# Santex Back-end Developer Hiring Test

The goal is to create a **project** that exposes an API built with GraphQL, with a mutation and some queries. In case you don't know GraphQL you can do this test with a rest API, but we appreciate you trying it with GraphQL anyway (researching on how to do it).

**What should this API do?**

We'll be hitting http://www.football-data.org/ API *(you can see the documentation in the site, use the API v4)* to populate the data locally and then expose it.

**Import League:**

There should be a mutation (or endpoint in case you're working with a rest API) to import a league, named **importLeague**, that takes a **leagueCode** as input.

The import league implementation must get data using the given {leagueCode}, by making requests to the http://www.football-data.org/ API, and **import** the data into a **DB**. Any SQL or NoSQL DB can be used, as long as there are **clear instructions** on how to run the project locally as well as an explanation for the decision in the README.

The data we're importing is:

- Competition ("name", "code", "areaName")
- Team ("name", "tla", "shortName", "areaName", "address")
- Player ("name", "position", "dateOfBirth", "nationality")

Note: When making the implementation, if there is no data (no players) inside team ***squads***, then, instead of importing *players*, you should import only the ***coach***:

- Coach("name", "dateOfBirth", "nationality") (import that data even if the values are null)

Feel free to add to this data structure any other field that you might need.

**Information to retrieve:**

Additionally, expose the following queries (or endpoints), that should rely exclusively on the data saved inside the DB (it **must not** access the API *football-data.org*):

- **players**: takes **leagueCode** as a parameter and returns the players that belong to all teams participating in the given league. If the given leagueCode is not present in the DB, it should respond with an error message. Add an optional input to the query to filter players also by team name. **Important:** If there is no players' data available (see *Note* above), then make the same implementation for this using ***coaches*** instead.

- **team:** takes a name and returns the corresponding team. Additionally, if requested in the query, it should resolve the players for that team (or *coaches*, if players are not available at the moment of implementation).

**What we expect:**

- **Once you have finished the project, you must upload all the relevant files inside a ZIP compressed file. It must include all the sources, plus the files related to project configuration and/or dependency management.**
- Please notice that even though this is a paid API, you can get a free token and perform your testing with some specific competitions (PD, CL, PL)
- It's important that the code handles in some way the limit frequency to the requests performed with a free-token
- You are allowed to use any library related to the language in which you are implementing the project.
- All the mentioned DB entities must keep their proper relationships (the players with which team they belong to; the teams in which leagues participate).
- It might happen that when a given leagueCode is being imported, the league has participant teams that are already imported (because each team might belong to one or more leagues). For these cases, it **must** add the relationship between the league and the team(s) (and omit the process of the preexistent teams and their players).
- Please explain your train of thought and your decision making (for libraries/frameworks used) in the README or another doc inside the project**.**

**Nice to have:**

- It is a plus that the project automatically generates any necessary schemas when it runs the first time.
- Usage of Docker or any other containerization is also a plus.
- Think about this as a real product and add anything you feel could add value (like other queries or mutations/endpoints). **SHOW ALL YOUR SKILLS, SURPRISE US!**