Coding Challenge

Please read the fictional product description and instructions below (read to the end before starting). The challenge time may vary greatly from developer to developer depending on past experience, but it is estimated that it should take **less than 1 day *(working hours)*** to complete.

If any questions arise during the challenge, please contact **Alex Santos** (alex@blissgrowth.com) for clarification.

The project should use the following technologies:

- **Language:** TypeScript
- **Framework:** Nest.js
- **Database:** Any SQL database *(sqlite3, postgres, mysql…)*;

Any choice of technology required for the project, not described above, may be chosen by the developer.

Once the project is ready, please share by pushing it to a git repository and sending a link to alex@blissgrowth.com and ella@blissgrowth.com (if you want to keep the project private, please create a private repository in **Github** and share it with the username **Colex**).

API Connection Manager

FictionalAPI wants to build an API connection repository/manager. A service that allows developers to submit credentials for an integration, and then use the service to call the underlying integration. The vision is that the service would handle credentials life-cycle management, data transformations, end-user integration management and so on. Owning all parts of the API/Integrations management.

The goal is to create a connection repository for integrations (a simplified version of a unified API service). It should allow developers to easily submit credentials for an integration, and then use the service to query the underlying integration.

The project should not require any kind of authentication or account creation to be used.

Scope

For this challenge, we would like to build a subset of the features as a POC:

- Endpoint for submitting a new connection (hereinafter **New Conn API**);
    - An API that accepts the provider + credentials (e.g. access token + refresh token), stores the data and returns a reference;
    - Note: different APIs have different set of credentials for authentication (some use access token, other use basic authentication, some require an instance url, and so on);
- Endpoint for calling the underlying integration (hereinafter **Call API**);
    - For a connection added before, an API can be called giving the connection reference, a path (not the whole URL), and a method for the underlying integration and the service will make the call (building the correct URL and authentication mechanism);
    - It can be one endpoint or more (however you see necessary), but it needs to allow reaching the underlying service;
    - There's no authentication for this service, so the endpoint should be somehow protected so no one can do the call with a connection they shouldn't have access to;
        - (This security mechanism may be used by other endpoints around the Connection resource, so it should be re-usable);

You do not need to support a lot of providers for the challenge - **at least one** provider should be supported. The project should support being extended with new providers though. You may use whatever provider(s) you wish - preferably something you can quickly access the API (e.g. **Stripe, GitHub, Pipedrive, Salesforce, Hubspot**).

API Examples

Here is an example of what the API could look like. This is **only for clarification**, do not use this as the actual endpoint.

New Conn API

Request

```
POST /add-a-connection
Body
{
 "provider": "google",
 "accessToken": "1234567890",
 "refreshToken": "0987654321"
}
```

Response

```
{
 "SomeReferenceKey": "XYZ"
}
```

Call API

Request

```
POST /call-connection
Body
{
 "ConnectioRefFromBefore": "XYZ",
 "Path": "/application.list",
 "Payload": {
   "cursor": "1"
 }
}
```

Response

The Response should be whatever response comes from the underlying service.

Extra Scope

The following can be seen as extra scope features, you may pick and implement depending on time. They are ordered by priority.

- **Data Mapping**
  - Add an API that allows creating a custom mapping (e.g. “first_name” should be extracted from field “FirstName”);

```
POST /create-mapping
{
 "mapping": {
   "first_name": "firstName",
   "last_name": "lastName",
   "email": "emails[0].email"
 }
}
Result:
{ "id": "XYZ" }
```

  - When calling the **Call API** allow specifying a reference to a mapping created before;

        - **If a mapping reference is specified**, then return the result with the mapped field (otherwise just return the raw result);
- **Token lifecycle management**
    - For integrations that require refreshing the token, periodically (e.g. every 15 minutes) refresh the token and update the database with the new token;

Final Remarks

- The most important point of this project is the code architecture. Code organization and best practices;
- It's understandable that there may not be enough time to create the "perfect project" - feel free to make your choices according to the first point if you need to shortcut;
- Any infrastructure detail is not required;