 'AGENDADO'] }
})

const billingCycleSchema = new mongoose.Schema({
  name: { type: String, required: true },
  month: { type: Number, min: 1, max: 12, required: true },
  year: { type: Number, min: 1970, max: 2100, required: true },
  credits: [creditSchema],
  debts: [debtSchema]
})

module.exports = restful.model('BillingCycle', billingCycleSchema)
```

7.2. Serviço de Ciclo de Pagamentos

Listagem 117 - *Criar arquivo billingCycleService.js*

CursoFramesWeb/backend/api/billingCycle/billingCycleService.js

```
const BillingCycle = require('./billingCycle')

BillingCycle.methods(['get', 'post', 'put', 'delete'])

module.exports = BillingCycle
```

7.3. Criando o Arquivo de Rotas

Listagem 118 - *Criar arquivo routes.js*

CursoFramesWeb/backend/config/routes.js

```
const express = require('express')

module.exports = function(server) {

  // API Routes
  const router = express.Router()
  server.use('/api', router)
}
```

Listagem 119 - *Alterar arquivo loader.js*

CursoFramesWeb/backend/loader.js *(aprox. linha 1 e 3)*

```
const server = require('./config/server')
require('./config/database')
require('./config/routes')(server)
```

Listagem 120 - *Alterar arquivo server.js*

CursoFramesWeb/backend/config/server.js *(aprox. linha 14)*

```
const port = 3003

const bodyParser = require('body-parser')
const express = require('express')
const server = express()

server.use(bodyParser.urlencoded({ extended: true }))
server.use(bodyParser.json())

server.listen(port, function() {
  console.log(`BACKEND is running on port ${port}.`);
})

module.exports = server
```

7.4. Registrando as Rotas do Serviço de Ciclo de Pagamentos

Listagem 121 - *Alterar arquivo routes.js*

CursoFramesWeb/backend/config/routes.js *(aprox. linha 9)*

```
const express = require('express')

module.exports = function (server) {

  // API Routes
  const router = express.Router()
  server.use('/api', router)

// rotas da API
  const billingCycleService = require('../api/billingCycle/billingCycleService')
  billingCycleService.register(router, '/billingCycles')
}
```

7.5. Testando a API de Ciclo de Pagamentos (Parte 1)

Listagem 122 - *Alterar arquivo database.js*

CursoFramesWeb/backend/config/database.js *(aprox. linha 4)*

```
const mongoose = require('mongoose')
module.exports = mongoose.connect('mongodb://localhost/db_finance')

mongoose.Error.messages.general.required = "O atributo '{PATH} é obrigatório'"
```

Listagem 123 - *Alterar arquivo billingCycle.js*

CursoFramesWeb/backend/api/billingCycle/billingCycle.js *(aprox. linha 11)*

```
const restful = require('node-restful')
const mongoose = restful.mongoose

const creditSchema = new mongoose.Schema({
  name: { type: String, required: true },
  value: { type: Number, min: 0, required: true }
})

const debtSchema = new mongoose.Schema({
  name: { type: String, required: true },
  value: { type: Number, min: 0, required: [true, 'Informe o valor do débito!']
},
  status: { type: String, required: false, uppercase: true,
    enum: ['PAGO', 'PENDENTE', 'AGENDADO'] }
})

const billingCycleSchema = new mongoose.Schema({
  name: { type: String, required: true },
  month: { type: Number, min: 1, max: 12, required: true },
  year: { type: Number, min: 1970, max: 2100, required: true },
  credits: [creditSchema],
  debts: [debtSchema]
})

module.exports = restful.model('BillingCycle', billingCycleSchema)
```

7.6. Testando a API de Ciclo de Pagamentos (Parte 2)

Listagem 124 - *Alterar arquivo billingCycleService.js*

CursoFramesWeb/backend/api/billingCycle/billingCycleService.js *(aprox. linha 4)*

```
const BillingCycle = require('./billingCycle')

BillingCycle.methods(['get', 'post', 'put', 'delete'])
BillingCycle.updateOptions({new: true, runValidators: true})

module.exports = BillingCycle
```

Listagem 125 - *Alterar arquivo database.js*

CursoFramesWeb/backend/config/database.js *(aprox. linha 5)*

```
const mongoose = require('mongoose')
module.exports = mongoose.connect('mongodb://localhost/db_finance')

mongoose.Error.messages.general.required = "O atributo '{PATH} é obrigatório'"
mongoose.Error.messages.Number.min = "O '{VALUE}' informado é menor que o limite
mínimo de '{MIN}'."
mongoose.Error.messages.Number.max = "O '{VALUE}' informado é maior que o limite
máximo de '{MAX}'."
mongoose.Error.messages.String.enum = "'{VALUE}' não é válido para o atributo
'{PATH}'."
```

7.7. Testando a API de Ciclo de Pagamentos (Parte 3)

7.8. Serviço Contador (count) de Ciclo de Pagamentos

Listagem 126 - *Alterar arquivo billingCycleService.js*

CursoFramesWeb/backend/api/billingCycle/billingCycleService.js *(aprox. linha 6)*

```
const BillingCycle = require('./billingCycle')

BillingCycle.methods(['get', 'post', 'put', 'delete'])
BillingCycle.updateOptions({new: true, runValidators: true})

BillingCycle.route('count', function (req, res, next) {
  BillingCycle.count(function (error, value) {
    if (error) {
      res.status(500).json({errors: [error]})
    }else {
      res.json({value})
    }
  })
})

module.exports = BillingCycle
```

8. Backend - Sumário de Pagamento API

8.1. Serviço de Sumário de Pagamentos

Criar uma pasta chamada `billingSummary` em `backend/api/billingSummary`

Listagem 127 - *Criar arquivo billingSummaryService.js*

`CursoFramesWeb/backend/api/billingSummary/billingSummaryService.js`

```
const _ = require('lodash')
const BillingCycle = require('../billingCycle/billingCycle')

// Mais uma função middleware
function getSummary(req, res) {
  BillingCycle.aggregate({
    $project: {credit: {$sum: "$credits.value"}, debt: {$sum: "$debts.value"}}
    }, {
      $group: {_id: null, credit: {$sum: "$credit"}, debt: {$sum: "$debt"}}
    }, {
      $project: {_id: 0, credit: 1, debt: 1}
    }, function (error, result) {
      if (error) {
        res.status(500).json({errors: [error]})
      } else {
        res.json(_.defaults(result[0], {credit:0, debt: 0}))
      }
  })
}

module.exports = { getSummary }
```

8.2. Registrando a Rota do Serviço de Sumário de Pagamentos

Listagem 128 - *Alterar arquivo routes.js*

CursoFramesWeb/backend/config/routes.js *(aprox. linha 13)*

```
const express = require('express')

module.exports = function (server) {

  // API Routes
  const router = express.Router()
  server.use('/api', router)

// rotas da API
  const billingCycleService = require('../api/billingCycle/billingCycleService')
  billingCycleService.register(router, '/billingCycles')

  const billingSummaryService =
require('../api/billingSummary/billingSummaryService')
  router.route('/billingSummary').get(billingSummaryService.getSummary)
}
```

9. Backend - Ajustes Finais

9.1. Uniformizando as Mensagens de Erro

Listagem 129 - *Alterar arquivo billingCycleService.js*

CursoFramesWeb/backend/api/billingCycle/billingCycleService.js *(aprox. linha 7, 9 e 20)*

```
const BillingCycle = require('./billingCycle')

BillingCycle.methods(['get', 'post', 'put', 'delete'])
BillingCycle.updateOptions({new: true, runValidators: true})

BillingCycle.after('post', sendErrorsOrNext).after('put', sendErrorsOrNext)

function sendErrorsOrNext(req, res, next) {
  const bundle = res.locals.bundle

  if (bundle.errors) {
    const errors = parseErrors(bundle.errors)
    res.status(500).json({errors})
  } else {
    next()
  }
}

function parseErrors(nodeRestfulErrors) {
  const errors = []
  _.forIn(nodeRestfulErrors, error => errors.push(error.message))
  return errors
}

BillingCycle.route('count', function (req, res, next) {
  BillingCycle.count(function (error, value) {
    if (error) {
      res.status(500).json({errors: [error]})
    }else {
      res.json({value})
    }
  })
})

module.exports = BillingCycle
```

10. Refactories e Correções

10.1. One-Time Binding e AngularJS Batarang

Foi instalado uma extensão do Chrome chamada `AngularJS Batarang` e para acessar basta ir em ferramentas do desenvolvedor na aba `AngularJS` e ativar.

Link para instalar o AngularJS Batarang: https://chrome.google.com/webstore/detail/angularjs-batarang/ighdmehidhipcmcojjgiloacoafjmpfk

Listagem 130 - *Alterar arquivo list.html*

`CursoFramesWeb/angular1/app/billingCycle/list.html` *(aprox. linha 13, 14 e 15)*

```
<div class="box-body">
  <table class="table">
    <thead>
      <tr>
        <th>Nome</th>
        <th>Mês</th>
        <th>Ano</th>
        <th>Ações</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="billingCycle in bcCtrl.billingCycles">
        <td>{{:: billingCycle.name }}</td>
        <td>{{:: billingCycle.month }}</td>
        <td>{{:: billingCycle.year }}</td>
        <td class="table-actions">
          <button class="btn btn-warning" ng-click
="bcCtrl.showTabUpdate(billingCycle)"><i class="fa fa-pencil"></i></button>
          <button class="btn btn-danger" ng-click
="bcCtrl.showTabDelete(billingCycle)"><i class="fa fa-trash-o"></i></button>
        </td>
      </tr>
    </tbody>
  </table>
  <div class="box-footer clearfix">
    <paginator url="/#!/billingCycles" pages="{{ bcCtrl.pages }}"></paginator>
  </div>
</div>
```

10.2. Atualizando o Projeto para Angular 1.6

Abra o terminal e dentro da pasta `angular1` execute o seguinte comando:

Listagem 131 - *Atualizar as dependências*

```
npm update
```

Listagem 132 - *Executar a aplicação*

```
npm run dev
```

Listagem 133 - *Alterar arquivo billingCycleController.js*

`CursoFramesWeb/angular1/app/billingCycle/billingCycleController.js` *(aprox. linha 16, 18, 21, 29, 32, 33, 51, 54, 55, 61, 64 e 65)*

```
(function(){
    angular.module('primeiraApp').controller('BillingCycleCtrl', [
    '$http',
    '$location',
    'msgs',
    'tabs'
    BillingCycleController
    ])

    function BillingCycleController($http, $location, msgs, tabs) {
        const vm = this
        const url = 'http://localhost:3003/api/billingCycles'

        vm.refresh = function () {
            const page = parseInt($location.search().page) || 1
            $http.get(`${url}?skip=${(page - 1) * 10}&limit=10`).then(function
(response) {
                vm.billingCycle = {credits: [{}], debts:[{}]}
                vm.billingCycles = response.data
                vm.calculateValues()

                $http.get(`${url}/count`).then(function (response) {
                    vm.pages = Math.ceil(response.value / 10)
                    tabs.show(vm, {tabList: true, tabCreate: true})
                })
            })
        }
```

```
        vm.create = function(){
            $http.post(url, vm.billingCycle).then(function(response){
                vm.refresh()
                msgs.addSuccess('Operação realizada com sucesso!!')
            }).catch(function(response) {
                msgs.addError(response.data.errors)
            })
        }

        vm.showTabUpdate = function (billingCycle) {
            vm.billingCycle = billingCycle
            vm.calculateValues()
            tabs.show(vm, {tabUpdate: true})
        }

        vm.showTabDelete = function (billingCycle) {
            vm.billingCycle = billingCycle
            vm.calculateValues()
            tabs.show(vm, {tabDelete: true})
        }

        vm.update = function(){
            const updateUrl = `${url}/${vm.billingCycle._id}`
            $http.put(updateUrl, vm.billingCycle).then(function(response){
                vm.refresh()
                msgs.addSuccess('Operação realizada com sucesso!')
            }).catch(function(response){
                msgs.addError(response.data.errors)
            })
        }

        vm.delete = function(){
            const deleteUrl = `${url}/${vm.billingCycle._id}`
            $http.delete(deleteUrl, vm.billingCycle).then(function(response){
                vm.refresh()
                msgs.addSuccess('Operação realizada com sucesso!')
            }).catch(function(response){
                msgs.addError(response.data.errors)
            })
        }

        vm.addCredit = function (index) {
            vm.billingCycle.credits.splice(index + 1, 0,{} )
        }

        vm.cloneCredit = function (index, {name, value}) {
            vm.billingCycle.credits.splice(index + 1, 0,{name, value} )
            vm.calculateValues()
        }
```

```
        vm.deleteCredit = function (index) {
            if(vm.billingCycle.credits.length > 1){
                vm.billingCycle.credits.splice(index, 1 )
                vm.calculateValues()
            }
        }

        vm.addDebt = function (index) {
            vm.billingCycle.debts.splice(index + 1, 0,{} )
        }

        vm.cloneDebt = function (index, {name, value, status}) {
            vm.billingCycle.debts.splice(index + 1, 0,{name, value, status})
            vm.calculateValues()
        }

        vm.deleteDebt = function (index) {
            if(vm.billingCycle.debts.length > 1){
                vm.billingCycle.debts.splice(index, 1 )
                vm.calculateValues()
            }
        }

        vm.calculateValues = function () {
            vm.credit = 0
            vm.debt = 0

            if (vm.billingCycle) {
                vm.billingCycle.credits.forEach(function ({value}) {
                    vm.credit += !value || isNaN(value) ? 0 : parseFloat(value)
                })

                vm.billingCycle.debts.forEach(function ({value}) {
                    vm.debt += !value || isNaN(value) ? 0 : parseFloat(value)
                })
            }
            vm.total = vm.credit - vm.debt
        }

        vm.refresh()
    }
})()
```

Listagem 134 - *Alterar arquivo field.js*

CursoFramesWeb/angular1/app/common/components/field.js *(aprox. linha 15)*

```
(function() {
  angular.module('primeiraApp').component('field', {
    bindings: {
      id: '@',
      label: '@',
      grid: '@',
      placeholder: '@',
      type: '@',
      model: '=',
      readonly: '<',
    },
    controller: [
      'gridSystem',
      function(gridSystem){
        this.$onInit = () => this.gridClasses = gridSystem.toCssClasses(
this.grid)
      }
    ],
    template: `
    <div class="{{ $ctrl.gridClasses }}">
      <div class="form-group">
        <label for="{{ $ctrl.id }}">{{ $ctrl.label }}</label>
        <input id="{{ $ctrl.id }}" class="form-control" placeholder="{{
$ctrl.placeholder }}"
            type="{{ $ctrl.type }}" ng-model="$ctrl.model" ng-
readonly="$ctrl.readonly"/>
      </div>
    </div>
    `
  })
})()
```

Listagem 135 - *Alterar arquivo dashboardController.js*

CursoFramesWeb/angular1/app/dashboard/dashboardController.js *(aprox. linha 11 e 12)*

```
(function(){
  angular.module('primeiraApp').controller('DashboardCtrl', [
    '$http',
    DashboardController
  ])

  function DashboardController($http) {
    const vm = this
    vm.getSummary = function () {
      const url = 'http://localhost:3003/api/billingSummary'
      $http.get(url).then(function (response) {
        const {credit = 0, debt = 0} = response.data
        vm.credit = credit
        vm.debt = debt
        vm.total = credit - debt
      })
    }

  vm.getSummary()
  }
})()
```

11. PrimeiraAPP: Autenticação

11.1. Backend: Novas Dependências

Iremos adicionar novas dependências ao projeto usando **versões específicas** para garantir compatibilidade. As mudanças importantes nas versões dos módulos adicionados serão tratadas em vídeos adicionais no final do curso.

Novas dependências são:

- bcrypt@1.0.2
- jsonwebtoken@7.3.0

11.1.1. Instalação

Abra o terminal e dentro da pasta do projeto backend execute o seguinte comando:

Listagem 1 - *Instalando as dependências*

```
npm i --save bcrypt@1.0.2 jsonwebtoken@7.3.0
```

11.2. Backend: Implementar Autenticação

11.2.1. Criar arquivo *.env*

O objetivo principal do `.env` é armazenar váriáveis de configuração da sua aplicação *backend*, por enquando será armazendo apenas a chave usada para gerar o token `JWT`.

Listagem 2 - *Criar arquivo .env*

`backend/.env`

```
module.exports = {
    // Você pode alterar essa chave!
    authSecret: 'skjdhf6$$%dojkhf^(sdkjhf'
}
```

É muito importante que o arquivo `.env` não seja commitado no repositório, pois nele está a chave secreta para geração do token.

Listagem 3 - *Adicionar .env ao .gitignore*

backend/.gitignore

```
node_modules
*.log
.env
```

11.2.2. Criar arquivo *user.js*

Criar uma nova pasta ao projeto *backend* chamada `user` dentro de `api`.

O objetivo principal do `user.js` é fazer o **ODM** (Mapeamento Objeto-Documento), ou seja, mapear o objeto javascript *user* para o documento que será armazenado no MongoDB.

Listagem 4 - *Criar arquivo user.js*

backend/api/user/user.js

```
const restful = require('node-restful')
const mongoose = restful.mongoose

const userSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true },
    password: { type: String, min: 6, max: 12, required: true }
})

module.exports = restful.model('User', userSchema)
```

11.2.3. Criar arquivo *authService.js*

Listagem 5 - *Estrutura básica de authService.js*

backend/api/user/authService.js

```
const _ = require('lodash')
const jwt = require('jsonwebtoken')
const bcrypt = require('bcrypt')
const User = require('./user')
const env = require('../../.env')

const emailRegex = /\S+@\S+\.\S+/
const passwordRegex = /((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{6,12})/

const sendErrorsFromDB = (res, dbErrors) => {
    const errors = []
    _.forIn(dbErrors.errors, error => errors.push(error.message))
    return res.status(400).json({errors})
}
```

Listagem 6 - *Criar método login de authService.js*

backend/api/user/authService.js

```
const login = (req, res, next) => {
    const email = req.body.email || ''
    const password = req.body.password || ''

    User.findOne({email}, (err, user) => {
        if(err) {
            return sendErrorsFromDB(res, err)
        } else if (user && bcrypt.compareSync(password, user.password)) {
            const token = jwt.sign(user, env.authSecret, {
                expiresIn: "1 day"
            })
            const { name, email } = user
            res.json({ name, email, token })
        } else {
            return res.status(400).send({errors: ['Usuário/Senha inválidos']})
        }
    })
}
```

Listagem 7 - *Criar método validateToken de authService.js*

backend/api/user/authService.js

```
const validateToken = (req, res, next) => {
    const token = req.body.token || ''
    jwt.verify(token, env.authSecret, function(err, decoded) {
        return res.status(200).send({valid: !err})
    })
}
```

Listagem 8 - *Criar método signup de authService.js*

backend/api/user/authService.js

```
const signup = (req, res, next) => {
    const name = req.body.name || ''
    const email = req.body.email || ''
    const password = req.body.password || ''
    const confirmPassword = req.body.confirm_password || ''

    if(!email.match(emailRegex)) {
        return res.status(400).send({errors: ['O e-mail informado está invá
lido']})
    }

    if(!password.match(passwordRegex)) {
        return res.status(400).send({errors: [
            "Senha precisar ter: uma letra maiúscula, uma letra minúscula, um n
úmero, uma caractere especial(@#$%) e tamanho entre 6-12."
        ]})
    }

    const salt = bcrypt.genSaltSync()
    const passwordHash = bcrypt.hashSync(password, salt)
    if(!bcrypt.compareSync(confirmPassword, passwordHash)) {
        return res.status(400).send({errors: ['Senhas não conferem.']})
    }

    User.findOne({email}, (err, user) => {
        if(err) {
            return sendErrorsFromDB(res, err)
        } else if (user) {
            return res.status(400).send({errors: ['Usuário já cadastrado.']})
        } else {
            const newUser = new User({ name, email, password: passwordHash })
            newUser.save(err => {
                if(err) {
                    return sendErrorsFromDB(res, err)
                } else {
                    login(req, res, next)
                }
            })
        }
    })
}
```

Listagem 9 - *Exportar os métodos de authService.js*

backend/api/user/authService.js

```
module.exports = { login, signup, validateToken }
```

11.2.4. Criar middleware *auth.js*

Esse *[middleware]* será o responsável por validar o token *JWT* para as routas protegidas e garantir que a API esteja protegida.

Listagem 10 - *Criar middleware auth.js*

backend/config/auth.js

```
const jwt = require('jsonwebtoken')
const env = require('../.env')

module.exports = (req, res, next) => {

    // CORS preflight request
    if(req.method === 'OPTIONS') {
        next()
    } else {
        const token = req.body.token || req.query.token ||
req.headers['authorization']

        if(!token) {
            return res.status(403).send({errors: ['No token provided.']})
        }

        jwt.verify(token, env.authSecret, function(err, decoded) {
            if(err) {
                return res.status(403).send({
                    errors: ['Failed to authenticate token.']
                })
            } else {
                // req.decoded = decoded
                next()
            }
        })
    }
}
```

Listagem 11 - *Adicionar novo header em cors.js*

backend/config/cors.js

```
module.exports = (req, res, next) => {
    res.header('Access-Control-Allow-Origin', '*')
    res.header('Access-Control-Allow-Methods',
        'GET, POST, OPTIONS, PUT, PATCH, DELETE')
    res.header('Access-Control-Allow-Headers',
        'Origin, X-Requested-With, Content-Type, Accept, Authorization')
    next()
}
```

11.2.5. Alterar arquivo *routes.js*

Além de cadastrar as novas rotas *login*, *signup* e *validateToken*, essa alteração tem por objetivo separar as routas públicas das rotas privadas (acesso com autenticação).

Listagem 12 - *Alterar arquivo routes.js*

my-money-app/backend/src/config/routes.js

```
const express = require('express')
const auth = require('./auth')

module.exports = function (server) {

    /*
     * Rotas abertas
     */
    const openApi = express.Router()
    server.use('/oapi', openApi)

    const AuthService = require('../api/user/AuthService')
    openApi.post('/login', AuthService.login)
    openApi.post('/signup', AuthService.signup)
    openApi.post('/validateToken', AuthService.validateToken)

    /*
     * Rotas protegidas por Token JWT
     */
    const protectedApi = express.Router()
    server.use('/api', protectedApi)

    protectedApi.use(auth)

    const billingCycleService = require(
'../api/billingCycle/billingCycleService')
    billingCycleService.register(protectedApi, '/billingCycles')

    const billingSummaryService =
require('../api/billingSummary/billingSummaryService')
    protectedApi.route('/billingSummary').get(billingSummaryService.getSummary)
}
```

11.3. Frontend: Implementar Autenticação

11.3.1. Widget do Usuário

Listagem 13 - *Descomentar o item user-menu*

angular1/app/template/header/navbar/navbar.html *(aprox. linha 9)*

```
<li class="dropdown user user-menu"
    ng-include="'/template/header/navbar/user.html'"></li>
```

Listagem 14 - *Remover Menu Body*

angular1/app/template/header/navbar/user.html *(aprox. linhas 14-27)*

```
<!-- Menu Body -->
<li class="user-body">
    <div class="row">
        <div class="col-xs-4 text-center">
            <a href>Seguidores</a>
        </div>
        <div class="col-xs-4 text-center">
            <a href>Vendas</a>
        </div>
        <div class="col-xs-4 text-center">
            <a href>Amigos</a>
        </div>
    </div>
</li>
```

Listagem 15 - *Remover Botão Perfil*

angular1/app/template/header/navbar/user.html *(aprox. linhas 16-18)*

```
<div class="pull-left">
    <a href class="btn btn-default btn-flat">Perfil</a>
</div>
```

Listagem 16 - *Criar AuthController*

angular1/app/auth/authController.js *(aprox. linhas 16-18)*

```
angular.module('primeiraApp').controller('AuthCtrl', [
  '$location',
  'msgs',
  AuthController
])

function AuthController($location, msgs) {
    const vm = this

    vm.getUser = () => ({ name: 'Usuário MOCK', email: 'mock@cod3r.com.br' })

    vm.logout = () => {
        console.log('Logout...')
    }
}
```

Listagem 17 - *Adicionar ng-controller*

angular1/app/template/header/navbar/navbar.html *(aprox. linha 9)*

```
ng-controller="AuthCtrl as auth"
```

Listagem 18 - *Alterar user.html para acessar controller*

angular1/app/template/header/navbar/user.html *(aprox. linha 3)*

```
<span class="hidden-xs">{{:: auth.getUser().name }}</span>
```

angular1/app/template/header/navbar/user.html *(aprox. linhas 10-11)*

```
{{:: auth.getUser().name }}
<small>{{:: auth.getUser().email }}</small>
```

angular1/app/template/header/navbar/user.html *(aprox. linha 16)*

```
<a href class="btn btn-default btn-flat" ng-click="auth.logout()">Sair</a>
```

11.3.2. Estrutura Básica do Login/Cadastro

Listagem 19 - *Funções para suportar a tela Login/Cadastro*

angular1/app/auth/authController.js

```
vm.loginMode = true

vm.changeMode = () => vm.loginMode = !vm.loginMode

vm.login = () => {
    console.log(`Login... ${vm.user.email}`)
}

vm.signup = () => {
    console.log(`Signup... ${vm.user.email}`)
}
```

Listagem 20 - *Componente authField*

angular1/app/common/components/authField.js

```
angular.module('primeiraApp').component('authField', {
    bindings: {
        id: '@',
        label: '@',
        type: '@',
        grid: '@',
        icon: '@',
        model: '=',
        placeholder: '@',
        hide: '<'
    },
    controller: function() {
        this.$onInit = () => {
            this.iconClasses = `glyphicon glyphicon-${this.icon} form-control-
feedback`
        }
    },
    template: `
        <div class="form-group has-feedback">
            <input ng-model="$ctrl.model" id="{{ $ctrl.id }}" class="form-
control"
                type="{{ $ctrl.type }}" placeholder="{{ $ctrl.placeholder }}"
                ng-hide="$ctrl.hide" />
            <span class="{{ $ctrl.iconClasses }}"></span>
        </div>
    `
})
```

Listagem 21 - *Formulário Login/Cadastro*

angular1/app/auth/form.html

```
<div class="login-box" ng-controller="AuthCtrl as auth">
    <div class="login-logo"><b> Primeira</b> APP</div>
    <div class="login-box-body">
        <p class="login-box-msg">Bem vindo!</p>
        <form>
            <auth-field id="authName" model="auth.user.name" type="input"
                placeholder="Informe o Nome" hide="auth.loginMode"></auth-field>
            <auth-field id="authEmail" model="auth.user.email" type="email"
                placeholder="Informe o E-mail"></auth-field>
            <auth-field id="authPass" model="auth.user.password" type="password"
                placeholder="Informe a Senha"></auth-field>
            <auth-field id="authConfirmPass" model="auth.user.confirm_password"
type="password"
                placeholder="Confirme a Senha" hide="auth.loginMode"></auth-
field>
            <div class="row">
                <div class="col-sm-4">
                    <button class="btn btn-primary btn-block btn-flat"
                        ng-click="auth.login()" ng-show="auth.loginMode"
>Entrar</button>
                    <button class="btn btn-primary btn-block btn-flat"
                        ng-click="auth.signup()" ng-hide=
"auth.loginMode">Registrar</button>
                </div>
            </div>
        </form>
        <br />
        <a href="javascript:;" ng-click="auth.changeMode()">
            {{ auth.loginMode ?
                'Novo usuário? Registrar aqui!' :
                'Já é cadastrado? Entrar aqui!' }}
        </a>
    </div>
</div>

<style>
    html, .wrapper { background-color: #fff!important; }
    .login-box-body { background-color: #eee; }
</style>
```

Listagem 22 - *Página de Login/Cadastro*

angular1/app/auth.html

```
<!DOCTYPE html>
<html ng-app="primeiraApp">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>{{ consts.appName }}</title>
  <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no" name="viewport">
  <link rel="stylesheet" href="/assets/css/deps.min.css">
  <link rel="stylesheet" href="/assets/css/app.min.css">
</head>
<body class="skin-blue fixed sidebar-mini">
  <div class="wrapper" ng-include="'/auth/form.html'"></div>
  <script src="/assets/js/deps.min.js"></script>
  <script src="/assets/js/app.min.js"></script>
</body>
</html>
```

11.3.3. O coração do Login/Cadastro (AuthFactory)

Listagem 23 - *Novas Constantes*

angular1/app/auth/authController.js

```
oapiUrl: 'http://localhost:3003/oapi',
userKey: '_primeira_app_user'
```

Listagem 24 - *Estrutura inicial*

angular1/app/common/factories/authFactory.js

```
(function() {
    angular.module('primeiraApp').factory('auth', [
        '$http',
        'consts',
        AuthFactory
    ])

    function AuthFactory($http, consts) {

        return {}
    }
})()
```

Listagem 25 - *Funções de Login/Cadastro*

angular1/app/common/factories/authFactory.js

```
function signup(user, callback) {
    submit('signup', user, callback)
}

function login(user, callback) {
    submit('login', user, callback)
}

function submit(url, user, callback) {
    $http.post(`${consts.oapiUrl}/${url}`, user)
        .then(resp => {
            localStorage.setItem(consts.userKey, JSON.stringify(resp.data))
            $http.defaults.headers.common.Authorization = resp.data.token
            if (callback) callback(null, resp.data)
        }).catch(function (resp) {
            if (callback) callback(resp.data.errors, null)
        })
}
```

angular1/app/common/factories/authFactory.js *(exportar)*

```
signup, login
```

Listagem 26 - *Adicionando AuthFactory no AuthController*

angular1/app/auth/authController.js *(Lista de Dependências)*

```
'auth'
```

angular1/app/auth/authController.js *(Parâmetro da Função Controller)*

```
auth
```

Listagem 27 - *Alterando AuthController (login)*

angular1/app/auth/authController.js

```
auth.login(vm.user, err => err ? msgs.addError(err) : msgs.addSuccess(
'Sucesso!'))
```

Listagem 28 - *Alterando AuthController (signup)*

angular1/app/auth/authController.js

```
auth.signup(vm.user, err => err ? msgs.addError(err) : msgs.addSuccess(
'Sucesso!'))
```

Listagem 29 - *Função de Logout*

angular1/app/common/factories/authFactory.js

```
function logout(callback) {
    localStorage.removeItem(consts.userKey)
    $http.defaults.headers.common.Authorization = ''
    if (callback) callback(null)
}
```

angular1/app/common/factories/authFactory.js *(exportar)*

```
logout
```

Listagem 30 - *Alterando AuthController (logout)*

angular1/app/auth/authController.js

```
auth.logout(() => msgs.addSuccess('Sucesso!'))
```

Listagem 31 - *Função de getUser*

angular1/app/common/factories/authFactory.js

```
let user = null

function getUser() {
    if(!user) {
        user = JSON.parse(localStorage.getItem(consts.userKey))
    }
    return user
}
```

angular1/app/common/factories/authFactory.js *(exportar)*

```
getUser
```

Listagem 32 - *Alterando AuthController (getUser)*

angular1/app/auth/authController.js

```
auth.getUser()
```

11.3.4. Protegendo as Rotas

Listagem 33 - *Será executado na inicialização da aplicação*

angular1/app/config/routes.js

```
.run([
    '$rootScope',
    '$http',
    '$location',
    '$window',
    'auth',
    function ($rootScope, $http, $location, $window, auth) {
        console.log('Executando...')
    }
])
```

Listagem 34 - *Definindo função para validar usuário*

angular1/app/config/routes.js

```
validateUser()
$rootScope.$on('$locationChangeStart', () => validateUser())

function validateUser() {
    console.log('Executando...')
}
```

Listagem 35 - *Validar Usuário (Versão 1)*

angular1/app/config/routes.js

```
const user = auth.getUser()
const authPage = '/auth.html'
const isAuthPage = $window.location.href.includes(authPage)

if (!user && !isAuthPage) {
    $window.location.href = authPage
} else if (user && !user.isValid) {
    user.isValid = true
    $http.defaults.headers.common.Authorization = user.token
    isAuthPage ? $window.location.href = '/' : $location.path('/dashboard')
}
```

Listagem 36 - *Alterar URL após Login/Signup/Logout*

angular1/app/auth/authController.js *(função login)*

```
auth.login(vm.user, err => err ? msgs.addError(err) : $location.path('/'))
```

angular1/app/auth/authController.js *(função signup)*

```
auth.signup(vm.user, err => err ? msgs.addError(err) : $location.path('/'))
```

angular1/app/auth/authController.js *(função logout)*

```
auth.logout(() => $location.path('/'))
```

angular1/app/common/factories/authFactory.js *(função logout)*

```
user = null
```

==== Validar Token

backend/api/user/authService.js *(aprox. linha 25)*

```
expiresIn: "10 seconds"
```

angular1/app/common/factories/authFactory.js

```
function validateToken(token, callback) {
    if (token) {
        $http.post(`${consts.oapiUrl}/validateToken`, { token })
            .then(resp => {
                if (!resp.data.valid) {
                    logout()
                } else {
                    $http.defaults.headers.common.Authorization = getUser().token
                }
                if (callback) callback(null, resp.data.valid)
            }).catch(function (resp) {
                if (callback) callback(resp.data.errors)
            })
    } else {
        if (callback) callback('Token inválido.')
    }
}
```

angular1/app/common/factories/authFactory.js **(exportar)**

```
validateToken
```

Listagem 37 - Chamar validateToken_

angular1/app/config/routes.js *(aprox. linha 34)*

```
auth.validateToken(user.token, (err, valid) => {
    if (!valid) {
        $window.location.href = authPage
    } else {
        user.isValid = true
        $http.defaults.headers.common.Authorization = user.token
        isAuthPage ? $window.location.href = '/' : $location.path('/dashboard')
    }
})
```

Listagem 38 - Remover rota padrão_

angular1/app/config/routes.js *(Remover linha)*

```
$urlRouterProvider.otherwise('/dashboard')
```

11.3.5. Tratar Resposta com Erro

Listagem 39 - *Factory para tratar respostas com erro*

angular1/app/common/factories/handleResponseErrorFactory.js

```
(function () {
    angular.module('primeiraApp').factory('handleResponseError', [
        '$q',
        '$window',
        'consts',
        HandleResponseErrorFactory
    ])

    function HandleResponseErrorFactory($q, $window, consts) {
        function responseError(errorResponse) {
            if (errorResponse.status === 403) {
                localStorage.removeItem(consts.userKey)
                $window.location.href = '/'
            }
            return $q.reject(errorResponse)
        }

        return { responseError }
    }
})()
```

Listagem 40 - *Chamar HandleResponseErrorFactory*

angular1/app/config/routes.js *(lista de dependências)*

```
'$httpProvider',
```

angular1/app/config/routes.js *(parâmetro da função)*

```
$httpProvider
```

angular1/app/config/routes.js *(Adicionar aos interceptors)*

```
$httpProvider.interceptors.push('handleResponseError')
```

Appendix A: Tabela de Códigos

Glossário

JWT

: ...