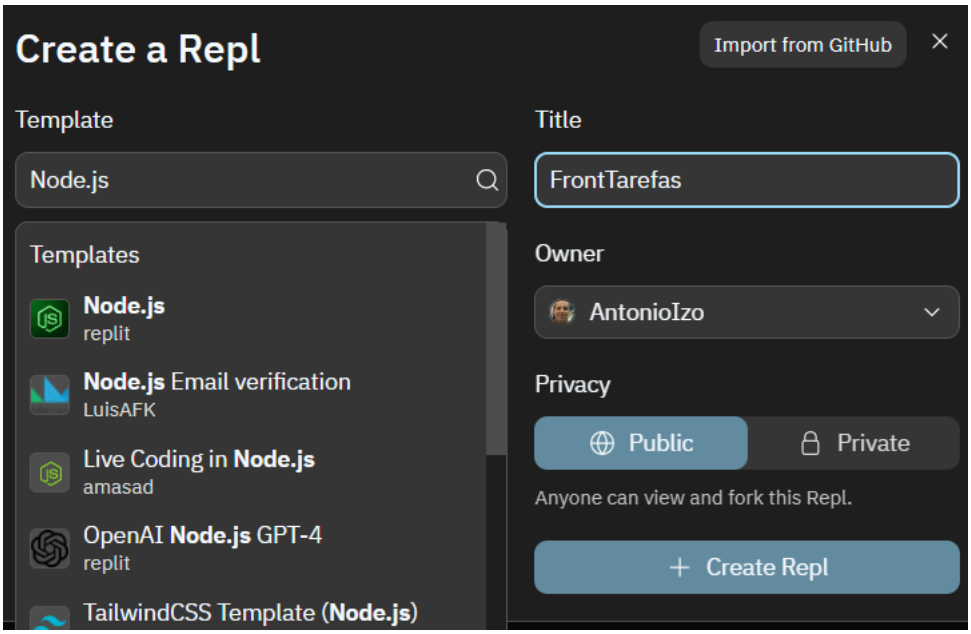
Iniciando a criação da aplicação front-end.

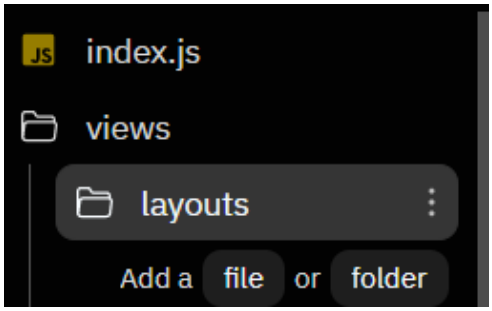
Nesta aula, vamos criar nossa aplicação front-end.

A ideia é que nossa aplicação consuma a API que criamos.

Abra um novo projeto no replit com o **nome de FrontTarefas.**



Vamos criar uma pasta views/layouts para guardar nossas views.



Vamos instalar as bibliotecas que utilizaremos no projeto.

São elas:

_ express

_ handlebars

_ axios

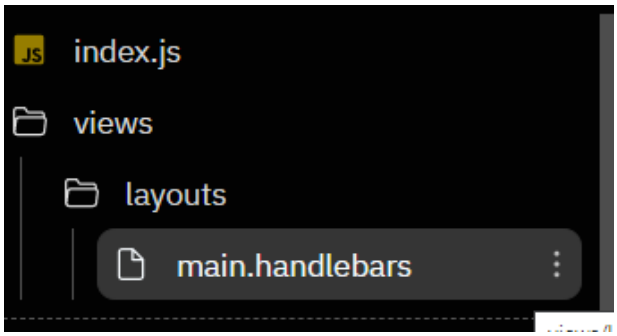
A única biblioteca que ainda não utilizamos é a axios. A axios será responsável pela comunicação com a API.

```
npm install express
```

```
npm install express-handlebars
```

```
npm install axios
```

Na pasta layouts vamos criar o arquivo main.handlebars.

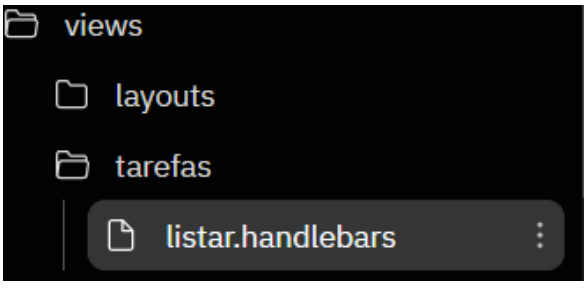


O código do arquivo é mostrado abaixo.

```
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
     {{! BOOSTRAP 5 }}
     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN
9+nJOZ" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQ
SZe" crossorigin="anonymous"></script>
{{!-- ARQUIVOS LOCAL --}}
    <link rel="stylesheet" href="css/styles.css" />
    <title>Sistema de Lista de Tarefas</title>
  </head>
```

```
  <body id="body-mobile">
    <nav class="navbar text-bg-dark">
      <div class="container">
        <span class="navbar-brand text-white">Sistema de Tarefas</span>
        <ul class="navbar nav">
          <li class="nav-item">
          </li>
          <li class="nav-item">
            <a class="nav-link text-white" href="/">Listar Tarefas</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-white" href="/Cadastrar">Cadastrar Tarefa</a>
          </li>
        </ul>
      </div>
    </nav>
    <div class="container">
      {{{body}}}
    </div>
  </body>
</html>
```

Dentro da pasta views, vamos criar uma pasta chamada tarefas e dentro dela vamos criar um arquivo chamado listar.handlebars.



O arquivo listar será responsável por mostrar todas as tarefas criadas.
O modelo do arquivo listar.handlebars é mostrado abaixo.

```
<h1 class="text-center my-3">Lista de tarefas</h1>
<table class="table">
  <div class="row">
    <div class="col-12">
      <thead>
        <th>Código da tarefa</th>
        <th>Título</th>
        <th>Descrição</th>
      </thead>
      <tbody>
        {{#each tarefa}}
          <tr>
            <th>{{this.id_tarefa}}</th>
            <th>{{this.titulo}}</th>
            <th>{{this.descricao}}</th>
            <th><a href="/tarefas/alterar/{{this.id_tarefa}}">Editar<i class="bi bi-pencil-
square"></i></a></th>

            <th> <form class="borde-0 m-0" action="/tarefas/remover" method="POST">
                <input type="hidden" name="id_tarefa" value={{this.id_tarefa}}>
              <input type="submit" value="Excluir" name="Excluir"/>

              </form></th>

          </tr>
        {{/each}}
      </tbody>
    </div>
  </div>
</table>
```

É uma tabela com colunas que mostrarão nossos campos do banco de dados. Coloquei códigos para retorno dos campos da API em vermelho vamos ver cada um deles agora.

{{#each tarefa}} – Cria um loop com o objeto tarefa retornado da nossa função com os valores da API.

{{this.id_tarefa}} – Mostra o valor do campo id_tarefa.

{{this.titulo}} – Mostra o valor do campo titulo.

{{this.descricao}} – Mostra o valor do campo descrição.

<a href="/tarefas/alterar/{{this.id_tarefa}}"> - Envia uma requisição para a página /tarefas/alterar passando o parâmetro do id_tarefa para realizar a alteração.

Criamos também um formulário para passar a opção de remover via method post. Ele chama uma página /tarefas/remover com o valor do id_tarefa.

Vamos criar nossas views update e cadastrar na pasta tarefas.

O arquivo cadastrar.handlebars terá a codificação abaixo.

```
<h1 class="text-center my-3">Cadastro de tarefas</h1>
<form action="/tarefas/Create" method="post">
  <div class="row w-50 d-block m-auto g-3">
    <div class="col-12">
      <label class="form-label" for="titulo">Titulo:</label>
      <input
        class="form-control"
        type="titulo"
        name="titulo"
        id="titulo"
        placeholder="Digite o título da Tarefa"
      />
    </div>
    <div class="col-12">
      <label class="form-label" for="descricao">Descrição:</label>
      <input
        class="form-control"
        type="descricao"
        name="descricao"
        id="descricao"
        placeholder="Digite a descrição da tarefa."
      />
    </div>

    <div>
      <input class="btn btn-primary my-3" type="submit" value="Cadastrar tarefa"
/>
    </div>
  </div>
</form>
```

O arquivo update.handlebars é mostrado abaixo.

```
<h1 class="text-center my-3">Atualizar Tarefa: {{tarefas.titulo}}</h1>
<form action="/tarefas/Update" method="post"">
  <input type="hidden" name="id_tarefa" value="{{tarefas.id_tarefa}}" />
  <div class="row w-50 d-block m-auto g-3">
    <div class="col-12">
      <label class="form-label" for="titulo">Titulo:</label>
      <input
        class="form-control"
        type="titulo"
        name="titulo"
        id="titulo"
        value="{{tarefas.titulo}}"
      />
    </div>
    <div class="col-12">
      <label class="form-label" for="descricao">Descrição:</label>
      <input
        class="form-control"
        type="descricao"
        name="descricao"
        id="descricao"
        value="{{tarefas.descricao}}"
      />
    </div>
    <div>
      <input class="btn btn-primary my-3" type="submit" value="Atualizar tarefa"
/>
    </div>
  </div>
</form>
```

Dentro da pasta views, crie um arquivo chamado mensagem.handlebars.

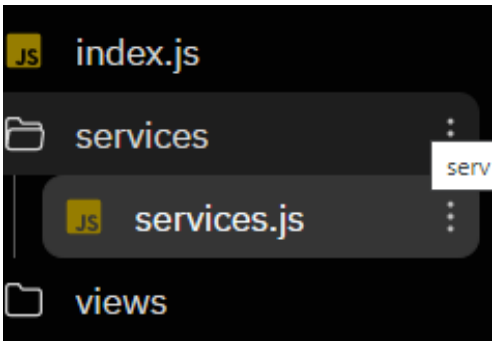
O código será somente uma passagem de parâmetro.

```
<h4>{{mensagem}}</h4>
```

Nossas views foram criadas.

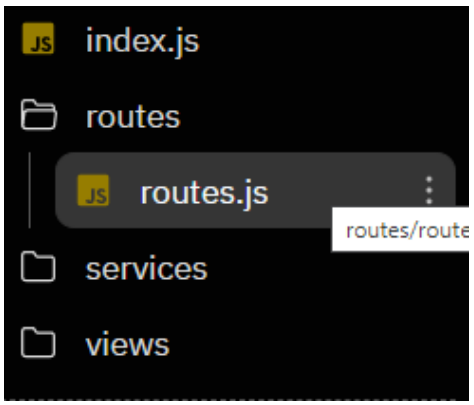
Vamos criar agora uma pasta chamada services e dentro dela um arquivo chamado services.js.

O arquivo services.js será responsável pelo processo de pegar os dados das views e enviar para a API e receber da API e retornar para as views.



Crie também uma pasta routes e um arquivo routes.js

Vamos agora começar a estruturar o arquivo index.js



Vamos estruturar nosso arquivo index agora.

Vamos começar incluindo nossas bibliotecas.

```
const express = require("express");
const app = express();
const hand = require("express-handlebars");
const Services = require("./services/services");
const routes = require("./routes/routes");
```

Após fazemos a configuração da engine, setamos o handlebars e configuramos o retorno para JSON.

```
app.engine("handlebars", hand.engine());
app.set("view engine", "handlebars");
app.use(express.urlencoded({extended: true,}));
app.use(express.json());
```

Vamos setar as rotas para dentro do arquivo routes.js

```
app.use("/", routes);
```

Vamos acrescentar o listen com a porta do servidor.

```
app.listen(3000);
```

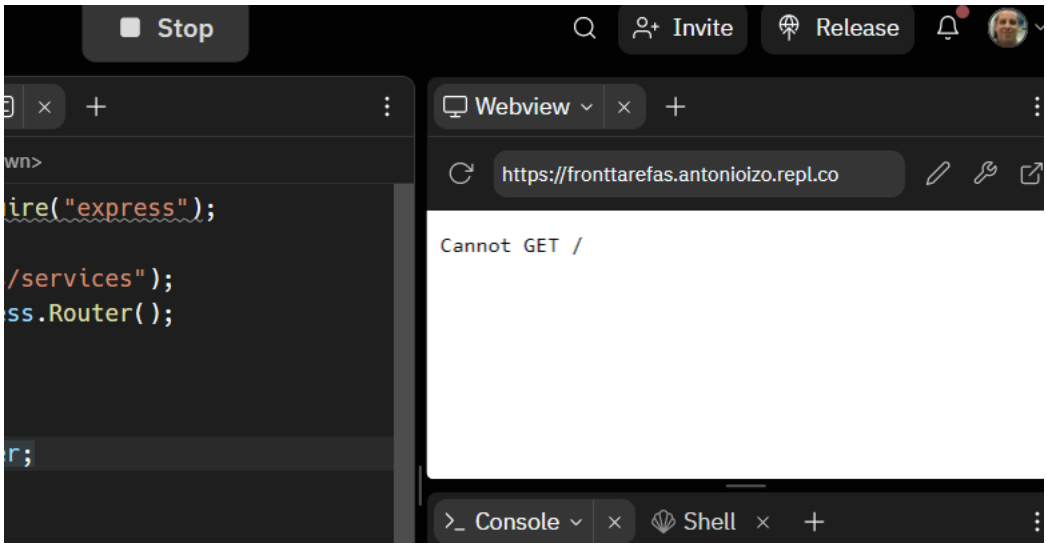
Vamos no arquivo routes.js para iniciar a configuração e exportar o módulo.

```
const express = require("express");
const Services = require("../services/services");
const router = express.Router();
```

Pule algumas linhas e inclua o module.exports=routes;

```
module.exports=router;
```

Vamos executar o projeto e verificar se está tudo correto.



Até agora tudo correto, na próxima aula faremos a configuração das rotas e os services.

até a próxima aula.