Struttura del progetto

Organizzazione dei files

Controllers

app/Http/Controllers
Contiene tutti i controllers che si occupano di gestire le richieste.

- **ProjectController.php:** Si occupa di fare le operazioni CRUD sulla entità *Projects* e della generazione delle *API KEYS*
- **RequestController.php:** Si occupa di salvare le richieste degli utenti alle API e della tariffazione. **(DA IMPLEMENTARE)**

app/Http/Controllers/API
Contiene i controllers delle API che si occupano di gestire le richieste e restituire i dati.

app/Http/Controllers/API/v1
Contiene i controllers della versione 1 delle API.

- **IndicatorController.php:** Contiene i metodi che si occupano di gestire e restituire i dati riguardo gli indicatori.
- **ModelController.php:** Contiene i metodi che si occupano di gestire e restituire i dati riguardo i modelli.
- **ReportController.php:** Contiene i metodi che si occupano di gestire le segnalazioni e restituire le segnalazioni degli utenti.
- **StationController.php:** Contiene i metodi che si occupano di restituire i dati riguardo le stazioni.
- **WeatherController.php:** Contiene i metodi che si occupano di restituire i dati .

app/Http/Controllers/Auth
Insieme di classi che gestiscono l'autenticazione.

Middlewares

app/Http/Middleware
Contiene tutti i *middlewares* che limitano l'accesso alle API e all'applicazione.

- **AcceptMiddleware.php:** Si occupa di controllare se nella richiesta è presente l'header `Accept: application/json`.
- **ApiKey.php:** Si occupa di controllare se nella richiesta è presente l'header `Authorization: Bearer 'Api-key'`.
- **ContentTypeMiddleware.php:** Si occupa di controllare se nella richiesta è presente l'header `Content-Type: application/json`.

Interfaces

app/Interfaces
Contiene le interfacce che definiscono il comportamento delle entità modellate

Models

app/Models
Contiene le implementazioni dei modelli e delle interfacce definite.

- **Source.php:** Classe ereditata da tutte le sorgenti dati.
- **Agroambiente.php:** Classe che estende **Source.php** e si occupa di prendere i dati dalle API esterne e restituirli formattati.
- **Project.php:** Classe che modella l'entità *Projects*
- **Report.php:** Classe che modella l'entità *Reports*
- **Request.php:** Classe che modella l'entità *Requests*
- **User.php:** Classe che modella l'entità *Users*

Traits

app/Traits
Contiene i *traits* utili per l'applicazione.

- **ResponsesJSON.php:** Si occupa di restituire una risposta *JSON* di errore già formattata.
- **UtilityMethods.php:** Insieme di metodi di validazione utili per l'applicazione.

Database

database/factories
Insieme di classi utili per l'auto-creazione di modelli, popolazione del database e per la realizzazione dei test.

database/migrations
Classi che si occupano della creazione delle tabelle del database.

database/seeders
Classi che si occupano della popolazione del database utilizzando le factories.

Routes

routes/api-v1.php
Contiene le rotte della prima versione delle API.

routes/web.php
Contiene le rotte dell'applicazione.

Tests

tests/Feature/API
Contiene i test che si occupano di verificare il corretto funzionamento delle API.

- **Authentication.php:** Si occupa di controllare che le richieste non vadano a buon fine se non è presente l'API KEY nella richiesta.
- **IndicatorTest.php:** Si occupano di testare le API degli indicatori.
- **ModelTest.php:** Si occupano di testare le API dei modelli.
- **ReportTest.php:** Si occupano di testare le API delle segnalazioni.
- **StationTest.php:** Si occupano di testare le API delle stazioni.
- **WeatherTest.php:** Si occupano di testare le API del meteo.

Dettagli implementativi

Per facilitare l'aggiunta di nuove sorgente dati all'applicazione si è deciso di aggiungere il query parameter ***source*** che può essere presente in tutte le richieste:

```php
// Controllo se c'è il parametro source (sorgente) di default è agroAmbiente
if ($request->query( key: 'source') && is_string($request->query( key: 'source'))) {
    $this->source = $request->query( key: 'source');
}

// Restituisco i dati in base alla sorgente
if ($this->source === 'agroAmbiente') {

    $agroambiente = new Agroambiente();
    return $agroambiente->stations($request, $province);

} elseif ($this->source === 'other_source') {
    //
} else {
    return $this->ResponseError( status: 400, title: 'Bad request', detail: 'Undefined source');
}
```

NB: Questo l'ho implementato solo nel metodo che ritorna tutte le stazione, se lei approva questo approccio allora lo implemento in tutti i metodi.