**MongoDB com mongoose no node**

1. Iremos fazer um sistema para agendamento em um calendário.
2. O núcleo do sistema será composta pela biblioteca Calendar.

**INICIANDO O PROJETO E INSTALANDO AS DEPENDÊNCIAS**

1. npm init -y
2. npm install express –save
3. npm install ejs –save
4. npm install body-parser –save
5. npm install mongoose –save
6. npm install nodemon --save

**SETUP PADRÃO NO INDEX.JS**

1. Na pasta do projeto crie as pastas view e public.
2. Na pasta do projeto crie o arquivo index.js
3. Faça as configurações abaixo
4. //importando o express
5. *const* express = require('express');
6. //importando o body-parser para formulário e json
7. *const* bodyParser = require('body-parser');
8. //importando o mongoose para utilizar o banco de dados mongodb
9. *const* mongoose = require('mongoose');
10. //criando a instancia do express em um app
11. *const* app = express();
12. //configurando o uso de arquivos estáticos para a página do calendar utilizar
13. //o css e o javascript
14. app.use(express.static('public'));
15. //configurano o app pra utilizar o express em formulários e o json
16. app.use(express.urlencoded({extended: false} ));
17. app.use(express.json());
18. //configurando o framework da view engine
19. app.set('view-engine','ejs');
20. //configurando a conexao padrao do mongodb
21. mongoose.connect("mongodb://localhost:27017/agendamento",{useNewUrlParser: true, useUnifiedTopology: true});
22. //rota principal
23. app.get('/',(*req*, *res*)*=>*{
24. *res*.send("Olá!");
25. })
26. //iniciando o servidor
27. app.listen(8080,()*=>*{
28. console.log("Servidor rodando");
29. })

**CRIANDO O BANCO DE DADOS NO MONGODB**

1. Vamos criar o banco de dados chamado agendamento
2. Abra o cmd, digite mongo
3. Digite : use agendamento.

**INSTALANDO O FULLCALLENDAR NO PROJETO**

1. Vá na página <http://fullcalendar.io>
2. Entre em Get Started
3. Vá em Pre-built Bundles
4. Clique “Initialize with Script Tags”
5. Clique dowload: “fullcalendar-5.10.1.zip”
6. Após baixar copie todo conteúdo da pasta lib para dentro da pasta public do projeto.

**CRIANDO A VIEW DE CRIAÇÃO DE CONSULTAS**

1. Na pasta view crie o arquivo create.ejs
2. Configure a estrutura básica do html:5
3. <!DOCTYPE html>
4. <html lang="en">
5. <head>
6. <meta charset="UTF-8">
7. <meta http-equiv="X-UA-Compatible" content="IE=edge">
8. <meta name="viewport" content="width=device-width, initial-scale=1.0">
9. <title>Cadastro</title>
10. </head>
11. <body>
13. </body>
14. </html>
15. Vamos colocar o bootstrap na página através do cdn, copie para o head o link do cdn do bootstrap.
16. Copie também os links do pooper e jquery e cole abaixo do body.
17. <!DOCTYPE html>
18. <html lang="en">
19. <head>
20. <meta charset="UTF-8">
21. <meta http-equiv="X-UA-Compatible" content="IE=edge">
22. <meta name="viewport" content="width=device-width, initial-scale=1.0">
23. <title>Cadastro</title>
24. <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
26. </head>
27. <body>
29. </body>
30. <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js" integrity="sha256-u7e5khyithlIdTpu22PHhENmPcRdFiHRjhAuHcs05RI=" crossorigin="anonymous"></script>
31. <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>
32. <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery.maskedinput/1.4.1/jquery.maskedinput.min.js" type="text/javascript"></script>
34. </html>

**CRIANDO FORMULÁRIO DE CADASTRO**

1. <body>
2. <h1 class="content">Cadastro de consulta</h1>
3. <hr>
4. <form class="form">
5. <input type="text" name="name" id="name" placeholder="Nome do paciente" class="form-control">
6. <input type="text" name="email" id="email" placeholder="email@email.com" class="form-control">
7. <input type="text" name="cpf" id="cpf" placeholder="CPF" class="form-control">
8. <input type="text" name="description" id="description" placeholder="Descrição" class="form-control">
9. <input type="date" name="date" id="date" class="form-control" placeholder="01/01/2012">
10. <input type="time" name="time" id="time" class="form-control">
11. <button class="btn btn-success btn-block">Agendar</button>
12. </form>

**CRIANDO MÁSCARAS PARA CAMPOS DE FORMULÁRIOS**

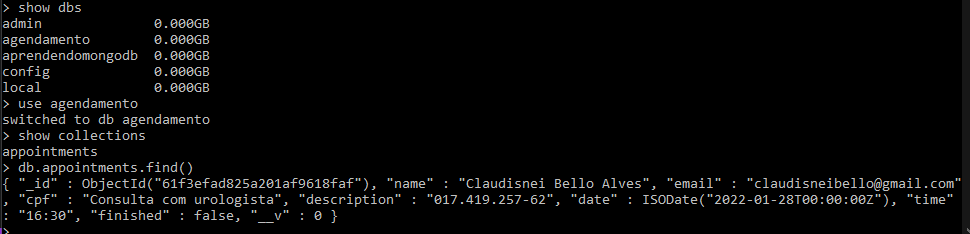
1. Iremos utilizar um plugin jquery-mask-plugin, entre no endereço <https://igorescobar.github.io/jQuery-Mask-Plugin/docs.html>.
2. Clique em download now. E iremos baixar um zip.
3. Na pasta dist vamos copiar somente o arquivo jquery.mask.min e colar na pasta public do projeto.
4. Na view onde a mascara será utilizada devemos criar um script importando o jquery-mask.
5. Crie outra tag script e dentro dela vamos configurar as mácaras conforme modelos do endereço citado acima.
6. Crie um script também para importar as máscaras de input do jquery.
7. <!--Inicio Script para utilizar máscaras no formulário-->
9. <script scr="../public/jquery.mask.min.js"></script>
10. <script>
11. $(document).ready(*function*(*$*){
12. *$*('.cpf').mask('999.999.999-99');
13. });
14. </script>
15. <!--Fim Script para utilizar máscaras no formulário    , {reverse: true}); -->

**CRIANDO O MODEL DE CONSULTA MÉDICA**

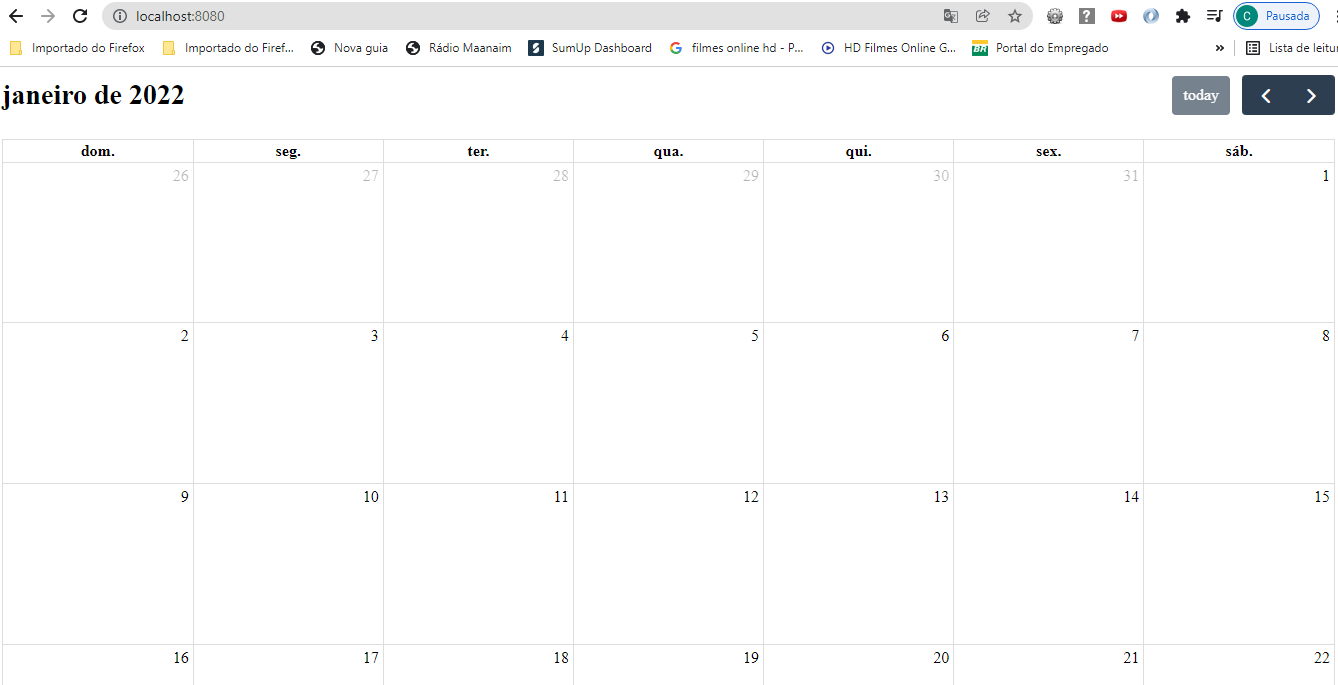
1. Crie a pasta models e um arquivo chamado Appointment.js.
2. Neste arquivo importe o moongose e crie o sckema para o model.
3. *const* mongoose = require('mongoose');
4. *const* appointment = new mongoose.Schema({
5. name: *String*,
6. email: *String*,
7. cpf: *String*,
8. description: *String*,
9. date: *Date*,
10. time: *String*,
11. finished: *Boolean*
12. });
13. module.exports = appointment;

**CRIANDO O SERVIÇE**

1. Iremos utilizar a arquitetura de services.
2. Vamos criar a pasta services e um arquivo chamado AppointmentService.js
3. Esse novo arquivo será uma classe com métodos.
4. Crie uma variável importando o schema Appointment.
5. Crie a variável mongoose que irá receber o ‘mongoose’.
6. *var* appointment = require('../models/Appointment');
7. *var* mongoose = require('mongoose');
8. crie o model AppModel, onde passamos o nome da tabela que será criada e o schema
9. *const* AppoModel = mongoose.model("Appointment",appointment);
10. crie a classe AppointmentService com um método assíncrono ‘create’ que irá receber todos os campos do formulário.
11. Dentro deste método crie uma variável newAppo que irá receber o model criado AppModel. Passamos todos os valores dos inputs como parâmetros para o AppModel.
12. Chame o método save do ‘model’ newApp.
13. Como o método é assíncrono, vamos chamar este método com o await.
14. *class* AppointmentService{
15. async create(*name*, *email*, *description*, *cpf*, *date*, *time*){
16. *var* newAppo = new AppoModel({
17. name: *name*;
18. email: *email*,
19. description: *description*,
20. cpf: *cpf*,
21. date: *date*,
22. time: *time*,
23. finished: false
24. });
25. try {
26. await newAppo.save();
27. return true;
28. } catch (error) {
29. console.log(error);
30. return false;
31. }
32. }
33. }
34. *module*.*exports* = new AppointmentService();
35. c
36. vamos importar para o index.js o service que criamos.
37. No arquivo index.js digite
38. //importando o service de consulta medica
39. *const* appointmentService = require('./services/AppointmentService');
40. Vamos criar uma rota para que quando o usuário clicar no botão do formulário os dados sejam salvos no banco de dados.
41. //rota para pegar os dados do formulário e realizar o cadastro no
42. //banco de dados
43. app.post('/create', async (*req*,*res*)*=>*{
45. //fazendo um destruct e capturando as informações passadas via post
46. *var* {name, email, cpf, description, date, time} = *req*.body;

49. *var* status =   await appointmentService.Create(
50. name,
51. email,
52. cpf,
53. description,
54. date,
55. time
56. );
57. if(status){
58. //volta para página inicial
59. *res*.redirect('/');
60. }else{
61. *res*.send("Ocorreu uma falha!");
62. }
63. })
64. c
65. após isso vamos na página de cadastro create.ejs
66. vamos adicionar o method no formulário e o action.
67. <form method="post" action="create">
68. C
69. Rode o servidor entre na página de cadastro e faça um registro, se tudo ocorrer bem, o usuário será direcionado para a página principal .
70. Para verificar se o registro foi salvo no banco de dados entre no cmd digite:
71. Mongo e irá abrir a página de configuração do mongo
72. Show dbs irá abrir os bancos de dados salvos
73. Use agendamento irá abrir o banco de dados que foi criado
74. Show collections irá mostrar a tabela appointments
75. db.appointments.find() irá mostrar todos os registros salvos no banco de dados.
76. 

**SETUP DO FULL CALLENDAR**

1. Vamos criar a página inicial do sistema chamada index.ejs, onde terá o calendário com os apontamentos.
2. Neste arquivo iremos importar os formatos que foram carregados junto da biblioteca do fullcallendar.
3. Vamos criar uma div com um id chamado calendar onde o calendário será exibido.
4. No final da página vamos criar um script neste script vamos criar uma variável chamada elemento que irá receber a div de id calendar.
5. Vamos criar uma variável chamada calendar que receberá uma nova instancia de FullCalendar,
6. Esta instância possui um método chamado Calendar onde passamos como parâmetro a div element e um json com os parâmetros do callendar que neste caso é o formato da view e o idioma.
7. No final nós renderizamos o calendar para exibir o calendário na página.
8. <!DOCTYPE html>
9. <html lang="en">
10. <head>
11. <meta charset="UTF-8">
12. <meta http-equiv="X-UA-Compatible" content="IE=edge">
13. <meta name="viewport" content="width=device-width, initial-scale=1.0">
14. <title>Sistema de Agendamentos</title>
15. <link rel="stylesheet" href="/fullcallendar/main.css">
16. <script src="/fullcallendar/main.js"></script>
17. </head>
18. <body>
19. <div id="calendar"></div>
20. </body>
21. <script>
22. *var* element = document.getElementById('calendar');
23. *var* calendar = new FullCalendar.Calendar(element, {
24. initialView: 'dayGridMonth',
25. locale: 'pt-br'
26. });
27. calendar.render();
28. </script>
29. </html>
30. C
31. Após isso vamos na rota principal do arquivo index.js e alteramos para que seja renderizada a página index.ejs.
32. Feito isso basta entrar na rota principal e o calendário será exibido
33. 
34. C