## **Estrutura de pastas e arquivos do projeto Agendei – API**

AgendeiJS-API/

├── config/

│ ├── db.js

│ └── default.json

├── controllers/

│ ├── appointmentController.js

│ ├── authController.js

│ ├── doctorController.js

│ └── userController.js

├── middleware/

│ ├── authMiddleware.js

│ └── errorHandler.js

├── models/

│ ├── Appointment.js

│ ├── Doctor.js

│ └── User.js

├── routes/

│ ├── appointmentRoutes.js

│ ├── authRoutes.js

│ ├── doctorRoutes.js

│ └── userRoutes.js

├── utils/

│ ├── emailService.js

│ └── logger.js

├── .env

├── .gitignore

├── app.js

├── package.json

├── package-lock.json

└── README.md

**📌 Estrutura e Conexões entre os Arquivos**

A API é responsável pelo backend do sistema **AgendeiJS**, gerenciando autenticação, usuários, médicos e agendamentos. Ela segue um padrão **MVC (Model-View-Controller)** e usa **Express.js** para criação de rotas.

**🔹 Arquivos Principais e Suas Funções**

**1️⃣ Configuração (config/)**

Essa pasta gerencia as configurações gerais do sistema.

* **db.js**
  + Realiza a conexão com o banco de dados (provavelmente **MongoDB** ou **PostgreSQL**).
  + Exporta essa conexão para ser usada nos modelos (models/).
* **default.json**
  + Contém variáveis de ambiente, como:
    - URL do banco de dados.
    - Segredo para autenticação JWT.
    - Porta do servidor.

📌 **Conexões:**

* db.js é chamado no **app.js** para conectar a API ao banco de dados.

**2️⃣ Modelos (models/)**

Define as estruturas dos dados no banco.

* **User.js** → Define os usuários (pacientes ou administradores).
* **Doctor.js** → Define os médicos registrados.
* **Appointment.js** → Representa os agendamentos.

📌 **Conexões:**

* São utilizados pelos controladores em controllers/ para buscar, criar ou alterar dados.
* Interagem com db.js para acessar o banco de dados.

**3️⃣ Controladores (controllers/)**

Implementam a lógica de cada funcionalidade do sistema.

* **authController.js**
  + Realiza login e registra usuários.
  + Gera e valida **tokens JWT** para autenticação.
  + Chamado em authRoutes.js.
* **userController.js**
  + Gerencia usuários: criação, atualização e listagem.
  + Chamado em userRoutes.js.
* **doctorController.js**
  + Gerencia médicos cadastrados.
  + Chamado em doctorRoutes.js.
* **appointmentController.js**
  + Gerencia os agendamentos: criação, atualização e exclusão.
  + Chamado em appointmentRoutes.js.

📌 **Conexões:**

* Chamam os **modelos** (models/) para interagir com o banco.
* São chamados pelas **rotas** (routes/) para processar requisições.

**4️⃣ Rotas (routes/)**

Gerenciam os endpoints da API.

* **authRoutes.js** → Login, logout e registro de usuários.
* **userRoutes.js** → Gerencia usuários do sistema.
* **doctorRoutes.js** → Gerencia médicos.
* **appointmentRoutes.js** → Gerencia agendamentos.

📌 **Conexões:**

* Chamam os **controladores** (controllers/) para processar requisições.
* São incluídos no **app.js** para serem usados na API.

**5️⃣ Middleware (middleware/)**

São funções que interceptam requisições para adicionar segurança e tratamento de erros.

* **authMiddleware.js**
  + Protege rotas privadas verificando tokens JWT.
  + Chamado em rotas protegidas (ex: agendamentos).
* **errorHandler.js**
  + Captura erros e os padroniza em respostas JSON.

📌 **Conexões:**

* authMiddleware.js é usado nas rotas para restringir acesso.
* errorHandler.js é incluído em **app.js** para capturar erros da API.

**6️⃣ Utilitários (utils/)**

Contém funções auxiliares usadas em diferentes partes da API.

* **emailService.js** → Envia emails (exemplo: confirmação de agendamento).
* **logger.js** → Registra logs do sistema.

📌 **Conexões:**

* emailService.js pode ser usado em appointmentController.js.
* logger.js pode ser usado em errorHandler.js para registrar erros.

**7️⃣ Arquivos de Inicialização**

* **app.js**
  + Arquivo principal que inicia o servidor.
  + Configura **Express.js** e carrega as rotas.
* **.env**
  + Armazena variáveis de ambiente sensíveis (ex: senha do banco).
* **package.json**
  + Lista as dependências do projeto.

📌 **Conexões:**

* app.js chama db.js, routes/, middleware/ e inicia o servidor.

**🔗 Fluxo Geral da API**

1. O usuário faz **login** em /api/auth/login.
2. O token JWT é gerado e usado para acessar rotas protegidas.
3. O usuário pode:
   * Criar um agendamento (/api/appointments).
   * Ver médicos cadastrados (/api/doctors).
   * Gerenciar usuários (/api/users).
4. A API se comunica com o **banco de dados** através dos **modelos**.
5. As requisições passam por **middlewares** para autenticação e tratamento de erros.