Documentação NosConectados
Back-end

Controllers: 

Routes:

Models: 

Settings: 

dependencies.php:

Este é um arquivo de configuração para contêiner de injeção de dependência (DIC) em um aplicativo PHP construído com o framework Slim. O DIC é usado para gerenciar e injetar dependências em diferentes partes do aplicativo.
A configuração define três serviços:

renderer: Este serviço é responsável por renderizar visualizações. Ele usa a classe PhpRenderer do framework Slim e é configurado com um caminho para o diretório que contém os arquivos de modelo.

logger: Este serviço configura um logger Monolog. Ele é configurado com um nome e caminho de arquivo de log da matriz de configurações e um nível de log.

db: Este serviço configura uma conexão de banco de dados usando o Eloquent ORM do framework Slim. Ele cria uma nova instância do Capsule, adiciona uma conexão do array de configurações, define a instância como global, inicializa o Eloquent e retorna a instância do Capsule.

Cada serviço é definido como um encerramento que retorna uma instância da classe correspondente. O parâmetro $c é a instância do contêiner DIC, que é usado para recuperar outras dependências conforme necessário.
Esse arquivo de configuração normalmente é usado em conjunto com outros arquivos que definem rotas, controladores e outras lógicas de aplicativos. Esses arquivos podem acessar o contêiner DIC para recuperar dependências conforme necessário.

middleware.php:

O primeiro middleware é um middleware de autenticação JWT fornecido pelo pacote Tuupola\Middleware\JwtAuthentication. Ele valida JSON Web Tokens no cabeçalho "Authorization" das solicitações recebidas para o caminho "/api" (ou uma matriz de caminhos incluindo "/api" e "/admin"), usando uma chave secreta das configurações do aplicativo. A opção "ignorar" especifica uma lista de caminhos que devem ser excluídos da autenticação JWT, como login, registro de usuário e endpoints de sensor e listagem de dados.

O segundo middleware é um middleware CORS que adiciona cabeçalhos CORS à resposta, permitindo solicitações entre origens de qualquer origem. O cabeçalho "Access-Control-

Allow-Origin" permite solicitações de qualquer domínio, enquanto os cabeçalhos "Access-Control-Allow-Headers" e "Access-Control-Allow-Methods" especificam quais cabeçalhos e métodos de solicitação são permitidos.

No geral, essas funções de middleware ajudam a proteger e permitir a comunicação entre diferentes partes do aplicativo.

routes.php:

O código define uma rota OPTIONS que corresponde a qualquer rota. Isso é usado para lidar com solicitações de simulação CORS (Compartilhamento de recursos de origem cruzada).

O código então requer vários arquivos PHP que definem rotas para diferentes funcionalidades do aplicativo, como autenticação, sensores, usuários, etc.

Por fim, há uma rota abrangente que corresponde a qualquer método HTTP e a qualquer rota que não tenha sido definida anteriormente. Esta rota é usada para lidar com erros 404, retornando um manipulador padrão de página Slim não encontrada.

No geral, esse código configura o roteamento para um aplicativo Slim com várias rotas definidas e uma rota abrangente para lidar com erros 404.

settings.php:

displayErrorDetails: um booleano que determina se os detalhes do erro são exibidos ou não para o usuário. É definido como verdadeiro somente em desenvolvimento, setamos como falso em produção.

addContentLengthHeader: um booleano que determina se o servidor web tem ou não permissão para enviar o cabeçalho de comprimento de conteúdo. É definido como falso, o que significa que o servidor da Web não tem permissão para enviar esse cabeçalho.

renderer: um array que contém configurações para o renderizador, que é responsável por renderizar os templates HTML. A configuração template_path especifica o caminho para o diretório que contém os modelos HTML.

logger: uma matriz que contém configurações para a biblioteca de registro Monolog, que é usada para registrar eventos e erros no aplicativo. A configuração de nome especifica o nome do criador de logs e a configuração de caminho especifica o caminho para o arquivo de log. A configuração de nível especifica o nível de criação de log, que é definido como depuração.

db: um array que contém configurações para a conexão com o banco de dados. As configurações de driver, host, banco de dados, nome de usuário, senha, charset, agrupamento e prefixo especificam os detalhes da conexão para o banco de dados MySQL. A configuração unix_socket especifica o caminho para o arquivo de soquete do MySQL.

secretKey: uma string que contém uma chave secreta para o aplicativo. Essa chave é usada para fins de criptografia ou autenticação.

No geral, esse arquivo de configuração fornece configurações importantes para um aplicativo da Web, incluindo tratamento de erros, criação de log, conectividade de banco de dados e segurança.

atribuicao.php:

Este código PHP define rotas para a classe AtribuiçãoSensor. Inclui rotas para listar, adicionar, recuperar, atualizar e excluir objetos AtribuicaoSensor. Também inclui uma rota para listar objetos AtribuicaoSensor com objetos Usuario associados.

autenticacao.php:

Este código PHP define rotas para as classes de Usuários e Sensores. Inclui rotas para verificar o login do usuário, gerar o token após acesso bem sucedido, verificar o sensor que está enviando dados, gerando o token caso o sensor seja autenticado.

usuarios.php:

Este código cria um grupo de rotas para o terminal '/api/v1'.

A primeira rota é uma solicitação GET para '/usuários/lista' que retorna uma lista de todos os usuários.

A segunda rota é uma solicitação GET para '/usuarios/nomes' que retorna uma lista de todos os usuários com seus nomes e sobrenomes.

A terceira rota é uma requisição POST para '/usuários/cadastro' que adiciona um novo usuário ao banco de dados. Requer a senha do usuário, isAdmin status, registerConfirmed status, firstName, lastName, phone, email, gender, document, street, numberU, city, state, zipcode, district.

A quarta rota é uma solicitação GET para '/usuários/lista/{id}' que retorna o usuário com o ID especificado.

A quinta rota é uma requisição PUT para '/usuarios/atualiza/{id}' que atualiza o usuário com o ID especificado.

A sexta rota é uma solicitação DELETE para '/usuarios/remove/{id}' que remove o usuário com o ID especificado.

localizacoes_usuarios.php:

Este código cria um grupo de rotas para a API versão 1, que estão relacionadas ao modelo LocalizacaoUsuario.
As rotas são:

GET '/api/v1/localizacoes-users/lista' - Lista todos os objetos LocalizacaoUsuario.

POST '/api/v1/localizacoes-users/adiciona' - Adiciona um novo objeto LocalizacaoUsuario.

GET '/api/v1/localizacoes-users/lista/{id}' - Recupera um objeto LocalizacaoUsuario com o ID especificado.
PUT '/api/v1/localizacoes-users/atualiza/{id}' - Atualiza um objeto LocalizacaoUsuario com o ID especificado.

DELETE '/api/v1/localizacoes-users/remove/{id}' - Exclui um objeto LocalizacaoUsuario com o ID especificado.

sensores.php:

As rotas são agrupadas sob o prefixo "/api/v1/sensores" e incluem:

- "/lista": retorna uma lista de sensores cadastrados.
- "/lista-geral": retorna uma lista de sensores com informações adicionais, desde que sejam públicas.
- "/lista/todos": retorna uma lista de sensores com informações adicionais.
- "/lista/geral/{id}": retorna informações de um sensor específico com base no ID fornecido.

As informações adicionais incluem descrição do sensor, localização geográfica, dados meteorológicos medidos pelo sensor e informações de produção.

O código usa as classes Sensor, Dado, InformacaoSensor e AtribuicaoSensor para interagir com o banco de dados. Ele também utiliza a biblioteca JWT da Firebase para gerar tokens de autenticação.

informacao_sensor.php:

Este código PHP faz parte de um grupo de rotas para o terminal '/api/v1'. Ele contém cinco rotas para gerenciar sensores de informação.

A primeira rota é uma requisição GET para '/informacoes-sensor/lista' que retorna uma lista de sensores de informação.

A segunda rota é uma solicitação POST para '/informacoes-sensor/adiciona' que adiciona um novo sensor de informação. Ele requer um cabeçalho de autorização com um token JWT e também requer um corpo analisado com os seguintes parâmetros: propriedade, typeProduction, lowDescription, descrição, área, isActive, isPublic, estado, cidade, latitude e longitude. Também requer três parâmetros adicionais: administradores, patrocinadores e visualizadores, que são todos objetos JSON.

A terceira rota é uma requisição GET para '/informacoes-sensor/lista/{id}' que retorna um sensor de informação para um determinado ID.

A quarta rota é uma requisição PUT para '/informacoes-sensor/atualiza/{id}' que atualiza um sensor de informação para um dado ID. Requer um corpo analisado com os parâmetros a serem atualizados.

A quinta rota é uma solicitação DELETE para '/informacoes-sensor/remove/{id}' que remove um sensor de informação para um determinado ID.

dados.php:

Aqui, há uma rota /api/v1/dados/lista/{id}, que recebe uma solicitação HTTP GET e um ID como parâmetro de rota. A função associada a essa rota, então, busca todos os dados de sensores armazenados na base de dados e itera sobre eles para encontrar aquele com o ID correspondente. Uma vez encontrado, os dados são formatados em um array associativo e retornados em formato JSON como resposta para a solicitação.

Dado.php:

Este é um código PHP que define uma classe de modelo chamada "Dado" no namespace "App\Models". O modelo estende a classe "Illuminate\Database\Eloquent\Model".

A propriedade "$fillable" é uma matriz de atributos que podem ser atribuídos em massa. Esses atributos podem ser definidos usando o método "preencher" ou "criar" do modelo, que pode receber uma matriz de pares chave-valor. Todos os outros atributos que não estiverem nesta matriz não poderão ser atribuídos em massa.

Os atributos listados neste código incluem:

id: o identificador único do registro
readAt: o timestamp de quando os dados foram lidos
temperatureSoil: a temperatura do solo
temperatureAir: a temperatura do ar
luminosidade: o nível de luminosidade
pluviômetro: a quantidade de chuva
ultravioleta: o nível de radiação ultravioleta
temperatureCase: a temperatura da caixa do instrumento
rainIntensity: a intensidade da chuva
windDirection: a direção do vento
windSpeed: a velocidade do vento
gás: o nível de gás no ar
UmidadeAirRelative: a umidade relativa do ar
altitude: a altitude do local
pressão: a pressão do ar
idSensor: chave estrangeira
updated_at: o carimbo de data/hora da última atualização do registro
created_at: o timestamp de quando o registro foi criado.

Atribuicao.php:

Este é um código PHP que define uma classe AtribuicaoSensor no namespace App\Models. A classe estende a classe Illuminate\Database\Eloquent\Model, o que significa que ela herda todos os seus métodos e propriedades.

O objetivo desta classe é representar uma atribuição de um sensor a um usuário ou administrador. Ele possui os seguintes atributos, que são especificados no array $fillable:

idInfoSensor: o ID do sensor que está sendo atribuído.
isAdminSensor: um sinalizador booleano que indica se o usuário atribuído é um administrador do sensor.
idUsuario: o ID do usuário que está sendo atribuído ao sensor.

updated_at: a data e a hora em que a atribuição foi atualizada pela última vez.
created_at: a data e hora em que a atribuição foi criada.

A matriz $fillable é usada para especificar quais atributos da classe podem ser atribuídos em massa. A atribuição em massa é uma técnica do Slim que permite definir vários atributos de um modelo de uma só vez usando um array. Por padrão, todos os atributos de um modelo são protegidos, o que significa que eles não podem ser atribuídos em massa por motivos de segurança. No entanto, especificando a matriz $fillable, você pode permitir explicitamente que determinados atributos sejam atribuídos em massa.

No geral, esse código define uma classe de modelo que pode ser usada para representar atribuições de sensores a usuários do tipo administradores, patrocinadores e visualizadores.

LocalizacaoUsuario.php:

Este é um código PHP que define uma classe chamada "LocalizacaoUsuario" que estende a classe "Model" do namespace "Illuminate\Database\Eloquent".
A classe tem uma propriedade protegida "$fillable" que contém uma matriz de nomes de colunas que podem ser atribuídas em massa. Isso significa que essas colunas podem ser preenchidas com valores usando os métodos "criar" ou "atualizar" sem precisar definir explicitamente cada valor de coluna, um por um.

As colunas no array "$fillable" são:

'street': uma string representando o nome da rua do endereço do usuário
'numberU': uma string que representa o número do endereço do usuário
'city': uma string representando a cidade onde o usuário mora
'estado': uma string representando o estado onde o usuário mora
'zipcode': uma string que representa o CEP do endereço do usuário
'complemento': uma string que representa informações adicionais sobre o endereço do usuário
'neighborhood': uma string que representa o bairro onde o usuário mora
'idUser': um número inteiro que representa o ID do usuário ao qual o endereço pertence
'updated_at': um timestamp representando a última vez que o registro foi atualizado
'created_at': um timestamp representando a hora em que o registro foi criado.

Esses nomes de coluna correspondem a colunas na tabela de banco de dados que esse modelo representa.

Sensor.php:

Este é um trecho de código PHP que define uma classe de modelo chamada "Sensor" dentro do namespace "App\Models". A classe estende a classe "Illuminate\Database\Eloquent\Model".

A classe tem uma propriedade protegida chamada "$fillable", que é uma matriz de atributos que podem ser atribuídos em massa. Isso significa que você pode definir esses atributos usando os métodos "criar" ou "atualizar" fornecidos pelo Eloquent ORM do Slim.

Os atributos na matriz "$fillable" são:

readAt: Um timestamp representando quando os dados do sensor foram lidos
temperatureSoil: A temperatura do solo em Celsius
temperatureAir: A temperatura do ar em Celsius
luminosidade: O nível de luminosidade em lux
pluviômetro: A quantidade de chuva em milímetros
ultravioleta: O nível de radiação ultravioleta
temperatureCase: A temperatura da caixa do sensor em Celsius
rainIntensity: A intensidade da chuva em milímetros por hora
windDirection: A direção do vento em graus
windSpeed: A velocidade do vento em metros por segundo
gás: O nível de concentração de gás
UmidadeAirRelative: A umidade relativa do ar
altitude: A altitude do sensor em metros
pressão: A pressão atmosférica em kilopascais
updated_at: Um carimbo de data/hora representando a última vez que o registro foi atualizado
created_at: Um timestamp representando quando o registro foi criado

Usuario.php:

Esta é uma classe PHP chamada Usuario que estende a classe Model do namespace Illuminate\Database\Eloquent.

A classe representa um modelo que pode interagir com uma tabela de banco de dados para executar várias operações de banco de dados, como recuperar dados, criar, atualizar ou excluir registros de um banco de dados.

A propriedade $fillable é uma matriz de nomes de colunas que podem ser preenchidas com dados ao criar uma nova instância desse modelo ou atualizar uma existente. Qualquer outra coluna não listada na matriz $fillable não poderá ser atribuída em massa, protegendo contra modificações não intencionais.

Nesse caso, as colunas que podem ser preenchidas são user_password, isAdmin, registerConfirmed, firstName, lastName, document, phone, email, facebookProfile, gender, birthdayDate, updated_at e created_at.

InformacaoSensor.php:

Esta é uma classe PHP que estende a classe Illuminate\Database\Eloquent\Model, que é usada para interagir com uma tabela de banco de dados chamada "informacao_sensors". A classe tem um namespace App\Models e se chama InformacaoSensor.

A classe tem uma propriedade protegida $fillable, que é um array contendo os nomes das colunas da tabela "informacao_sensors" que podem ser preenchidos durante a atribuição em massa. Essas colunas são:

"lowDescription": Descrição breve do sensor em letras minúsculas.

"description": Descrição do sensor.
"isActive": Booleano que indica se o sensor está ativo ou não.
"isPublic": Booleano que indica se o sensor é público ou privado.
"area": Área de localização do sensor.

"typeProduction": Tipo de produção do sensor.
"latitude": Latitude da localização do sensor.
"longitude": Longitude da localização do sensor.
"property": Propriedade onde o sensor está instalado.
"state": Estado onde o sensor está localizado.
"city": Cidade onde o sensor está localizado.
"idSensor": ID do sensor associado a essas informações.
"updated_at": Data e hora da última atualização das informações do sensor.
"created_at": Data e hora da criação do registro de informações do sensor.

Isso significa que ao criar ou atualizar uma instância desse modelo usando os métodos create() ou update(), somente as colunas listadas em $fillable serão afetadas. Esta é uma medida de segurança para evitar alterações não autorizadas em outras colunas na tabela do banco de dados.