Mongoose: now, let’s apply all we’ve seen to a ‘To Do List’

****Vamos baixar os starting files da versão 2, para darmos início a esta segunda etapa.****

# ****Configuração inicial****

* Configuração da pasta:

mongod

// open a new window:

cd desktop

cd toDoList-DB

npm install

npm init

atom .

**Neste momento nós temos os itens iniciais salvos como const, e os adicionais são salvos em uma array.**

* Neste módulo, vamos salvar os itens por mongoose, em uma dataBase

npm install mongod

npm install mongoose

**const mongoose = require('mongoose');**

****// const items = ["Buy Food", "Cook Food", "Eat Food"];****

****// const workItems = [];****

****mongoose.connect("mongodb://localhost:27017/toDoListDB", {useNewUrlParser: true});****

* Agora, criamos um schema de itens:

****const itemsSchema = new mongoose.Schema ({****

****name: {****

****type: String,****

****required: [true, "Forgot the name!"]****

****}****

****});****

****const Item = mongoose.model("Item", itemsSchema);****

****// desta forma eu acabei de criar uma 'collection', com o nome 'Item', dentro do DataBase 'toDoListDB', que obedece a formatação do 'itemsSchema' !!****

****//– veja que o mongoose muda automaticamente o nome dado de “Item” para “items"****

* E vamos inserir alguns itens:

**const item1 = new Item({**

**name: "Welcome to your toDoList"**

**});**

**const item2 = new Item({**

**name: "Press the '+' to add itens"**

**});**

**const item3 = new Item({**

**name: "<-- Hit this to delete an item"**

**});**

**const defaultItens = [item1, item2, item3];**

**Item.insertMany(defaultItens, function(err){ //to insert an array, we need to call 'insertMany'**

**if (err){**

**console.log(err);**

**} else{**

**console.log("Sucess!!");**

**}**

**});**

// open a new window:

mongo

show dbs

use toDoListDB

switched to db toDoListDB

show collections

//items

db.items.find()

// it will show my items!!

# ****Rendering Database items to the App.****



* Acima vimos como usar o find(). Vamos ver como fazer isso no app.js:

**// Item.insertMany(defaultItens, function(err){ //to insert an array, we need to call 'insertMany'**

**// if (err){**

**// console.log(err);**

**// } else{**

**// console.log("Sucessfully saved default itens!!");**

**// }**

**// });**

**app.get("/", function(req, res) {**

**// const day = date.getDate();**

**Item.find({}, function(err, items){**

**if (err) {**

**console.log(err);**

**} else {**

**console.log(items);**

**console.log("Abaixo, vejamos apenas seus nomes:");**

**items.forEach(function(item){**

**console.log(item.name);**

**});**

**}**

**});**

**res.render("list", {listTitle: "Today", newListItems: items});**

**});**

* *****Veja: ao dar o run, teremos um erro! Eu vou conseguir o console log, mas os items não serão achados! Para isso, eu preciso renderizá-los:*****

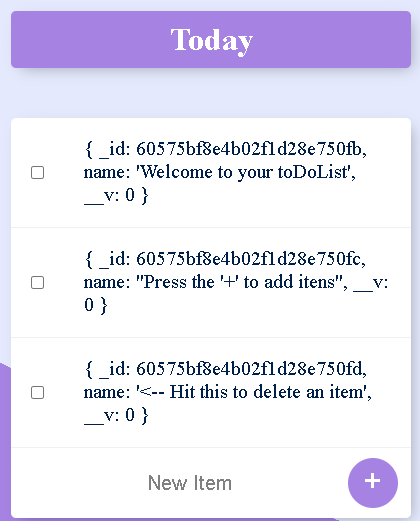
****Item.find({}, function(err, items){****

****res.render("list", {listTitle: "Today", newListItems: items});****

**.**

**.**

**.**

****

* *****Vamos agora mostrar somente os nomes.*****
* *****Dentro da página ‘views/list.ejs’ :*****

**<div class="box">**

**<% for (let i=0; i<newListItems.length; i++) { %>**

**<div class="item">**

**<input type="checkbox">**

****<p><%= newListItems[i].name %></p>****

**</div>**

**<% } %>**

**<form class="item" action="/" method="post">**

**<input type="text" name="newItem" placeholder="New Item" autocomplete="off">**

**<button type="submit" name="list">+</button>**

**</form>**

**</div>**

**Com o código acima conseguimos mostrar apenas os nomes!**

* *****Ou se preferir, posso aperfeiçoar meu código e chamar um ‘forEach()”, instead of “ for() ” loop*****

**<% newListItems.forEach(function(item){ %>**

**<div class="item">**

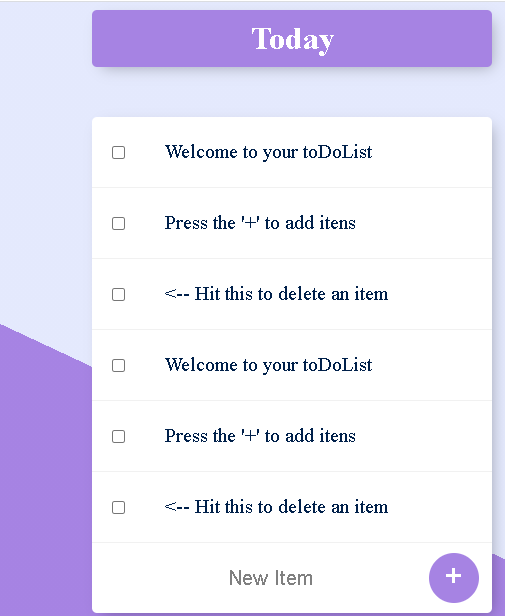
**<input type="checkbox">**

**<p><%= item.name %></p>**

**</div>**

**<% }); %>**

* *****Now, what if accidentally I forget to comment out the ‘insert’ method?*****

********

* *****Let’s first delete all of the, into Mongo Shell:*****

show dbs

admin 0.000GB

config 0.000GB

fruitsDB 0.000GB

listadaPapelariaDB 0.000GB

local 0.000GB

papelaria 0.000GB

shop 0.000GB

toDoListDB 0.000GB

use toDoListDB

switched to db toDoListDB

show collections

items

db.items.find()

{ "\_id" : ObjectId("60575bf8e4b02f1d28e750fb"), "name" : "Welcome to your toDoList", "\_\_v" : 0 }

{ "\_id" : ObjectId("60575bf8e4b02f1d28e750fc"), "name" : "Press the '+' to add itens", "\_\_v" : 0 }

{ "\_id" : ObjectId("60575bf8e4b02f1d28e750fd"), "name" : "<-- Hit this to delete an item", "\_\_v" : 0 }

{ "\_id" : ObjectId("6057b8bf025fd82f443b4e63"), "name" : "Welcome to your toDoList", "\_\_v" : 0 }

{ "\_id" : ObjectId("6057b8bf025fd82f443b4e64"), "name" : "Press the '+' to add itens", "\_\_v" : 0 }

{ "\_id" : ObjectId("6057b8bf025fd82f443b4e65"), "name" : "<-- Hit this to delete an item", "\_\_v" : 0 }

{ "\_id" : ObjectId("6057b93b563e6f205446dd6d"), "name" : "Welcome to your toDoList", "\_\_v" : 0 }

{ "\_id" : ObjectId("6057b93b563e6f205446dd6e"), "name" : "Press the '+' to add itens", "\_\_v" : 0 }

{ "\_id" : ObjectId("6057b93b563e6f205446dd6f"), "name" : "<-- Hit this to delete an item", "\_\_v" : 0 }

db

toDoListDB

db.dropDatabase()

{ "dropped" : "toDoListDB", "ok" : 1 }

show dbs

admin 0.000GB

config 0.000GB

fruitsDB 0.000GB

listadaPapelariaDB 0.000GB

local 0.000GB

papelaria 0.000GB

shop 0.000GB

* **Veja:eu deletei toda a dataBase!**
* **Agora, vamos fazer novamente, inserindo apenas uma vez o conteúdo:**

**app.get("/", function(req, res)**

**{**

**// const day = date.getDate();**

**Item.find({}, function(err, items){**

****if(items.length === 0)****

****{****

****Item.insertMany(defaultItens, function(err){ //to insert an array, we need to call 'insertMany'****

****if (err){****

****console.log(err);****

****} else{****

****console.log("Sucessfully saved default itens!!");****

****}****

****});****

****res.redirect("/");**** // isso irá redirecinar para a root route e renderizar: veja: se eu não faço isso, a primeira vez não irá renderizar

****} else{****

****res.render("list", {listTitle: "Today", newListItems: items});****

****}****

**if (err) {**

**console.log(err);**

**} else {**

**console.log(items);**

**console.log("Abaixo, vejamos apenas seus nomes:");**

**items.forEach(function(item){**

**console.log(item.name);**

**});**

**}**

**});**

**});**

# ****Adding new items****

**Vamos inserir novos itens na lista:**

****app.post("/", function(req, res) {****

****const itemName = req.body.newItem; //new item is the 'name' propertie of the post input****

****const newItemInserted = new Item({****

****name: itemName****

****});****

****item.save();****

**// if (req.body.list === "Work") {**

**// workItems.push(item);**

**// res.redirect("/work");**

**// } else {**

**// items.push(item);**

**// res.redirect("/");**

**// }**

****});****

# ****Deleting things from the list****

First, if I want that the checkbox do something when trigged, I need to add a form for it. Before, I will configure and apply the css only to those forms that have ‘item’ as a class – form.item {…}.

**form.item {**

**text-align: center;**

**margin-left: 20px;**

**}**

**<% newListItems.forEach(function(item){ %>**

****<form class="" action="/delete" method="post">****

**<div class="item">**

****<input type="checkbox" onchange="this.form.submit()" name="checkDeleted">****

**<p><%= item.name %></p>**

**</div>**

****</form>****

**<% }); %>**

* ****Quando eu coloco uma form, eu consigo acessar dentro do app.js; além disso, eu adicionei o código - “onchange= ‘this.form.submit()’ name = ‘checkDeleted’ ” – para eu poder dar o upload e trabalhar/localizar o post dentro do app.js****
* ****Agora, veja, com a linha de código abaixo eu recebo ‘on’ para os itens que foram ‘checkados’ :****

**app.post("/delete", function(req, res) {**

**console.log(req.body);**

**res.redirect("/");**

**});**

* ****Vou aperfeiçoar ainda mais meu código:****

**...**

**<input type="checkbox" **value="<%= item.\_id %>"** onchange="this.form.submit()" name="checkDeleted">**

**…**

{ checkDeleted: '6057bb5b76db38145010f2e0' }

* ****Ao dar o console.log(), o output acima é mostrado! Ou seja, consigo acessar o item pela sua \_id, que é única!****

# ****Creating custom Lists****

What if we had multiple lists ?

****app.get("/posts/:postName", function(req, res) {****

****//let find = historias.find(x => x.title === req.params.postName);****

****console.log(req.params.postName);****

****});****

//se eu escrever no url ‘localhost:3000/home’ ...

// ira ser dado no console ‘home’

* **Agora, iremos criar uma nova variável, que possui nome ( lista do trabalho, lista de casa, etc...), e possui itens em sua lista**

****const listSchema = {****

****name: String,****

****items: [itemsSchema]****

****}****

****const List = mongoose.model("List", listSchema);****

**...**

****app.get("/:postName", function(req, res) {****

****const listName = req.params.postName;****

****const list = new List({****

****name: listName,****

****items: defaultItens****

****});****

****list.save();****

****//let find = historias.find(x => x.title === req.params.postName);****

****console.log(req.params.postName);****

****});****

**Dentro do mongoShell, eu agora consigo ver que eu criei uma nova ‘collection’ !!**

e melhor: se eu tentar acessar /trabalho e /estudos ,foi criado uma database para elas, e é mostrado o seguinte:

show collections

items

lists

db.lists.find()

{ "\_id" : ObjectId("605a55a97c727a25ec40ea8c"), "name" : "trabalho", "items" : [ { "\_id" : ObjectId("605a55a97c727a25ec40ea89"), "name" : "Welcome to your toDoList" }, { "\_id" : ObjectId("605a55a97c727a25ec40ea8a"), "name" : "Press the '+' to add itens" }, { "\_id" : ObjectId("605a55a97c727a25ec40ea8b"), "name" : "<-- Hit this to delete an item" } ], "\_\_v" : 0 }

{ "\_id" : ObjectId("605a55cf7c727a25ec40ea90"), "name" : "estudos", "items" : [ { "\_id" : ObjectId("605a55a97c727a25ec40ea89"), "name" : "Welcome to your toDoList" }, { "\_id" : ObjectId("605a55a97c727a25ec40ea8a"), "name" : "Press the '+' to add itens" }, { "\_id" : ObjectId("605a55a97c727a25ec40ea8b"), "name" : "<-- Hit this to delete an item" } ], "\_\_v" : 0 }

Veja: se eu tentar entrar novamente em “localhost:3000/trabalho”, irá ser criado novamente uma database... como consertar isso ??

****app.get("/:postName", function(req, res) {****

****const listName = req.params.postName;****

****List.findOne({name: listName}, function(err, result){****

//the result should be a DOCUMENT

****if(!err){****

****if(result){****

****console.log("This list already exists!");****

****}****

****else {****

****console.log("nova lista!");****

****// criar a nova lista:****

****const list = new List({****

****name: listName,****

****items: defaultItens****

****});****

****list.save();****

****}****

****}****

****});****

****//let find = historias.find(x => x.title === req.params.postName);****

****console.log(req.params.postName);****

****});****

* Só tem mais um problema.... vamos redirecionar para algum lugar, no caso, para uma página renderizada com a info que eu quero:

**app.get("/:postName", function(req, res) {**

**const listName = req.params.postName;**

**console.log(req.params.postName);**

**List.findOne({name: listName}, function(err, result){ //the result should be a DOCUMENT**

**if(!err){**

**if(result){**

**console.log("This list already exists!");**

****res.render("list", {****

****listTitle: result.name,****

****newListItems: result.items****

****});****

**}**

**else {**

**console.log("nova lista!");**

**// criar a nova lista:**

**const list = new List({**

**name: listName,**

**items: defaultItens**

**});**

**list.save();**

****res.render("list", {****

****listTitle: list.name,****

****newListItems: list.items****

Ou posso escrever da seguinte forma:

// res.render("list", {

// listTitle: list.name,

// newListItems: list.items

// });

**res.redirect("/" + listName);**

// olha que legal! eu mando ele voltar pro "app.get("/:postName", function(req, res) { ...", e agora será possível renderizar, pois terá um resultado!

****});****

**}**

**}**

**else{**

**consle.log(err);**

**}**

**});**

**});**

db.lists.drop()

// vamos deletar a collection

# **Separando as listas**

At this point, every time we try to access a new list, we create an item, that has a name, and an array of to do items. That occurs because in our list.ejs we have:

****<form class="item" action="/" method="post">****

**<input type="text" name="newItem" placeholder="New Item" autocomplete="off">**

**<button type="submit" name="list">+</button>**

**</form>**

**All the ‘post’ is done to the action “/”.**

* **We need to characterize the post method:**

**<form class="item" action="/" method="post">**

**<input type="text" name="newItem" placeholder="New Item" autocomplete="off">**

**<button type="submit" name="list" **value="<%= listTitle %>**">+</button>**

**</form>**

* ****Now, we can see it on the *app.js*****

****const itemName = req.body.newItem; //new item is the 'name' propertie of the post input****

****const listName = req.body.list;****

**// we locate something from its name, and the 'const' assume its value !**

\*\* remember: we have 02 schemas (that are converted to collections on DB ):

1. items – has a string Name
2. lists – has a string Name, and a itemsSchema ITEMS

Now, I want to Choose where to save my items :

****app.post("/", function(req, res) {****

****const itemName = req.body.newItem; //new item is the 'name' propertie of the post input****

****const listName = req.body.list;****

****// we locate something from its name, and the 'const' assume its value !****

****const newItemInserted = new Item({****

****name: itemName****

****});****

****if(listName === "Today") { // on this case, we are at "/" default page****

****newItemInserted.save();****

****res.redirect("/");****

****}****

****else****

****{****

****List.findOne({name: listName}, function(err, foundList){****

****foundList.items.push(newItemInserted);****

****foundList.save();****

****res.redirect("/" + listName);****

****});****

****}****

****});****

# ****Deleting from the root page****

At this point, when I delete something from a page different of the ‘home’, I don’t delete the tem, and I’m redirected to the root route

**Item.findByIdAndRemove(checkedItemToDelete, function(err) { //only executes when you have a callback function**

****The line above don’t find the especific list.****

* ****Inside the list.ejs I write…****

**<form class="" action="/delete" method="post">**

**<div class="item">**

**<input type="checkbox" value="<%= item.\_id %>" onchange="this.form.submit()" name="checkDeleted">**

**<p><%= item.name %></p>**

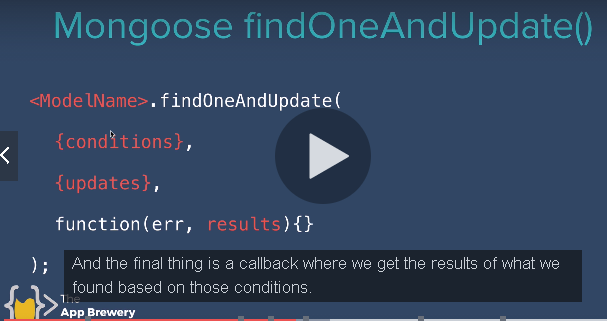
**</div>**

****<input type="hidden" name="listName" value="<%= listTitle %>"></input>****

**</form>**

**// the ‘hidden’ input helps me to access some data value in the app.js. see, it’s like a control tool that I have. Now, inside the ‘app.post(“/delete” …) I can access the list name I am in to.**

* Now, I can see where am I… but… I have to look into each item array and search for an id Item. It’s like, I have several corridors and I need to find the milk… I have to run into every single one… How do I do it?
* Lets see what the ‘pull()’ MONGO operator does….
* Or the ‘findOneAndUpdate()



* The $pull() will be my ‘update()’ method
* And that’s how we do it!

****app.post("/delete", function(req, res) {****

****const checkedItemToDelete = req.body.checkDeleted;****

****const checkedListName = req.body.listName;****

****if (checkedListName === "Today")****

****{****

****Item.findByIdAndRemove(checkedItemToDelete, function(err) { //only executes when you have a callback function****

****if (err) {****

****console.log(err);****

****} else {****

****console.log("Sucessfully deleted item!!");****

****res.redirect("/");****

****}****

****});****

****}****

****else****

****{****

****List.findOneAndUpdate(****

****{name: checkedListName},****

****{$pull:****

//I have several Lists... I will find the one with the same name that I am currently in

****{items:****

// Once I find it, that list has two docs: 'name', and 'items' - rememnber the Schemas - I want to pull from the 'items'

****{\_id: checkedItemToDelete}****

// I want to pull from the 'items' the item that has the particular '\_id'

****}****

****},****

****function(err, foundList) {****

****if (!err){****

****res.redirect("/" + checkedListName);****

****}****

****}****

****)****

****}****

****});****

# ****Lodash: Correcting the names:****

‘/home’ and “/Home” will create totally different lists!

npm install lodash

db.lists.drop()

****const \_ = require("lodash");****

**app.get("/:postName", function(req, res) {**

**…**

****const listName = \_.capitalize(req.params.postName);****

**…**