Bradoo - Python Developer

This document contains a problem we use to evaluate candidate skills.

You are expected to deliver a fully functional solution, given the specification described below. Your performance will be evaluated not just by presenting a functional solution, your code design and choices, deployment, documentation and tests will count in your final score.

Hints:

- Read the specification before working. If you have any questions feel free to ask
- Check the recommendations and reference material at the end of this specification
- Over Engineering isn't a mark of efficiency - keep your project simple, and deliver the requested specification

How to submit your exam

1. Create a public or private git repository
2. Commit your code and any instructions needed to run the project into the repository
3. Deploy your project on a hosting service (we recommend Heroku);
4. Send a email to renato.sabo@bradootech.com with:
    - Your name
    - The address of the git repository
    - The location and required credentials where you deployed the project

Specification

You should implement an application for registration of vendor catalogues.

This application must both provide an HTTP REST API and an HTTP Frontend to attend the requirements.

1. Front end Application

Deliver a simple HTTP front end application that will allow listing, editing, creating and removal of vendors and products. You can implement the list/edit in a same page or in two or more pages, as you see fit.

The vendor record need to have these fields:

- Id (self generated, surrogate key)

- Name (required)
- CNPJ (required, this value is unique, ie, you can only have one record given a single CNPJ)
- City

List section

- Listing of vendor records, should allow searching of the records by any of the fields or a combination of them all.
- Paging is expected
- Button to allow the removal of a single or group of vendor records
- Button to create a new vendor
- Button to edit a vendor

Edit section

- The edit/create section must include the required fields
- There must be a button to remove the record, if already existent
- There must be a button to create/save the record
- The create/save button must validate if the required fields are filled
- Upon successful create/save of record the user must be redirected to the vendor listing
- CNPJ value needs to be a valid number according to Brazilian rules

Products

From within the vendors record, you should provide a list and record creation method and input form for the products of the vendor.

The product record need to have these fields:
- Id (self generated, surrogate key)
- Name (required)
- Code (required)
- Price

Tip: don't forget this is a related record with the vendor as parent, ie, a product requires a vendor and can only have one vendor.

The products records should only be changed/create/deleted upon the vendor create/save button

The creation or changing of products also needs to check if required fields are filled.

2. Expose vendor data in an endpoint API

For the second part of this exam, you need to deliver a fully functional REST API for the vendor model.

This endpoint needs to return a paginated list with the vendor's data. Optionally the vendors can be searched by name, CNPJ, ID or a combination of these fields.

CRUD (Create, Read, Update, Delete) of vendors

You need to implement these actions in your API:

- Create a vendor
- Read vendor's data
- Update vendor's data
- Delete vendor's data

Example:

To create a vendor you need to send this payload (in json format) below:

```
{
 "name": // Name of the vendor,
 "CNPJ": // CNPJ number,
 "city": // City of the vendor,
 "products": [ {
                  "Name": //product name,
                  "Code" : //product code,
                  "Price" : //product price,
                  }

                ]
}
```

Additional details

- Use the appropriated HTTP verbs for the CRUD action, example, *POST*, *GET*.
- When no data is found, you should return HTTP Code 204 - *No content*.

Project Requirements

- Application must be written in Python using Django or Flask
- You must use an ORM to interface with the database, like own Django ORM
- Use Python >= 3.5
- Use PEP-8 for code style - Python Coding Style
- Every text or code must be in English
- Write the project documentation containing:
    - Description;
    - Installing (setup) and testing instructions;
    - If you provide a docker solution for setup, ensure it works without docker too.
    - Brief description of the work environment used to run this project (Computer/operating system, text editor/IDE, libraries, etc).
- Provide API documentation (in English)
- Variables, code and strings must be all in English.
- You can use whatever frontend framework you want like Angular, even no one.
- You must use a relational database, like Postgres (preferable) or MySQL.
- Your application should be stateless
- Deploy and deliver your code (eg. Heroku)

Bonus

- Provide a fully functional docker image of your application. We prefer container deployments whenever possible
- Provide postman collection for the API part of this exam

Recommendations

- Practice the 12 Factor-App concepts;
- Use SOLID design principles;
- Use git best practices, with clear messages (the commit text can be in PT-BR)
- Don't over engineer
- Not sure about REST? Check this sample guide