**Integration Guide: Connecting Your App with Mevo API**

Mevo provides a RESTful API that allows applications to interact with their mobility service. The API enables functionalities such as user authentication, vehicle search, booking management, and trip tracking.

1. **Understand Mevo API Documentation**

* Read through the official API documentation: [Mevo API Docs](https://developer.mevo.co.nz/)
* Identify available endpoints and authentication methods.
* Check API rate limits to avoid exceeding requests.

1. **Authentication & Authorization**

Obtain API Access: Register app with Mevo and receive Client ID and Client Secret for OAuth 2.0. Contact Mevo’s API team at [api@mevo.co.nz](mailto:api@mevo.co.nz) to request access.

* Receive client\_id and client\_secret.
* Use Basic Authentication by encoding {client\_id:client\_secret} in **Base64**.

To make requests, include the *Authorization* header and *credentials* mentioned before(Img 1)

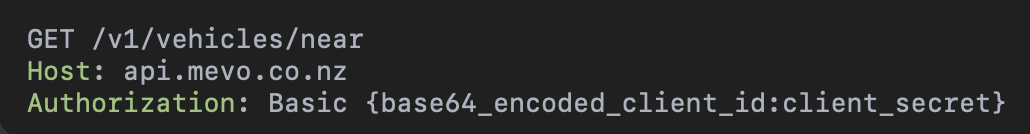


Image 1: GET request with credentials example

1. **Implement API Requests on server side** 
   1. **V1 Endpoints**

Mevo’s APIs are versioned via the URI. All endpoints of the V1 API are nested under:<https://api.mevo.co.nz/v1/>. All needed functionality divided by two main flows: cities and vehicles so we need to develop two main functionality flows API: /v1/cities/… and /v1/vehicles/…

**Cities**: Use GET /v1/cities/:cityName/vehicles to retrieve all available vehicles in a city (Img 2).

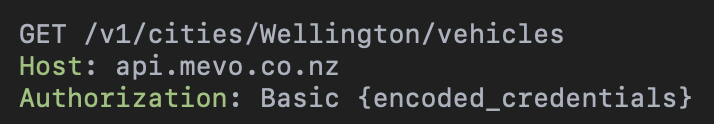


Image 2: GET request for available vehicles in Wellington

Store response data (Img 3) in our cache system.

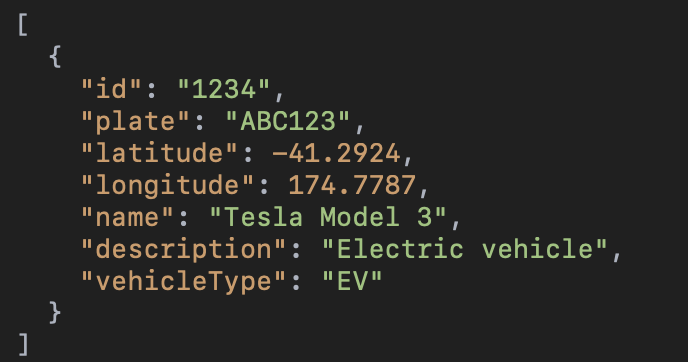


Image 3: available vehicles in Wellington response

**“Find Nearby Vehicles” Feature:**

GET /v1/vehicles/near?latitude={lat}&longitude={long}

For this request we need to know *latitude* and *longitude* params to form a request to Mevo API. Response statuses and behavior:

###### **Status: 200 OK:** The response body will contain a JSON list representing the available vehicles near the supplied latitude/longitude: **id** is the unique identifier of the vehicle. This can be used in a further details request or in a universal link. **name** is the display name of the vehicle. Typically the make and model. **latitude** represents the vehicle’s current latitude. **longitude** represents the vehicle’s current longitude(Img 4).

###### 

###### 

Image 4: Response body for status 200

**Status: 204 No Content:** Returned if there are no available vehicles near the supplied latitude/longitude.

###### **Status: 400 Bad Request:** Returned if latitude/longitude is missing or if they are not numeric.

###### **Status: 401 Unauthorized:** Returned if the encoded credentials are incorrect. Ensure that your application client ID and client secret match those provided by Mevo. Check that you have base64 encoded your credentials. You might need to remove newline characters from the encoded string.

**Detailed Vehicle Information:**

GET /v1/vehicles/:vehicleId

Retrieves detailed information about a specific available vehicle. Vehicle ids can be obtained by querying the Nearby Vehicles endpoint first. Request contains only one param and it is *vehicleId.* Response statuses and behavior:

**Status: 200 OK:** The response body will contain a JSON payload with the vehicle’s details.

****

Image 5: Response for status 200 for GET /v1/vehicles/:vehicleId

**Status: 401 Unauthorized:** Returned if the encoded credentials are incorrect. Ensure that your application client ID and client secret match those provided by Mevo. Check that you have base64 encoded your credentials. You might need to remove newline characters from the encoded string.

**Status: 404 Not Found:** Returned if no vehicle exists with the supplied id, or the vehicle is unavailable for any reason.

* 1. **Public Endpoints**

A selection of endpoints are available without authentication—these never contain any sensitive information and do not return live data. They are useful for illustrative or marketing purposes; for example, displaying an animated map of Mevo vehicles within a city. The public API endpoints are also a perfect resource for testing the API and familiarising yourself with its usage.

**Vehicles**

Vehicle data returned by this endpoint is obfuscated. For example, the number of results is not the same as the actual number of available vehicles, and the position of the pins does not reflect the live position of a vehicle(Img 6). Request Url: GET /public/vehicles/:cityName



Image 6: Response for public endpoint “vehicles”

**Parking**

The parking available for a vehicle in each city depends on the operating model of that vehicle within the city. The two primary models are Mevo Flex (dockless or free-floating) and Mevo Loop (back to base). Request URL: GET /public/parking/:cityName



Image 7: Response for public endpoint “parking”

**4**.**Data Storage**

I think in what case we do not need to create a database for integration. We just need to use in-memory cache to store some information about city and vehicles just to increase speed of response to our clients and also to decrease rate-limit just by this simple decision. In datagram for integration you can see a schema of cache usage.

**Additional Features & Optimization**

Optimize API Calls: Use caching for vehicle locations. Implement rate limiting to avoid excessive requests.

Error Handling & Logging: Implement retry mechanisms for network failures. Log errors for debugging (e.g., ELK Stack or Datadog).

**Conclusion**

By following these steps, app will be able to integrate seamlessly with Mevo API, allowing users to find, book, and manage car rentals efficiently. All applied here endpoints are presented in datagram with name [Untitled Diagram.drawio](https://github.com/AndrewThief/mevo-api-integration-diagram).