Metro Train Prediction App - Service Configuration Guide

Author: David Morrison

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

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	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

```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
<?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
<?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

```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
# Vue App: 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
```

```
# 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" -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.