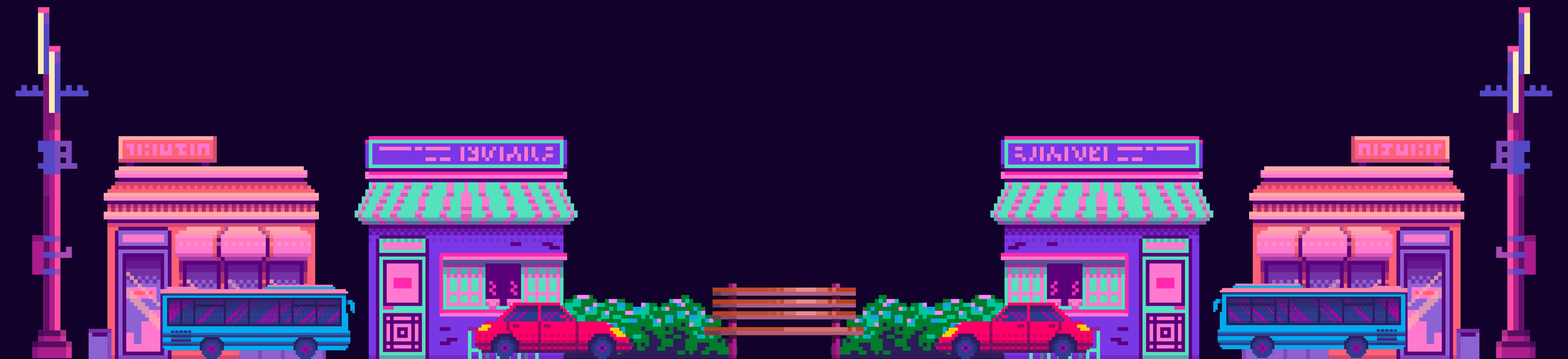

HELLO
WORLD

ROUTER E MVC

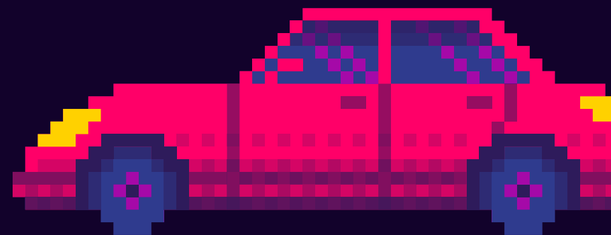


PADRÃO MVC:
MODEL-VIEW-CONTROLLER

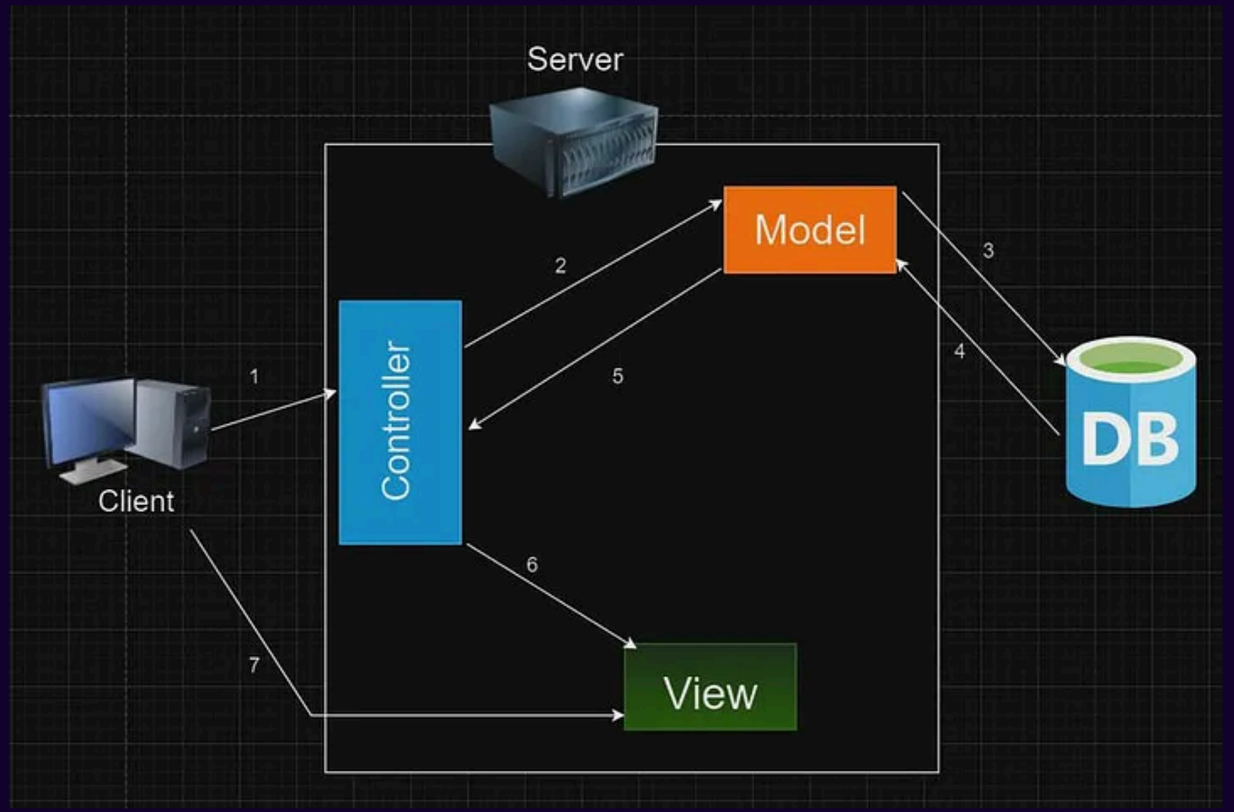
- **Modelo (Model)**: Responsável pela lógica de negócios e pela manipulação dos dados da aplicação. Ele interage com o banco de dados e fornece os dados para a visão.

- **Visão (View):** É a camada responsável pela apresentação dos dados ao usuário. Ela recebe informações do modelo e gera a interface gráfica ou a resposta HTML que será exibida no navegador.

- **Controlador (Controller):** Atua como intermediário entre o modelo e a visão. Ele processa as requisições do cliente, chama os métodos apropriados no modelo e decide qual visão deve ser renderizada como resposta.

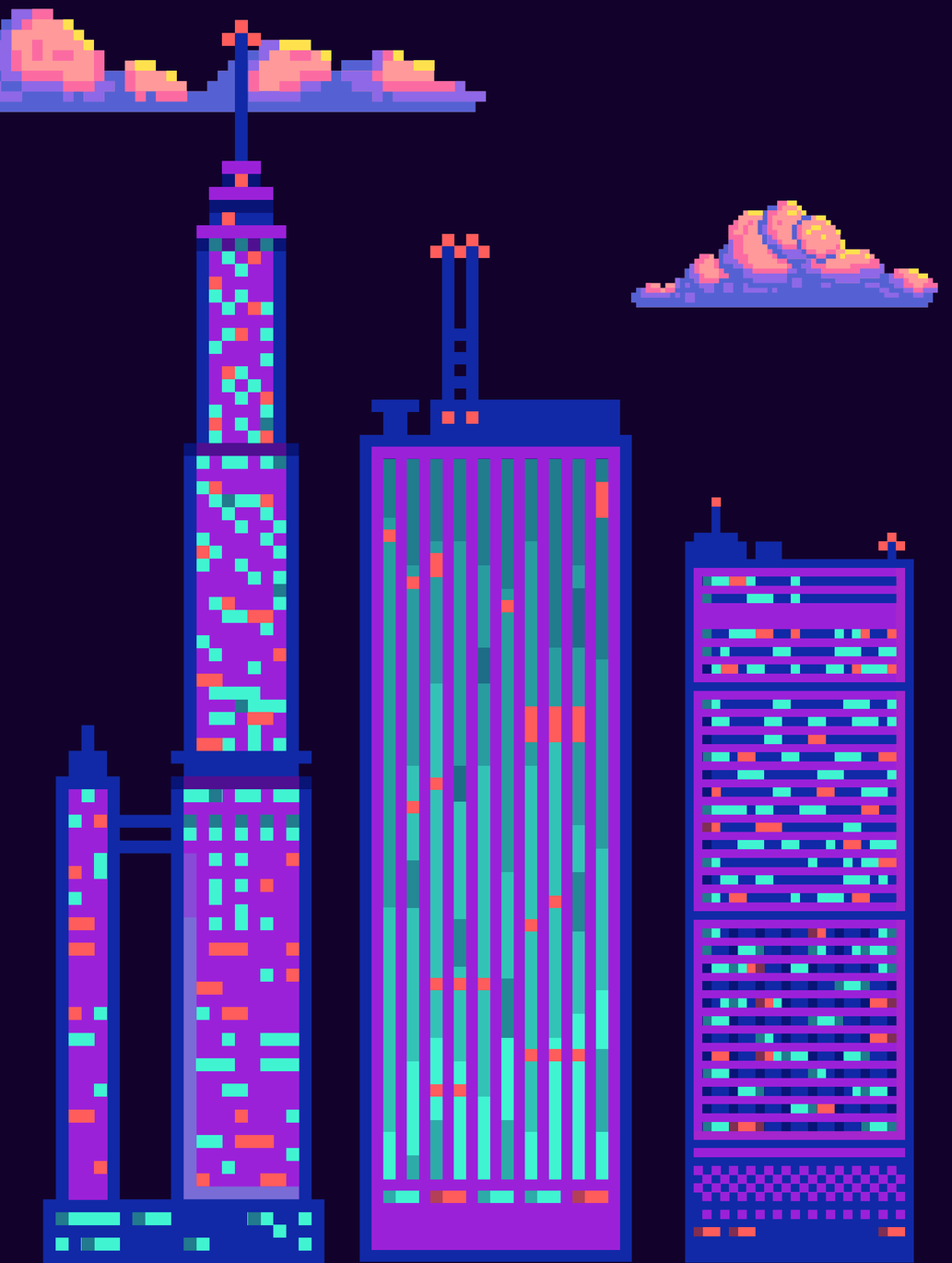
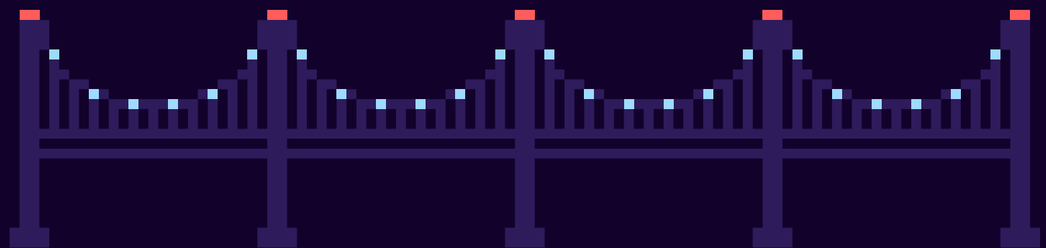
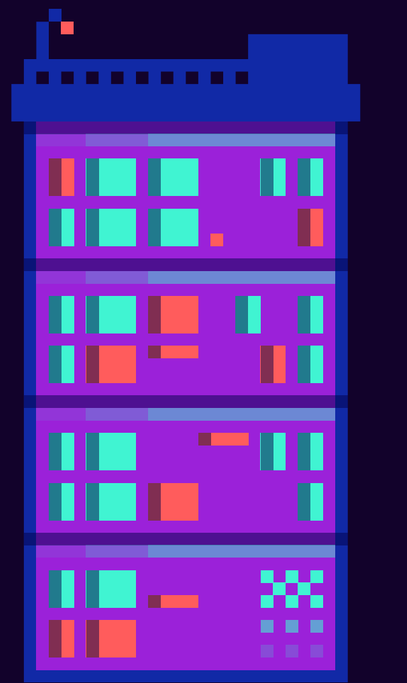


ENTENDENDO



ESTRUTURA DO PROJETO:

```
minha-aplicacao-mvc/
│
├── controllers/
│   └── tarefaController.js
│
├── models/
│   └── tarefaModel.js
│
├── views/
│   ├── index.ejs
│   └── adicionarTarefa.ejs
│
├── public/
│   └── style.css
│
├── app.js
└── package.json
```

ENTENDENDO A ESTRUTURA

01 controllers: Este diretório conterá os controladores da nossa aplicação.

02 models: Aqui ficarão os modelos que representam as entidades da nossa aplicação.

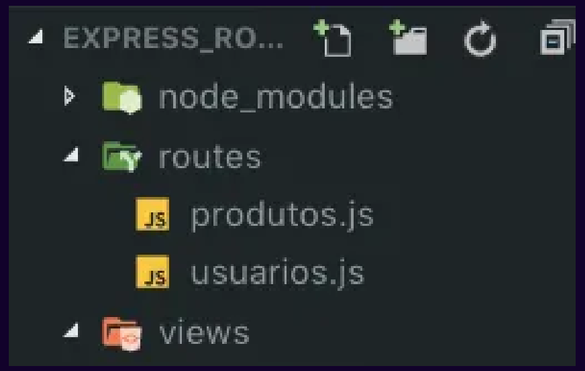
03 views: As visualizações da aplicação, que serão renderizadas e enviadas para o navegador.

04 public: Recursos estáticos, como arquivos CSS, JavaScript e imagens.

05 app.js: O arquivo principal que iniciará o servidor e configurará as rotas da aplicação.

ENTENDENDO O EXPRESS.ROUTER( )

O express.Router nos ajuda a manipular nossas rotas em aplicativos NodeJS, nesse exemplo criaremos rotas de forma modularizada, criando um arquivo para lidar com nossas rotas de produtos e um arquivo para nossa rota de usuarios.



Criamos uma pasta chamada routes e os arquivos produtos.js e usuarios.js, esses arquivos serão os responsaveis por manipular nossas rotas.
A pasta views abriga os arquivos HTML que as rotas irão servir ao serem chamadas, dentro dela há uma pasta produtos e uma pasta usuarios apenas para organizar melhor
o app.js é nosso arquivo principal, o responsavel por criar o servidor e importar nossos arquivos — produtos.js e usuarios.js — com as rotas

CRIANDO O SERVIDOR

01

No arquivo app.js vamos criar nosso servidor, para isso usaremos o seguinte código

```
1 const express = require('express')
2
3 const app = express()
4
5 app.listen(3000, () ⇒ {
6   console.log('Servidor de exemplo aberto na porta: 3000')
7 })
```

CRIANDO NOSSAS ROUTES NO ARQUIVO PRODUTOS.JS

02

No arquivo app.js vamos criar nosso servidor, para isso usaremos o seguinte código

```
const express = require("express");

const router = express.Router();

const path = require("path");

router.get("/", (req, res) => {
  res.sendFile(
    path.join(__dirname, "../", "views", "produtos", "produtos.html")
  );
});

router.get("/add_produto", (req, res) => {
  res.sendFile(
    path.join(__dirname, "../", "views", "produtos", "add_produto.html")
  );
});

module.exports = router;
```

IMPORTANDO EM APP.JS

02

A importação agora é bem simples, criamos uma constante produtosRoutes dando require no arquivo produtos.js e então usamos app.use(produtosRoutes)

```
1  const express = require('express')
2
3  const app = express()
4
5  const produtosRoutes = require('./routes/produtos')
6
7  app.use(produtosRoutes);
8
9  app.listen(3000, () => {
10   console.log('Example app listening on port 3000!')
11 })
```


BORA
CODAR