Versão 3

- Front-End
  - Eclipse - Frontend
- Back-End
  - Swagger
  - Mongo Express
- Banco de Dados
  - Mongo DB
- Build
  - Produção
  - Desenvolvimento
- Artefatos
  - .env
- ENV EasyPanel

Front-End

Front-End

Eclipse - Frontend

Eclipse é um software desenvolvido para a gestão completa do envio, avaliação e acompanhamento de relatórios no âmbito jurídico-administrativo. A plataforma oferece funcionalidades específicas para três tipos de usuários:

- **Defensores:** Criam e submetem triagens e peças processuais.
- **Avaliadores:** Avaliam peças e triagens enviadas.
- **Corregedor:** Gerencia usuários, aprovações e negações, além de supervisionar o sistema.

O sistema também oferece **notas automáticas**, critérios de avaliação e um fluxo para **solicitação de recursos**, promovendo transparência e controle.

Tecnologias

- **Next.js** (App Router)
- **TypeScript**
- **Tailwind CSS**
- **NextAuth (Autenticação)**
- **Docker** (ambiente de desenvolvimento e produção)Usando Docker (ambiente de desenvolvimento)
- **HeroUI**: Biblioteca de componentes visuais modernos e acessíveis (botões, tabelas, modais, etc.), amplamente utilizada na interface do Eclipse para padronizar a experiência do usuário e facilitar/otimizar o desenvolvimento.
- **Tailwind CSS**: Utilizado em conjunto com HeroUI para estilização utilitária e responsiva.

Estrutura de Pastas

```
eclipse/
├── app/                          # Rotas e páginas (App Router)
│   ├── api/auth/[...nextauth]/route.ts   # Rota de autenticação (NextAuth)
│   ├── registrar/                 # Página de registro de usuáros
```

```
│   ├── (user-routes)/              # Rotas específicas para usuários logados
│   └── (usuario)/                  # OTP e redefinição de senha
│
├── components/                     # Componentes utilizados no projeto
│   └── avaliador/                  # Componentes de tela do avaliador
│   └── corregedor/                 # Componentes de tela do corregedor
│   └── defensor/                   # Componentes de tela do defensor
│   └── inicial/                    # Componente de gerenciamento da Home
│   └── sidebar/                    # Componente de gerencialmento da troca entre telas
│
├── providers/                      # Contextos globais (auth, loading, sessão)
│   └── HeroProvider                # Gerenciamento da biblioteca de User Interface que facilita a criação de componentes
│   └── LoadingProvider             # Gerenciamento das funções de loading do software
│   └── ScreenProvider              # Gerenciamento das trocas de componente em tela e seus parâmentros
│   └── SessionProvider             # Gerenciamento das informações de sessão do usuário logado
│
├── public/                         # Arquivos estáticos(Imagens e Icones)
│
├── utils/                          # Funções utilitárias
│   └── NextAuthOptions             # Configurações de autenticação de usuário
│   └── *.tsx                       # Demais utilitários
│
├── Dockerfile.*                    # Dockerfiles para build e produção
├── next.config.ts                  # Configurações do Next.js
└── tailwind.config.js              # Configurações do Tailwind CSS
```

Autenticação

A autenticação é gerenciada por **NextAuth** com a configuração definida em:

```
app/api/auth/[...nextauth]/route.ts
```

Suporta autenticação por e-mail/senha com fluxo de redefinição de senha via token e OTP.

Padrões de Código

- Componentes funcionais com Hooks.
- Organização baseada em papéis de usuário.
- ESLint configurado com suporte a TypeScript.
- Estilização com Tailwind CSS.
- Uso de **HeroUI** para componentes reutilizáveis.

Colaboração

Branches

- `main`: Produção
- `develop`: Desenvolvimento
- `feat--*`: Novas funcionalidades
- `fix--*`: Correções

Funcionalidades em Destaque

- Triagens e submissões por defensores.
- Avaliação e notas automáticas.
- Acompanhamento de histórico.
- Fluxo de aprovação/negação com recurso e/ou usuários do sistema.
- Sistema robusto de autenticação e segurança.

Equipe de Desenvolvimento

- **Desenvolvedores Fullstack:**
  - *João Vitor Viana Chaves*
  - *Gustavo Costa*
  - *Vinícius Jesus*
  - *Guilherme Barbosa*

- **Coordenação Técnica:**
  - *Guilherme*
  - *Layon*

Back-End

Back-End

Swagger

Descrição Técnica do Backend

O backend do projeto é estruturado de forma modular, com separação clara de responsabilidades entre rotas, serviços, middlewares e utilitários. A aplicação é desenvolvida utilizando **Node.js** com **Express** e toda a API está documentada de forma interativa utilizando o **Swagger UI**, acessível pelo endereço:

> “ **http://localhost:3008/api-docs/#/**

Lembre-se de ter o projeto rodando no docker para funcionar!

Estrutura de Diretórios

```
/api/
├── src/
│   ├── docs/              # Configurações e definição dos endpoints no Swagger
│   │   └── swagger.ts
│
│   ├── middlewares/       # ⚙ Middlewares diversos, incluindo o multer para uploads
│   │   └── multer.ts
│
│   ├── router/            # Definição de todas as rotas do sistema organizadas por domínio
│   │   ├── avaliador/
│   │   ├── corregedor/
│   │   ├── defensor/
│   │   ├── usuario/
│   │   └── router.ts      # Roteador principal que importa e agrupa os módulos acima
│
│   ├── services/          # Regras de negócio e funcionalidades para cada tipo de usuário
```

```
│   │   ├── avaliador/
│   │   ├── corregedor/
│   │   ├── defensor/
│   │   └── usuario/
│   │
│   ├── utils/              # 🧰 Utilitários de suporte à aplicação
│   │   ├── opt/            # Envio de OTPs e recuperação de senha
│   │   ├── pdf/            # Operações de manipulação e leitura de PDFs
│   │   ├── validations/    # Validações como token, CPF, e campos diversos
│   │   └── ...             # Outros utilitários adicionais
│   │
│   └── app.tsx             # 🚀 Ponto de entrada da aplicação e inicialização dos serviços
```

Documentação Interativa com Swagger

A API conta com documentação completa utilizando **Swagger**, onde é possível:

- Visualizar **todos os endpoints disponíveis**
- Ver os **parâmetros esperados** por cada rota
- Testar requisições diretamente pela interface
- Visualizar exemplos de **respostas de sucesso e erro**

Exemplo de rotas documentadas:

- `POST /api/defensor/registrar/triagem`
  Registra uma triagem com múltiplos arquivos PDF, processo e protocolo.
- `POST /api/avaliador/pecas/avaliar`
  Permite que um avaliador avalie peças de um defensor.
- `GET /api/usuario/email`
  Retorna as informações do email mascarado.
- `POST /api/corregedor/pecas/aprovar`
  Permite ao corregedor aprovar ou reprovar uma triagem enviada.

Essas e outras rotas estão organizadas por grupos lógicos (Defensor, Avaliador, Corregedor, Usuário) dentro do Swagger UI.

Upload de Arquivos com Multer

A aplicação utiliza o middleware **Multer** para lidar com uploads de arquivos (como PDFs), salvando-os em diretórios específicos com nomes padronizados conforme o campo da requisição.

A lógica para isso está implementada em:

```
src/middlewares/multer.ts
```

Esse middleware identifica dinamicamente o campo do arquivo (`fileRaf`, `fileProcesso_1`, etc.) e direciona o salvamento para a pasta correta, como por exemplo:

- `documentos/raf/`
- `documentos/processos/`
- `documentos/recursos/`
- `documentos/avaliacoes/`
- `documentos/relatorios`

Back-End

Mongo Express

Interface Mongo Express

Além da API documentada via Swagger, o backend também conta com uma interface web para gerenciamento direto do banco de dados MongoDB utilizando o **Mongo Express** — uma ferramenta leve e funcional para visualização e execução de comandos no banco de forma gráfica.

Acesso à Interface Mongo Express

A interface está disponível no seguinte endereço (ajustável conforme o ambiente de execução):

> “ **http://localhost:8081**

Lembre-se de ter o projeto rodando no docker para funcionar!

As credenciais de acesso ao Express são as variáves: MONGO_INITDB_ROOT_USERNAME e MONGO_INITDB_ROOT_PASSWORD dos arquivos env

Através dela é possível:

- Visualizar todas as coleções disponíveis no banco
- Buscar documentos por filtros (queries MongoDB)
- Editar registros diretamente
- Remover dados manualmente (quando necessário para testes ou administração)
- Inserir novos documentos via interface visual
- Exportar dados para backup ou inspeção

Utilização no Contexto do Projeto

A interface do **Mongo Express** é extremamente útil para:

- **Testes de integração e debug** durante o desenvolvimento
- **Administração rápida de dados**, sem a necessidade de linha de comando
- **Verificação de dados persistidos** via requisições realizadas na API
- **Criação manual de documentos**, útil durante o desenvolvimento de novas rotas

---

Segurança

> ⚠ Em ambiente de produção, o Mongo Express **não deve ser exposto publicamente** sem autenticação e/ou encapsulamento por VPN, proxy ou firewall.
> No projeto atual, ele é **ativado apenas em ambientes de desenvolvimento**, sem acesso externo.

Integração com a API

Durante o desenvolvimento, a interface Mongo Express serve como um complemento ao Swagger:

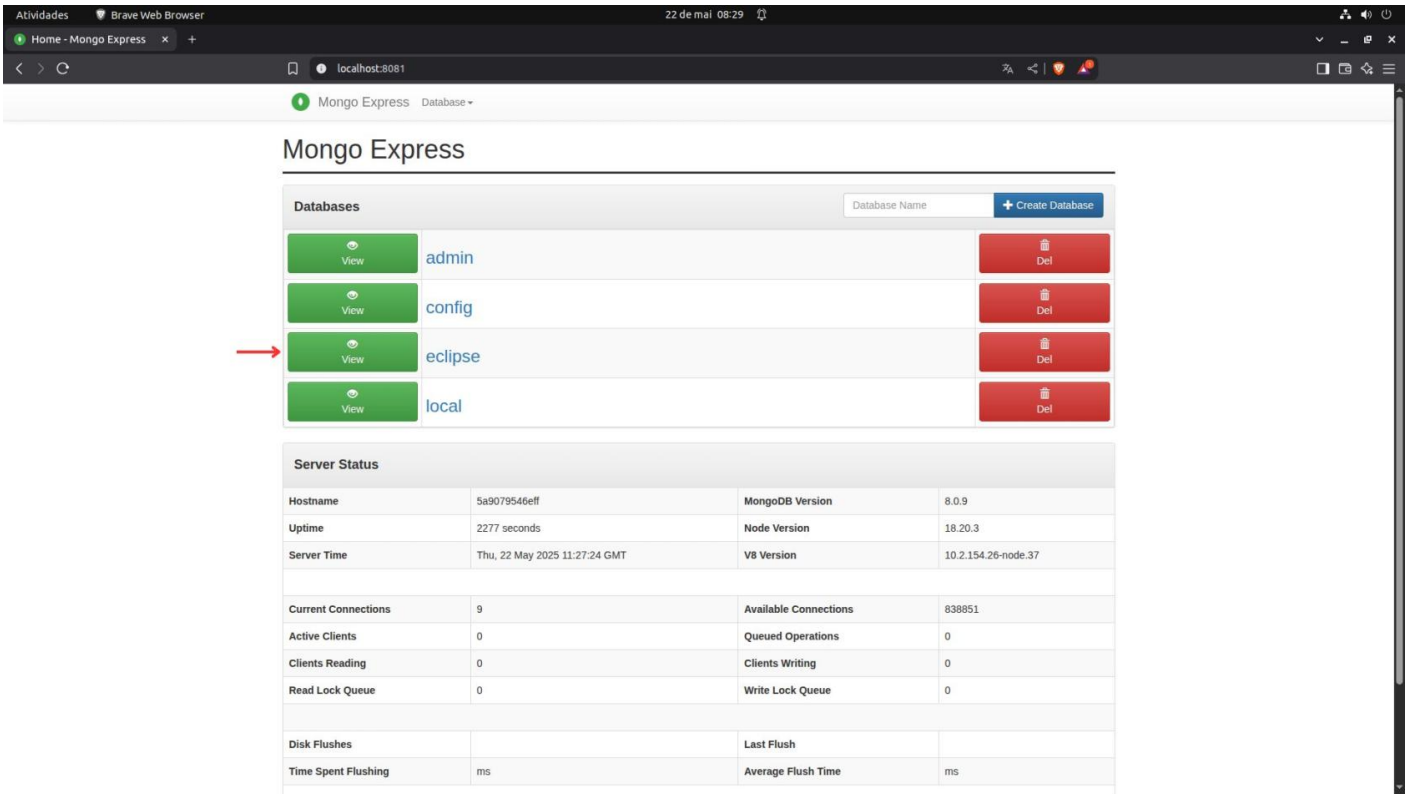
- Você faz uma requisição no **Swagger UI** (como POST `/api/avaliador/pecas/avaliar`)
- E pode **ver o resultado diretamente no Mongo Express**, conferindo como o documento foi persistido em `coleção: Avaliacao`

Exemplos de coleções visualizáveis:

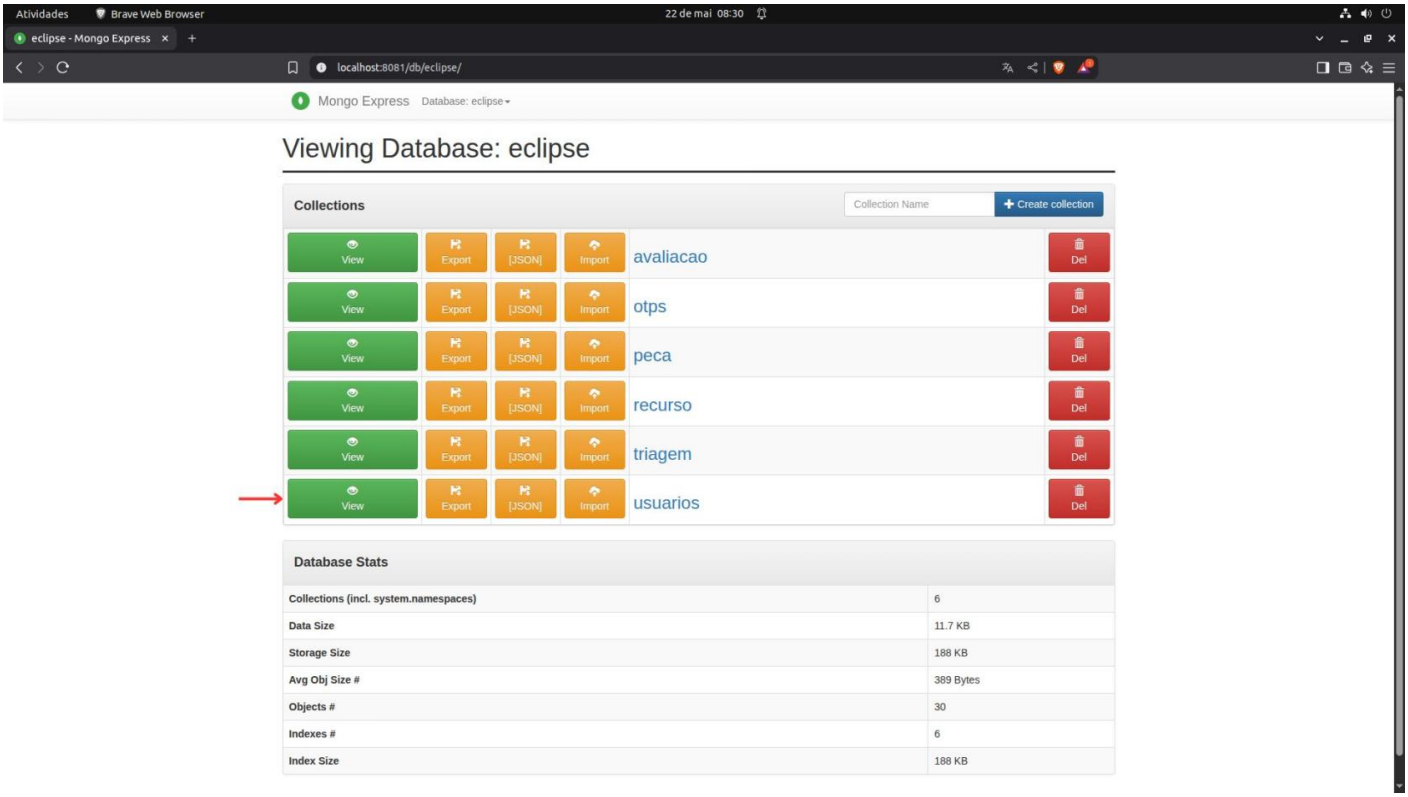
- `Usuarios`
- `Triagem`
- `Peca`
- `Avaliacao`
- `Recurso`
- `Otps`

Passo a Passo de Uso

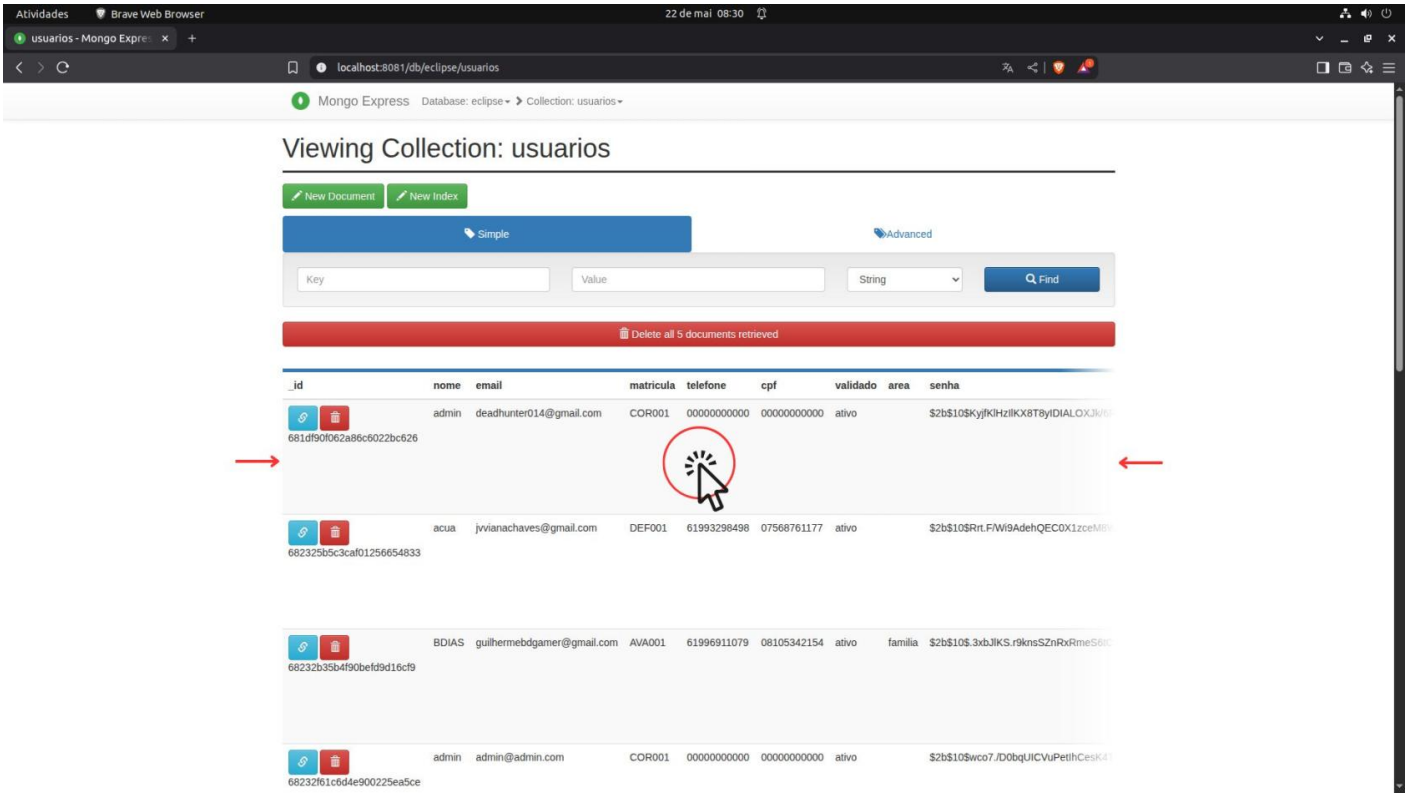
O mongo express criar algumas tabelas por padrão, mas a que estará sendo usada, é a gerada pelo docker com o nome ***eclipse,*** como indicado na seta vermelha abaixo, basta clicar em ***View*** ou no próprio nome do database:



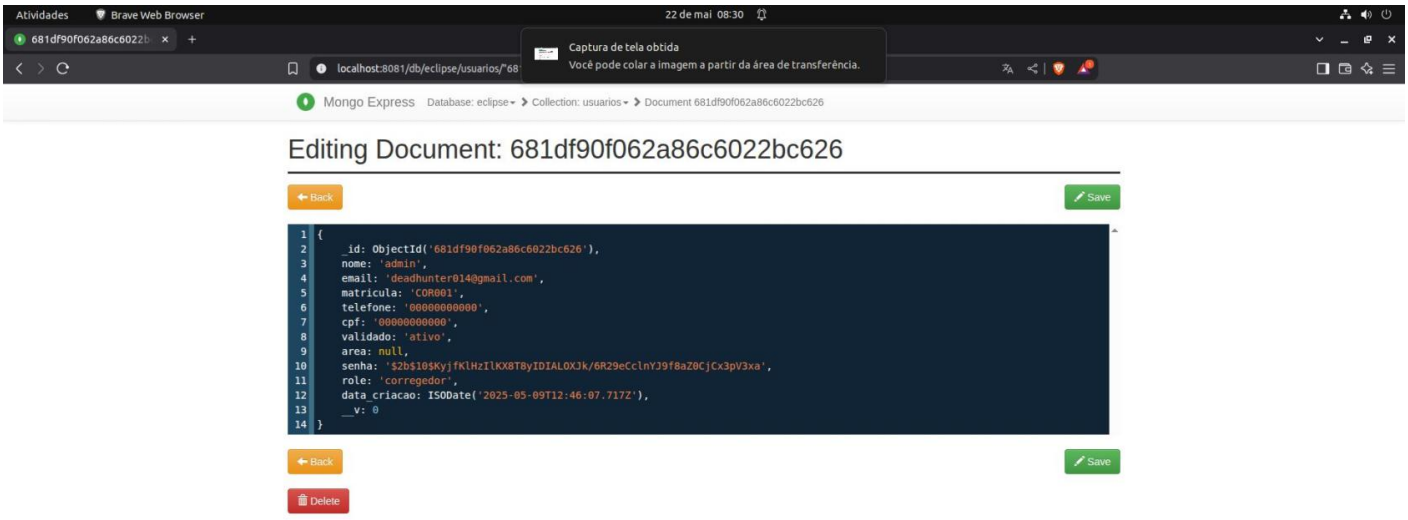
Apos clicar no database **eclipse**, temos acesso as *coleções* (entidades do banco), onde cada respectivo dado se encontra, no exemplo abaixo estaremos mostrando a coleção de ***usuários***, mas você pode repetir o mesmo procedimento para as demais entidades:



Depois de abrir a coleção de ***usuários***, você verá todos os dados cadastrados no banco referentes a essa entidade com todos os campos declarados nas ***Models*** da pasta **api**, como demonstrado abaixo, você pode clicar em cada um dos dados da coleção para abri-la e/ou edita-la:



Aqui você poderá ver todos os atributos e dados do documento do qual clicou, bem como o nome dos campos e valores. Aqui você pode clicar em qualquer campo e alterar seu valor **(Não se esqueça de clicar em *Save* após as alterações)**, e clique em **Back** para voltar para a sessão de coleções:



Banco de Dados

Mongo DB

Configurações do MongoDB e Mongo Express

Para o correto funcionamento do banco de dados **MongoDB** e da interface administrativa **Mongo Express** em ambiente de desenvolvimento, são necessários dois serviços: o **MongoDB** e o **Mongo Express**. A seguir, detalhamos as configurações de cada serviço:

Serviço MongoDB

O MongoDB é configurado como o banco de dados principal. Ele é configurado usando o **Docker**, e as variáveis de ambiente necessárias para a conexão estão definidas nos arquivos `.env.dev`, `.env` e `.env.production` que contém informações sensíveis, como nome do banco de dados e credenciais de acesso.

Configuração do MongoDB **(docker-compose)**

```
services:
  mongo:
    image: mongo
    env_file:
      - .env.dev # Arquivo de variáveis de ambiente
    ports:
      - "27017:27017" # Expondo a porta padrão do MongoDB
    volumes:
      - dbdata-volume:/data/db # Volume persistente para dados
    environment:
      MONGO_INITDB_DATABASE: ${MONGO_INITDB_DATABASE} # Nome do banco de dados
      MONGO_INITDB_ROOT_USERNAME: ${MONGO_INITDB_ROOT_USERNAME} # Nome de usuário admin
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_INITDB_ROOT_PASSWORD} # Senha do usuário admin
```

- **Imagem**: A imagem Docker utilizada é a oficial do MongoDB (`mongo`).
- **Variáveis de Ambiente**: O arquivo `.env.dev` é responsável por fornecer as variáveis de ambiente, como o nome do banco de dados, o usuário admin e a senha.
- **Portas**: O MongoDB estará disponível na porta **27017**, a porta padrão.
- **Volumes**: O volume `dbdata-volume` garante que os dados do banco sejam persistidos entre reinicializações do container.
- **Variáveis de Ambiente**:
  - `MONGO_INITDB_DATABASE`: Define o nome do banco de dados inicial.
  - `MONGO_INITDB_ROOT_USERNAME`: Nome do usuário root/admin do MongoDB.
  - `MONGO_INITDB_ROOT_PASSWORD`: Senha para o usuário root/admin do MongoDB.

Serviço Mongo Express

Mongo Express é uma interface web administrativa para o MongoDB, permitindo que você visualize e manipule dados diretamente via navegador. Ele também está configurado com Docker e se comunica com o banco de dados MongoDB através de uma URL de conexão específica.

Configuração do Mongo Express **(docker-compose)**

```
services:
  mongo-express:
    image: mongo-express
    ports:
      - "8081:8081" # Expondo a porta para acesso via navegador
    environment:
    ME_CONFIG_MONGODB_URL:
mongodb://${MONGO_INITDB_ROOT_USERNAME}:${MONGO_INITDB_ROOT_PASSWORD}@mongo:27017/${MONGO_INITDB_DATABASE}?authSource=admin # URL de conexão com o MongoDB
    ME_CONFIG_BASICAUTH_USERNAME: ${MONGO_INITDB_ROOT_USERNAME} # Usuário de autenticação básica

    ME_CONFIG_BASICAUTH_PASSWORD: ${MONGO_INITDB_ROOT_PASSWORD} # Senha de autenticação básica
```

- **Imagem**: A imagem Docker utilizada é a oficial do Mongo Express (`mongo-express`).

- **Portas**: A interface do Mongo Express estará disponível na porta **8081**.
- **Variáveis de Ambiente**:
  - `ME_CONFIG_MONGODB_URL`: A URL de conexão com o banco MongoDB, que inclui o usuário e a senha definidos no MongoDB, o nome do banco e o endereço do serviço de banco de dados.
  - `ME_CONFIG_BASICAUTH_USERNAME` e `ME_CONFIG_BASICAUTH_PASSWORD`: Configurações de autenticação básica para acessar o Mongo Express. O nome de usuário e a senha

são os mesmos definidos para o MongoDB.

Explicação da URL de Conexão

A URL de conexão `mongodb://${MONGO_INITDB_ROOT_USERNAME}:${MONGO_INITDB_ROOT_PASSWORD}@mongo:27017/${MONGO_INITDB_DATABASE}?authSource=admin` segue a seguinte estrutura:

- **mongodb://**: Protocolo de conexão com o MongoDB.
- **${MONGO_INITDB_ROOT_USERNAME}:${MONGO_INITDB_ROOT_PASSWORD}**: As credenciais de acesso (usuário e senha) que são lidas a partir das variáveis de ambiente definidas.
- **@mongo:27017**: Nome do serviço Docker onde o MongoDB está rodando, seguido pela porta padrão (27017).
- **/${MONGO_INITDB_DATABASE}**: Nome do banco de dados que será acessado.
- **?authSource=admin**: Especifica o banco de dados de autenticação (admin).

> **Exemplo:** mongodb://root:admin123@eclipse-db:27017/eclipse?authSource=admin

Estrutura de Funcionamento do Mongo

Para as configurações do **Mongo** e seus modelos, são essas as patas importantes:

```
/api/
├── src/
    ├── config/ # Configurações da CONNECTION STRING
    │   ├── connectDb.ts
    ├── model/ # Modelos de todas as entidades
    │   ├── avaliacao
    │   ├── otp
    │   ├── peca
    │   ├── recurso
    │   ├── triagem
    │   ├── usuario
    ├── app.tsx # Criação do primerio Corregedor acontece aqui
```

Build

Build

Produção

*Para rodar o projeto em **produção**, é necessários um arquivo de ambiente. que deve possuir o nome "**.env**" e deve ser colocado na pasta ECLIPSE-V3*

> **A falta desse arquivo de ambiente pode acarretar em erros inesperados de build do docker!**

> **Lembre de adicionar este arquivo na pasta /ECLIPSE-V3**

.env:

```
# □□ MONGO (usado pelo db)

MONGO_INITDB_ROOT_USERNAME=root # Nome do usuário root do banco
MONGO_INITDB_ROOT_PASSWORD=Senha-forte  # Gerar senha forte
MONGO_INITDB_DATABASE=eclipse-db-deploy



# □□ API

API_PORT=3008
CONNECTION_STRING=mongodb://root:Senha-forte@eclipsev3-db-deploy:27017/eclipse-db-deploy?authSource=admin



# □□ Segurança

JWT_SECRET=Senha-forte # Necessário gerar uma chave forte
NEXTAUTH_SECRET=Senha-forte # Necessário gerar uma chave forte

# □□ CORS
#Adicionar domínios conforme necessário, separados por vírgula.
#Exemplo: http://dominio1.com,http://dominio2.com,https://eclipse.defensoria.df.gov.br
```

```
CORS_ALLOWED_ORIGINS=http://localhost:3000,http://localhost:5173,http://localhost:80,http://localhost,http://eclipse-api,http://eclipse-api:5173,http://localhost:3008

# □□ Next.js
NEXT_PUBLIC_APP_VERSION=3.3.09.25

NEXT_PUBLIC_API_URL=http://localhost:3008/api # trocar conforme o domínio
API_INTERNAL_URL=http://eclipse-api:3008 # URL interna para comunicação entre serviços
NEXTAUTH_URL=http://localhost:3000 # trocar conforme o domínio

#Exemplo de domínio -> NEXT_PUBLIC_API_URL=https://eclipse.defensoria.df.gov.br/api/v1
#                   -> NEXTAUTH_URL=https://eclipse.defensoria.df.gov.br



# □□ Admin Inicial
BASE_ADMIN_PASSWORD=Senha-forte #Necessário gerar uma senha forte
BASE_ADMIN_EMAIL=admin@admin.com

# ✉ E-mail
EMAIL_USER=admin123@gmail.com
PASSWORD_USER=senha-gerada
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_SECURE=false

# ⚙ Ambiente
NODE_ENV=production

MOBILE_BASE_URL=http://localhost:3000 #env para rota mobile de verificação QRCode
```

Rodando Docker

> **Lembre-se de criar o arquivo .env na pasta raíz!**

```
sudo docker compose up --build
```

AVISOS IMPORTANTES

Não utilize os valores padrões em produção!

Antes de subir o sistema para qualquer ambiente real, substitua os seguintes campos por valores seguros e válidos:

- **Segurança** (*Criptografia*):

  JWT_SECRET e ***NEXTAUTH_SECRET*** → Gere uma **chave segura** com pelo menos 32 caracteres.

- **Corregedor Inicial**

  BASE_ADMIN_EMAIL→ Altere para um e-mail real e controlado por você.
  BASE_ADMIN_PASSWORD→ Crie uma **senha forte**.

- ✉ **Envio de E-mails:**

  EMAIL_USER → Um e-mail **Gmail válido**.
  PASSWORD_USER → A **senha de app** gerada em

  https://myaccount.google.com/apppasswords com a verificação em duas etapas ativada.

- **Atenção com o MongoDB:**

  Se você alterar os valores de **MONGO_INITDB_ROOT_USERNAME**, **MONGO_INITDB_ROOT_PASSWORD** ou **MONGO_INITDB_DATABASE**, lembre-se de atualizar também o valor da **CONNECTION_STRING** *manualmente*.

  O arquivo **.env não** suporta interpolação de variáveis. Por isso, a string de conexão não é gerada automaticamente com base nos valores acima.

Nunca compartilhe arquivos .env com informações sensíveis publicamente (GitHub, fóruns etc.).

Build

Desenvolvimento

*Para rodar o projeto em **desenvolvimento**, é necessário um arquivo de ambiente que deve ser colocado na pasta raiz do projeto com o nome **.env.dev***

> **A falta desse arquivo de ambiente irá acarretar em erro de build do docker!**

.env.dev:

```
# □□ API
API_PORT=3008

# □□ MongoDB
MONGO_INITDB_ROOT_USERNAME=root
MONGO_INITDB_ROOT_PASSWORD=admin123
MONGO_INITDB_DATABASE=eclipse-db
CONNECTION_STRING=mongodb://root:admin123@eclipse-db:27017/eclipse?authSource=admin

# □□ Segurança
JWT_SECRET=secret
NEXTAUTH_SECRET=secret

# □□ CORS
CORS_ALLOWED_ORIGINS=http://localhost:3000,http://localhost:5173,http://localhost:80,http://localhost,http://eclipse-api,http://eclipse-api:5173,http://localhost:3008

# □□ Next.js
NEXT_PUBLIC_APP_VERSION=3.1.05.25
NEXT_PUBLIC_API_URL=http://localhost:3008/api
API_INTERNAL_URL=http://eclipse-api:3008
NEXTAUTH_URL=http://localhost:3000

# □□ Admin Inicial
BASE_ADMIN_PASSWORD=admin123
```

```
BASE_ADMIN_EMAIL=admin@admin.com
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_SECURE=false

# ✉ E-mail
EMAIL_USER=dominio-email@gmail.com
PASSWORD_USER=senha-gerada

# ⚙ Ambiente
NODE_ENV=development
MOBILE_BASE_URL=http://localhost:3000
```

Rodando Docker

> **Lembre-se de criar os arquivos .env na pasta raíz e o .env.production na pasta /eclipse antes!**

```
sudo docker compose -f docker-compose.dev.yaml up --build
```

> **AVISOS IMPORTANTES**

Não utilize os valores padrões em produção!

Antes de subir o sistema para qualquer ambiente real, substitua os seguintes campos por valores seguros e válidos:

- **Segurança** (*Criptografia*):

  JWT_SECRET e ***NEXTAUTH_SECRET*** → Gere uma **chave segura** com pelo menos 32 caracteres.

- **Corregedor Inicial**

  BASE_ADMIN_EMAIL→ Altere para um e-mail real e controlado por você.
  BASE_ADMIN_PASSWORD→ Crie uma **senha forte**.

- ✉ **Envio de E-mails:**

  EMAIL_USER → Um e-mail **Gmail válido**.
  PASSWORD_USER → A **senha de app** gerada em

  https://myaccount.google.com/apppasswords com a verificação em duas etapas ativada.

- ⚠ **Atenção com o MongoDB:**

  Se você alterar os valores de **MONGO_INITDB_ROOT_USERNAME**, **MONGO_INITDB_ROOT_PASSWORD** ou **MONGO_INITDB_DATABASE**, lembre-se de atualizar também o valor da **CONNECTION_STRING** *manualmente*.

  O arquivo **.env não** suporta interpolação de variáveis. Por isso, a string de conexão não é gerada automaticamente com base nos valores acima.

> **Nunca compartilhe arquivos .env com informações sensíveis publicamente (GitHub, fóruns etc.).**

Artefatos

.env

```
# □□ API
API_PORT=3008

# □□ MongoDB
MONGO_INITDB_ROOT_USERNAME=root
MONGO_INITDB_ROOT_PASSWORD=admin123
MONGO_INITDB_DATABASE=eclipse-db-deploy
CONNECTION_STRING=mongodb://root:admin123@eclipse-db-
deploy:27017/eclipse?authSource=admin

# □□ Segurança
JWT_SECRET=secret
NEXTAUTH_SECRET=secret

# □□ CORS
CORS_ALLOWED_ORIGINS=http://localhost:3000,http://localhost:5173,http://localhost:80,http://loca
lhost,http://eclipse-api,http://eclipse-api-deploy:5173

# □□ Next.js
NEXT_PUBLIC_APP_VERSION=3.1.05.25
NEXT_PUBLIC_API_URL=http://localhost:3008/api/v1
NEXTAUTH_URL=http://localhost

# □□ Admin Inicial
BASE_ADMIN_PASSWORD=admin123
BASE_ADMIN_EMAIL=admin@admin.com

# ✉ E-mail
EMAIL_USER=dominio-email@gmail.com
PASSWORD_USER=senha-gerada

# ⚙ Ambiente
NODE_ENV=production
```

ENV EasyPanel

```
# □□ MONGO (usado pelo db)
MONGO_INITDB_ROOT_USERNAME=root
MONGO_INITDB_ROOT_PASSWORD=admin123
MONGO_INITDB_DATABASE=eclipse-db-deploy


# □□ API
API_PORT=3008
CONNECTION_STRING=mongodb://root:admin123@10.233.168.5:27017/eclipse?tls=false&authSou
rce=admin


# □□ Segurança

JWT_SECRET=a7f3c19e6b2d8f40
NEXTAUTH_SECRET=3c9a5d1f0e72b6c8

# □□ CORS
CORS_ALLOWED_ORIGINS=https://eclipse.ljit.com.br,https://eclipseapi.ljit.com.br
#CORS_ALLOW_ALL_ORIGINS=True

# □□ Next.js
NEXT_PUBLIC_APP_VERSION=3.3.09.25

NEXT_PUBLIC_API_URL=https://eclipseapi.ljit.com.br/api
NEXTAUTH_URL=https://eclipse.ljit.com.br


# □□ Admin Inicial
BASE_ADMIN_PASSWORD=dpdf@2024
BASE_ADMIN_EMAIL=admin@admin.com

# ✉ E-mail
EMAIL_USER=admin123@gmail.com
PASSWORD_USER=dpdf@2024
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_SECURE=false

# ⚙ Ambiente
NODE_ENV=production
```