Task Backend Developer Python

In this challenge, you are asked to develop REST API for a metallics optimization service.

The metallics optimizer calculates the cheapest charge mix for an electric arc furnace (EAF) in a steel plant. The charge materials are different scrap types and virgin material. In summary those materials are called *commodities*. The optimization algorithm uses the *chemical composition of commodities* as input values to guarantee that the chemical composition of the tapped melt is within a specific range. The melt is tapped after the melting process in the EAF is finished. One complete melting process as well as tapping the melt into a ladle is called a *hea*t.

The solution should be published in a public repo on github. Please write a brief readme file to describe what we need to install and how to run your project. It also should include

Requirements

- The service should be implemented in Python 3.5+.
- The data should be stored in an SQL-database. You are free to choose any database management system you like.
- Please include the database initialization into the project. Use a script or data migration to fill the DB with some initial data.
- A method call should not block the service (they need to work asynchronously). We use FastAPI, but you may use any async web framework you like.
- All API methods should require user authorisation. You are free to choose any autorisation schema you want. User registration is not required, just add a user to the database when you initialize it.

Models

There are two main models in the service: a chemical element and a commodity.

A chemical element has the following properties:

- id - id of the element
- name - name of the element

Properties of a commodity are:

- id - id of the commodity
- name - name of the commodity
- inventory - current amount of the commodity on stock (in tons)
- price - current price of the commodity ($/ton)
- chemical_composition - chemical elements and their percentage in the commodity.

Please notice that different commodities may have the same element in their chemical composition.
Total concentration of the elements in a commodity may be below 100%. In this case, it should include a concentration of an “Unknown” element that brings the total concentration to 100%. For example, if a commodity has 15% of Cu, 25% of Al and 50% of Fe, it should also have 10% of the “Unknown” element.

API Endpoints

The solution should provide several API methods. We use JSON as the data interchange format.

1. Get all chemical elements.

This endpoint should return a list of all chemical elements with their names and ids.
Example output:

```
[
   {
      "id": 6,
      "name": "C"
   },
   {
      "id": 7,
      "name": "N"
   },
   {
      "id": 8,
      "name": "O"
   },
   {
      "id": 13,
      "name": "Al"
   }
]
```

2. Get a commodity by id

This endpoint returns one commodity by id. A commodity chemical composition should contain id and name of the elements. Example:

```
{
   "id": 42,
   "name": "Plate & Structural"
   "price": 1234.50,
   "inventory": 200.5,
   "chemical_composition": [
      {
         "element": {"id": 13, name: "Al"},
         "percentage": 25
```

```
      },
      {
         "element": {"id": 26, name: "Fe"},
         "percentage": 50
      },
      {
         "element": {"id": 29, name: "Cu"},
         "percentage": 10
      },
      {
         "element": {"id": 9999, name: "Unknown"},
         "percentage": 15
      }
   ]
}
```

3. Update commodity by id

We also need an endpoint to update commodity information. This endpoint should update commodity name, price and inventory by its id. If a field is not provided, it should not be updated.
For example, if we want to update the name and price of the commodity with id 1, input JSON may look like this:

```
{
   "id": 1,
   "name": "new commodity name",
   "price": 1000
}
```

4. Add and remove chemical concentration

We need two endpoints to manage chemical concentrations of a commodity. The first one takes a commodity id, element id and a percentage and adds it to the commodity chemical composition. For example:

```
{
   "commodity_id": 1,
   "element_id": 26,
   "percentage": 10.5
}
```

The second one removes an element from the chemical composition by id:

```
{
   "commodity_id": 1,
   "element_id": 26
}
```

Please notice that you need to adjust the “Unknown” element percentage every time when the composition is changed.