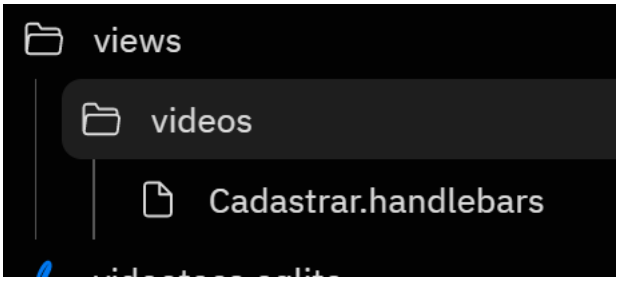
Criando o controller

Vamos começar a criar nossas views. Dentro da pasta views, crie uma pasta chamada vídeos e dentro dela crie um arquivo chamado Cadastrar.handlebars.



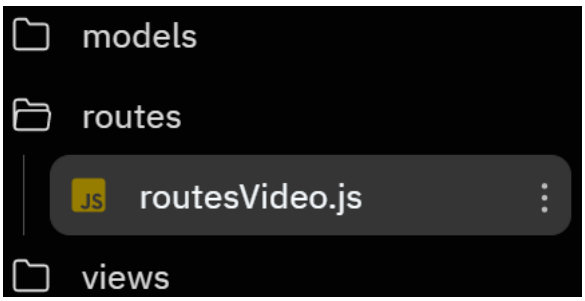
O handlebars, é um módulo que utilizamos para passar parâmetros de uma forma bem mais fácil dentro das views no express. Daqui a pouco vamos ver isso na prática.

Dentro do arquivo cadastrar.handlebars, vamos colocar o código abaixo.

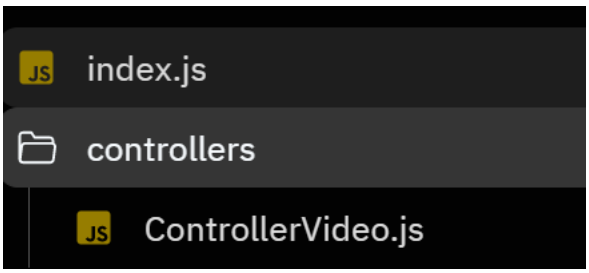
```
<h1 class="text-center my-3">Cadastro de videos Preferidos</h1>
```

Vamos agora, criar uma rota para visualizar nossa view.

Dentro da pasta routes, crie o arquivo **routesVideo.js**



Dentro da nossa pasta controllers, crie o arquivo **ControllerVideo.js.**



Agora trabalharemos com os 3 arquivos.

Vamos começar no arquivo **ControllerVideo.js.**

O controlador vai ser responsável pelo nosso crud.

Vamos criar nossa classe vídeo e nosso cadastrarVideo.

```
const Video = require("../models/Video");

 module.exports = class VideoController {

  static cadastrarVideo(req,res) {
   res.render("videos/Cadastrar");
  }
```

A primeira linha, criamos nossa constante Video do model.

Após utilizamos o module.exports para deixar nossa classe VideoController visível no projeto.

Criamos dentro da classe uma função static cadastrarVideo(req,res). Ela receberá a requisição e retornará uma renderização para a pasta views vídeos/cadastrar.

A página Cadastrar será uma view que incluiremos dentro da pasta views/vídeos.

Agora vamos criar a função **create** nosso **C** do CRUD.

```
static async VideoCreate(req,res) {

    const video = {
      autor: req.body.autor,
      titulo: req.body.titulo,
      assunto: req.body.assunto,
      descricao: req.body.descricao,
      link: req.body.link
    }
    await Video.create(video);
    res.send("Cadastro realizado com sucesso!");

  }
```

Dentro da função VideoCreate, vamos criar uma constante chamada vídeo e passaremos os valores referente aos atributos da tabela vídeos.

O comando await Video.create(vídeo) inseri o registro no banco.

Finalizamos com um retorno res.send com mensagem de cadastro realizado com sucesso para a tela do usuário.

Vamos aproveitar e criar a função **listar** na nossa classe.

```
static async listarVideos(req,res) {

    const video = await Video.findAll({ raw:true })

    res.render("videos/listar", {video});
  }
```

O comando findAll seleciona **todos os objetos** (raw) da tabela e disponibiliza na constante vídeo.

Renderizamos para nossa view listar o valor do “array” com todos os objetos.

Vamos agora realizar o **update.** O update possui duas funções.

```
//UPDATE
 static async UpdateVideo(req,res) {

    const id_video = req.params.id_video;

    const video = await Video.findOne({where: {id_video: id_video}, raw : true})

    res.render("videos/update", {video})

  }

  static async VideoUpdate(req, res) {

    const id_video = req.body.id_video

    const video = {
      autor: req.body.autor,
      titulo: req.body.titulo,
      assunto: req.body.assunto,
      descricao: req.body.descricao,
      link: req.body.link
    }

    await Video.update(video, { where: { id_video:id_video }})

    res.redirect("/")

  }
```

A função **UpdateVideo** vai receber o objeto que será manipulado pelo id_video quando nosso usuário selecionar o objeto na view.

Após a localização do objeto, a função renderiza para a view update os dados para alteração.

A função **VideoUpdate** recebe os valores alterados e atualiza no banco de dados.

Para finalizar nosso crud, vamos criar a função **delete**.

Temos que tomar muito cuidado com a função delete, pois após a exclusão, dificilmente conseguiremos recuperar os dados manipulados.

```
static async removerVideo(req,res) {

    const id_video = req.body.id_video;

    await Video.destroy({ where: { id_video: id_video }})

    res.redirect("/")
  }
```

A função pega o valor do id_evento que foi passado pelo usuário e realiza o destroy com a condição Where que seja igual ao id_video.

Nossa classe controller está pronta.

Veja como ficou o código da classe abaixo.

```
const Video = require("../models/Video");
 module.exports = class VideoController {

  static cadastrarVideo(req,res) {
   res.render("videos/Cadastrar");
  }

  static async VideoCreate(req,res) {

    const video = {
      autor: req.body.autor,
      titulo: req.body.titulo,
      assunto: req.body.assunto,
      descricao: req.body.descricao,
      link: req.body.link
    }
```

```
    await Video.create(video);
    res.send("Cadastro realizado com sucesso!");
    res.redirect("/");

  }
   //LISTAR VIDEOS
 static async listarVideos(req,res) {

    const video = await Video.findAll({ raw:true })

    res.render("videos/listar", {video});
  }
//UPDATE
 static async UpdateVideo(req,res) {

    const id_video = req.params.id_video;

    const video = await Video.findOne({where: {id_video: id_video}, raw : true})

    res.render("videos/update", {video})

  }

  static async VideoUpdate(req, res) {

    const id_video = req.body.id_video

    const video = {
      autor: req.body.autor,
      titulo: req.body.titulo,
      assunto: req.body.assunto,
      descricao: req.body.descricao,
      link: req.body.link
    }

    await Video.update(video, { where: { id_video:id_video }})

    res.redirect("/")

  }
//DELETE
 static async removerVideo(req,res) {
```

```
    const id_video = req.body.id_video;

    await Video.destroy({ where: { id_video: id_video }})

    res.redirect("/")
  }
}
```

Finalizamos nosso controller.