**Part 2 – Theory**

To transform the above example into a real application, we might create a backend microservice API that exposes a RESTful API endpoint that accepts a Pokemon name and returns its evolution chain in JSON format. This API endpoint could be used by the frontend application to display the evolution chain for a certain Pokemon.

* **API Design:** We must create the API endpoints and associated input/output formats. For example, we could have an endpoint **/evolution-chain/:pokemonName** that provides the evolution chain of a certain Pokémon. The output format should be the same as in Part 1, namely a JSON object containing the Pokémon name and its variations.
* **Data Caching:** Because the evolution chains of Pokémon are unlikely to change frequently, we may consider caching the replies from the Poke API for a set period (e.g., 1 hour) to reduce the frequency of API calls.
* **Testing:** Testing is an important element of developing any programme. To test the backend API, we would need to develop unit tests to ensure that the API endpoint is functioning properly. We can write and run these tests with a testing framework like Jest. To test the front-end application, we would need to develop end-to-end tests that imitate user interactions and validate that the programme works as planned. To accomplish this, we can utilise tools such as Cypress or Selenium.
* **Deployment and Releases:** We need to decide how to deploy the backend API and when to release new versions. One alternative is to use a cloud provider such as AWS or Heroku, along with a CI/CD pipeline, to automatically deploy new versions as they become available.
* **API Versioning:** In order to avoid breaking changes, we may need to version our API. For example, we can prefix the API endpoints with a version number (e.g., **/v1/evolution-chain/:pokemonName**) and keep various versions of the API for backward compatibility.
* **Non-functional Requirements:** We must address non-functional requirements such as scalability, dependability, and security. To manage a high volume of API calls, for example, we may need to deploy load balancers and auto-scaling groups.