Metro Train Prediction App - Service Configuration Guide

Author: David Morrison

Project Repo: <https://github.com/DavMorr/wmata-app>

[Table of Contents 1](#_Toc2015021868)

[Overview 1](#_Toc371126162)

[Service Architecture 1](#_Toc615260817)

[Configuration Philosophy 1](#_Toc386799871)

[Environment Configuration 1](#_Toc1736425989)

[Laravel Backend (.env) 2](#_Toc1166403190)

[Environment Variable Reference 3](#_Toc1148197949)

[Vue Frontend (.env) 4](#_Toc1822788474)

[Service Provider Registration 4](#_Toc1568767864)

[WmataServiceProvider Implementation 4](#_Toc192630065)

[Service Registration in config/app.php 5](#_Toc1136205927)

[Laravel Configuration Files 5](#_Toc1420581590)

[WMATA Configuration (config/wmata.php) 5](#_Toc1811582307)

[CORS Configuration (config/cors.php) 6](#_Toc1682573164)

[Route Configuration (routes/api.php) 7](#_Toc1352491069)

[Frontend Configuration 7](#_Toc423934109)

[Axios Configuration (vue-app/src/api/index.js) 7](#_Toc507224395)

[Metro API Service (vue-app/src/services/metroApi.js) 8](#_Toc1291013293)

[Development Setup 9](#_Toc1431169166)

[Initial Setup Steps 9](#_Toc231408325)

[Development Workflow 10](#_Toc1932524979)

[Development Testing 11](#_Toc290580815)

[Production Deployment 11](#_Toc259397840)

[Environment Configuration 11](#_Toc1460814342)

[Production Monitoring 13](#_Toc1654458652)

[Performance Optimization 13](#_Toc1894613868)

[CLI Commands 14](#_Toc569113553)

[Metro Sync Command 14](#_Toc449001573)

[Command Output Example 14](#_Toc383769376)

[Command Implementation Details 14](#_Toc977349873)

[Automation and Scheduling 15](#_Toc1217040619)

[Troubleshooting 15](#_Toc1166531507)

[Common Issues and Solutions 15](#_Toc328136086)

[1. WMATA API Key Issues 15](#_Toc1535842140)

[2. Cache Issues 16](#_Toc1791901390)

[3. Database Connection Issues 16](#_Toc1478979281)

[4. Frontend API Connection Issues 16](#_Toc1270693147)

[5. Rate Limiting Issues 17](#_Toc2014342927)

[Diagnostic Commands 17](#_Toc980791791)

[Performance Monitoring 18](#_Toc461057856)

## Overview

### Service Architecture

The Metro Train Prediction App uses a multi-layered service architecture:

* **WmataServiceProvider** - Registers core services with dependency injection
* **WmataApiService** - Handles all WMATA API communication with caching and rate limiting
* **MetroDataService** - Manages business logic and data synchronization
* **MetroController** - Provides REST API endpoints for the frontend

### Configuration Philosophy

* **Environment-driven** - All sensitive data in .env files
* **Layered caching** - Different TTL for different data types
* **Rate limiting** - Multiple layers to prevent API abuse
* **Fallback strategies** - Graceful degradation when services are unavailable

## Environment Configuration

### Laravel Backend (.env)

bash

*# Database Configuration*

DB\_CONNECTION=mysql

DB\_HOST=mysql

DB\_PORT=3306

DB\_DATABASE=laravel

DB\_USERNAME=sail

DB\_PASSWORD=password

*# WMATA API Configuration*

WMATA\_API\_KEY=your\_wmata\_api\_key\_here

WMATA\_BASE\_URL=https://api.wmata.com

WMATA\_TIMEOUT=30

WMATA\_RETRY\_ATTEMPTS=3

*# WMATA Cache TTL Configuration (seconds)*

WMATA\_CACHE\_LINES\_TTL=86400 *# 24 hours - lines rarely change*

WMATA\_CACHE\_STATIONS\_TTL=86400 *# 24 hours - stations rarely change*

WMATA\_CACHE\_PATHS\_TTL=86400 *# 24 hours - paths rarely change*

WMATA\_CACHE\_PREDICTIONS\_TTL=15 *# 15 seconds - real-time data*

*# WMATA Rate Limiting*

WMATA\_RATE\_LIMIT=1000 *# Requests per hour to WMATA API*

*# Frontend Integration*

WMATA\_FRONTEND\_REFRESH=30 *# Auto-refresh interval (seconds)*

*# Cache Configuration (recommended for production)*

CACHE\_STORE=redis

REDIS\_HOST=127.0.0.1

REDIS\_PASSWORD=null

REDIS\_PORT=6379

*# Application Settings*

APP\_NAME="Metro Train Prediction App"

APP\_ENV=local

APP\_KEY=base64:generated\_app\_key\_here

APP\_DEBUG=true

APP\_TIMEZONE=America/New\_York

APP\_URL=http://localhost

*# CORS Configuration*

FRONTEND\_URL=http://localhost:5173

### **Environment Variable Reference**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Type** | **Default** | **Description** |
| WMATA\_API\_KEY | string | required | Your WMATA API key from developer.wmata.com |
| WMATA\_BASE\_URL | string | [https://api.wmata.com](https://api.wmata.com/) | WMATA API base URL |
| WMATA\_TIMEOUT | integer | 30 | HTTP timeout for WMATA requests (seconds) |
| WMATA\_RETRY\_ATTEMPTS | integer | 3 | Number of retries for failed requests |
| WMATA\_CACHE\_LINES\_TTL | integer | 86400 | Cache TTL for line data (seconds) |
| WMATA\_CACHE\_STATIONS\_TTL | integer | 86400 | Cache TTL for station data (seconds) |
| WMATA\_CACHE\_PATHS\_TTL | integer | 86400 | Cache TTL for path data (seconds) |
| WMATA\_CACHE\_PREDICTIONS\_TTL | integer | 15 | Cache TTL for predictions (seconds) |
| WMATA\_RATE\_LIMIT | integer | 1000 | Max requests per hour to WMATA API |
| WMATA\_FRONTEND\_REFRESH | integer | 30 | Frontend auto-refresh interval (seconds) |

### Vue Frontend (.env)

bash

*# Laravel API Configuration*

VITE\_API\_BASE\_URL=http://localhost/api

*# Application Settings*

VITE\_APP\_NAME="Metro Train Prediction App"

VITE\_PREDICTIONS\_REFRESH\_INTERVAL=30

*# Development Settings*  
VITE\_APP\_ENV=development

## Service Provider Registration

### WmataServiceProvider Implementation

php

<?php

*// app/Providers/WmataServiceProvider.php*

namespace App\Providers;

use App\Services\WmataApiService;

use App\Services\MetroDataService;

use Illuminate\Support\ServiceProvider;

class WmataServiceProvider extends ServiceProvider

{

public function register(): void

{

*// Register WmataApiService as singleton*

$this->app->singleton(WmataApiService::class, function ($app) {

$config = config('wmata');

return new WmataApiService(

apiKey: $config['api']['key'],

baseUrl: $config['api']['base\_url'],

endpoints: $config['endpoints'],

cacheConfig: $config['cache'],

maxRequestsPerHour: $config['rate\_limit']['max\_requests\_per\_hour']

);

});

*// Register MetroDataService as singleton with dependency injection*

$this->app->singleton(MetroDataService::class, function ($app) {

return new MetroDataService(

$app->make(WmataApiService::class)

);

});

}

public function boot(): void

{

*// Publish configuration file*

$this->publishes([

\_\_DIR\_\_.'/../../config/wmata.php' => config\_path('wmata.php'),

], 'wmata-config');

}

}

### Service Registration in config/app.php

php

<?php

<?php   
*// bootstrap/providers.php*

return [   
 App\Providers\AppServiceProvider::class,   
 App\Providers\RouteServiceProvider::class,   
  
 *// Metro Train Prediction App Services*   
 App\Providers\WmataServiceProvider::class, *// ← Add this line*   
];

## Laravel Configuration Files

### WMATA Configuration (config/wmata.php)

php

<?php

*// config/wmata.php*

return [

*// WMATA API Configuration*

'api' => [

'key' => env('WMATA\_API\_KEY'),

'base\_url' => env('WMATA\_BASE\_URL', '<https://api.wmata.com>'),

'timeout' => env('WMATA\_TIMEOUT', 30),

'retry\_attempts' => env('WMATA\_RETRY\_ATTEMPTS', 3),

],

*// WMATA API Endpoints*

'endpoints' => [

'lines' => '/Rail.svc/json/jLines',

'stations' => '/Rail.svc/json/jStations',

'predictions' => '/StationPrediction.svc/json/GetPrediction',

'path' => '/Rail.svc/json/jPath',

],

*// Caching Configuration - TTL (Time To Live) settings*

'cache' => [

'lines\_ttl' => env('WMATA\_CACHE\_LINES\_TTL', 86400), *// 24 hours*

'stations\_ttl' => env('WMATA\_CACHE\_STATIONS\_TTL', 86400), *// 24 hours*

'paths\_ttl' => env('WMATA\_CACHE\_PATHS\_TTL', 86400), *// 24 hours*

'predictions\_ttl' => env('WMATA\_CACHE\_PREDICTIONS\_TTL', 15), *// 15 seconds*

],

*// Rate Limiting Configuration*

'rate\_limit' => [

'max\_requests\_per\_hour' => env('WMATA\_RATE\_LIMIT', 1000),

],

*// Frontend Configuration*

'frontend' => [

'predictions\_refresh\_interval' => env('WMATA\_FRONTEND\_REFRESH', 30),

],

];

### CORS Configuration (config/cors.php)

php

<?php

*// config/cors.php*

return [

'paths' => ['api/\*'],

'allowed\_methods' => ['\*'],

'allowed\_origins' => [

'<http://localhost:5173>', *// Vue dev server*

'<http://127.0.0.1:5173>', *// Alternative localhost*

env('FRONTEND\_URL'), *// Production frontend URL*

],

'allowed\_origins\_patterns' => [],

'allowed\_headers' => ['\*'],

'exposed\_headers' => [],

'max\_age' => 0,

'supports\_credentials' => true,

];

### Route Configuration (routes/api.php)

php

<?php

*// routes/api.php*

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\Api\MetroController;

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {

return $request->user();

});

*// API health check*

Route::get('/test', function () {

return response()->json([

'message' => 'Metro Train Prediction App API is working!',

'timestamp' => now()->toISOString(),

]);

});

*// Metro API endpoints*

Route::prefix('metro')->middleware(['throttle:60,1'])->group(function () {

Route::get('lines', [MetroController::class, 'getLines']);

Route::get('stations/{lineCode}', [MetroController::class, 'getStationsForLine']);

Route::get('predictions/{stationCode}', [MetroController::class, 'getTrainPredictions']);

*// Administrative endpoint (consider adding auth in production)*

Route::post('sync', [MetroController::class, 'syncData']);

});

## Frontend Configuration

### Axios Configuration (vue-app/src/api/index.js)

javascript

*// vue-app/src/api/index.js*

import axios from 'axios';

const api = axios.create({

baseURL: import.meta.env.VITE\_API\_URL || '<http://localhost/api>',

headers: {

'Content-Type': 'application/json',

'Accept': 'application/json'

},

withCredentials: true,

timeout: 10000 *// 10 second timeout for frontend requests*

});

*// Request interceptor*

api.interceptors.request.use(

(config) => {

*// Add any auth tokens here if needed in the future*

return config;

},

(error) => {

return Promise.reject(error);

}

);

*// Response interceptor*

api.interceptors.response.use(

(response) => response,

(error) => {

*// Handle common errors*

if (error.response?.status === 429) {

console.warn('Rate limit exceeded. Please slow down requests.');

}

return Promise.reject(error);

}

);

export default api;

### Metro API Service (vue-app/src/services/metroApi.js)

javascript

*// vue-app/src/services/metroApi.js*

import api from '../api/index.js'

class MetroApiService {

async makeRequest(endpoint) {

try {

const response = await api.get(endpoint)

if (!response.data.success) {

throw new Error(response.data.error || 'API request failed')

}

return response.data.data

} catch (error) {

console.error('Metro API Error:', error)

throw error

}

}

async getLines() {

return this.makeRequest('/metro/lines')

}

async getStationsForLine(lineCode) {

return this.makeRequest(`/metro/stations/${lineCode}`)

}

async getTrainPredictions(stationCode) {

try {

const response = await api.get(`/metro/predictions/${stationCode}`)

if (!response.data.success) {

throw new Error(response.data.error || 'API request failed')

}

return response.data.data

} catch (error) {

console.error('Metro API Error:', error)

throw error

}

}

}

export const metroApi = new MetroApiService()

## Development Setup

### Initial Setup Steps

**Clone and Setup Laravel Backend**

bash

*# Navigate to Laravel app*

cd laravel-app

*# Install dependencies*

sail composer install

*# Copy environment file*  
cp .env.example .env

**Database Setup**

bash

*# Run migrations*

sail artisan migrate

*# Sync initial data*  
sail artisan metro:sync

**Setup Vue Frontend**

bash

*# Navigate to Vue app*

cd ../vue-app

*# Install dependencies*

npm install

*# Copy environment file*  
cp .env.example .env

**Start Development Servers**

bash

*# Terminal 1: Laravel (Sail)*

cd laravel-app

sail up -d

*# Terminal 2: Vue development server*

cd vue-app

npm run dev

### Development Workflow

bash

*# Daily development routine*

*# Start Laravel backend*

cd laravel-app && sail up -d

*# Start Vue frontend*

cd vue-app && npm run dev

*# Access applications*

*# Laravel API:* [*http://localhost/api*](http://localhost/api)

*# Vue App:* [*http://localhost:5173*](http://localhost:5173)

*# Sync data when needed*

sail artisan metro:sync

### Development Testing

bash

*# Test API endpoints*

curl <http://localhost/api/test>

curl <http://localhost/api/metro/lines>

curl <http://localhost/api/metro/stations/RD>

curl <http://localhost/api/metro/predictions/A01>

*# Check Laravel logs*

sail artisan log:clear

sail logs -f

*# Run Laravel tests (if configured)*  
sail artisan test

## Production Deployment

### Environment Configuration

**Production .env Settings**

bash

*# Application*

APP\_ENV=production

APP\_DEBUG=false

APP\_URL=https://your-domain.com

*# Database (use production database)*

DB\_CONNECTION=mysql

DB\_HOST=your-production-metro-db-host

DB\_PORT=3306

DB\_DATABASE=your\_production\_metro\_db

DB\_USERNAME=your\_production\_metro\_db\_user

DB\_PASSWORD=your\_production\_metro\_db\_secure\_password

*# WMATA API*

WMATA\_API\_KEY=your\_production\_metro\_api\_key

*# Cache (use Redis for production)*

CACHE\_STORE=redis

REDIS\_HOST=your-production-metro-redis-host

REDIS\_PASSWORD=your\_production\_metro\_redis\_password

REDIS\_PORT=6379

*# CORS*

FRONTEND\_URL=https://your-production-metro-frontend-domain.com

**Optimize Laravel for Production**

bash

*# Clear and cache configurations*

php artisan config:cache

php artisan route:cache

php artisan view:cache

*# Optimize autoloader*  
composer install --optimize-autoloader --no-dev  
  
*# Generate application key (if not set)*  
php artisan key:generate

**Database Setup**  
bash  
*# Run migrations*  
php artisan migrate –force  
  
*# Sync initial Metro data*  
php artisan metro:sync

**Vue Production Build**

bash

*# Build for production*

npm run build

*# Serve static files through web server (nginx/apache)*

### Production Monitoring

bash

*# Schedule regular data sync (add to crontab)*

*# Daily sync at 3 AM*

0 3 \* \* \* cd /path/to/app && php artisan metro:sync

*# Monitor logs*

tail -f storage/logs/laravel.log

*# Check service status*  
php artisan metro:sync --validate

### Performance Optimization

**Redis Configuration**  
bash  
*# Redis memory optimization*  
maxmemory 256mb  
maxmemory-policy allkeys-lru

**Database Optimization**  
sql  
*-- Optimize MySQL for read-heavy workload*

SET innodb\_buffer\_pool\_size = 1073741824; *-- 1GB*

SET query\_cache\_size = 134217728; *-- 128MB*

SET query\_cache\_type = 1;

**Web Server Configuration**  
nginx  
*# Nginx configuration for Vue SPA*

location / {

try\_files $uri $uri/ /index.html;

}

*# API proxy*

location /api {

proxy\_pass <http://laravel-backend>;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

## CLI Commands

### Metro Sync Command

The primary administrative command for the Metro Train Prediction App:

bash

*# Basic sync*

sail artisan metro:sync

*# With cache validation*

sail artisan metro:sync --validate

### Command Output Example

Starting Metro data synchronization...

+---------------------+-------+

| Type | Count |

+---------------------+-------+

| Lines synced | 6 |

| Stations synced | 95 |

| Path entries synced | 95 |

+---------------------+-------+

Metro data sync completed successfully!

Stations will now display in proper sequence order

### Command Implementation Details

php

*// app/Console/Commands/SyncMetroData.php*

protected $signature = 'metro:sync

{--validate : Validate cache integrity first}';

protected $description = 'Sync Metro data from WMATA API including station paths';

public function handle(): int

{

if ($this->option('validate')) {

$this->info('Checking cache integrity...');

if ($this->metroService->validateCacheIntegrity()) {

$this->info('Cache is valid');

return Command::SUCCESS;

} else {

$this->warn('Cache validation failed, proceeding with sync...');

}

}

*// ... sync logic ...*

}

### Automation and Scheduling

php

*// app/Console/Kernel.php*

protected function schedule(Schedule $schedule): void

{

*// Sync Metro data daily at 3 AM*

$schedule->command('metro:sync')

->dailyAt('03:00')

->appendOutputTo(storage\_path('logs/metro-sync.log'));

*// Validate cache integrity every hour*

$schedule->command('metro:sync --validate')

->hourly()

->appendOutputTo(storage\_path('logs/metro-validation.log'));

}

## Troubleshooting

### Common Issues and Solutions

#### **1. WMATA API Key Issues**

**Problem**: API request failed with status: 401

**Solutions**:

bash

*# Check API key in .env*

grep WMATA\_API\_KEY .env

*# Verify API key at* [*https://developer.wmata.com*](https://developer.wmata.com)

*# Ensure key has proper permissions*

*# Test API key directly*

curl -H "api\_key: YOUR\_KEY" <https://api.wmata.com/Rail.svc/json/jLines>

#### **2. Cache Issues**

**Problem**: Stale or corrupted cache data

**Solutions**:

bash

*# Clear all cache*

sail artisan cache:clear

*# Clear specific Metro cache*

sail artisan tinker

Cache::forget('wmata.lines');

Cache::forget('metro.lines.frontend');

*# Rebuild cache*  
sail artisan metro:sync

#### **3. Database Connection Issues**

**Problem**: Database connection errors during sync

**Solutions**:

bash

*# Check database connection*

sail artisan tinker

DB::connection()->getPdo();

*# Check migration status*

sail artisan migrate:status

*# Reset database if needed*

sail artisan migrate:fresh

sail artisan metro:sync

#### **4. Frontend API Connection Issues**

**Problem**: Vue app cannot reach Laravel API

**Solutions**:

bash

*# Check environment variables*

cat vue-app/.env

*# Ensure VITE\_API\_BASE\_URL=http://localhost/api*

*# Test API directly*

curl <http://localhost/api/test>

*# Check CORS configuration*

*# Verify vue-app URL is in config/cors.php allowed\_origins*

#### **5. Rate Limiting Issues**

**Problem**: Too many requests errors

**Solutions**:

bash

*# Check current rate limits*

sail artisan tinker

Cache::get('wmata\_api\_rate\_limit');

*# Increase limits in .env*

WMATA\_RATE\_LIMIT=2000

*# Clear rate limit cache*

Cache::forget('wmata\_api\_rate\_limit');

### Diagnostic Commands

bash

*# Check service registration*

sail artisan route:list --path=api

*# Validate configuration*

sail artisan config:show wmata

*# Check database tables*

sail artisan tinker

App\Models\Line::count();

App\Models\Station::count();

App\Models\StationPath::count();

*# Monitor logs in real-time*

sail logs -f

tail -f storage/logs/laravel.log

### Performance Monitoring

bash

*# Check cache hit rates*

sail artisan tinker

Cache::get('wmata.lines') ? 'HIT' : 'MISS';

*# Monitor database queries*

*# Add to .env: DB\_LOG\_QUERIES=true*

*# Check storage/logs/laravel.log for query logs*

*# Check API response times*

curl -w "[@curl-format.txt](mailto:@curl-format.txt)" -o /dev/null -s <http://localhost/api/metro/lines>

This configuration guide provides comprehensive setup and deployment instructions for the Metro Train Prediction App's service architecture.