HelloFresh OpsTech Coding exercises

Duration : 2-3h

In the OpsTech team, 90% of our work consists to develop web applications to help other teams and support them. A web app has multiple advantages as : no installation required, updatable whenever we want, accessible from everywhere, easier to code …

For our apps, we always use Python and a Javascript front-end. The frameworks that we would like to test you on are **Flask**, **Peewee** and **VueJS**. All their documentation can be found with Google, as well as some answers that you could have on them.

Even if you don't know those frameworks, they are easy to get started with and Google is your friend. We want to evaluate your capacity to learn quickly something that is **logic** and **predictable**.

Context

HelloFresh is a food delivery company. We create recipes every week and deliver them to our customers everywhere in Australia. A **recipe** is a model but when we plan a recipe (in a **menu**, a.k.a the recipes available for a week), we create a **Mealkit** (bundle of ingredients that we deliver to our customers).

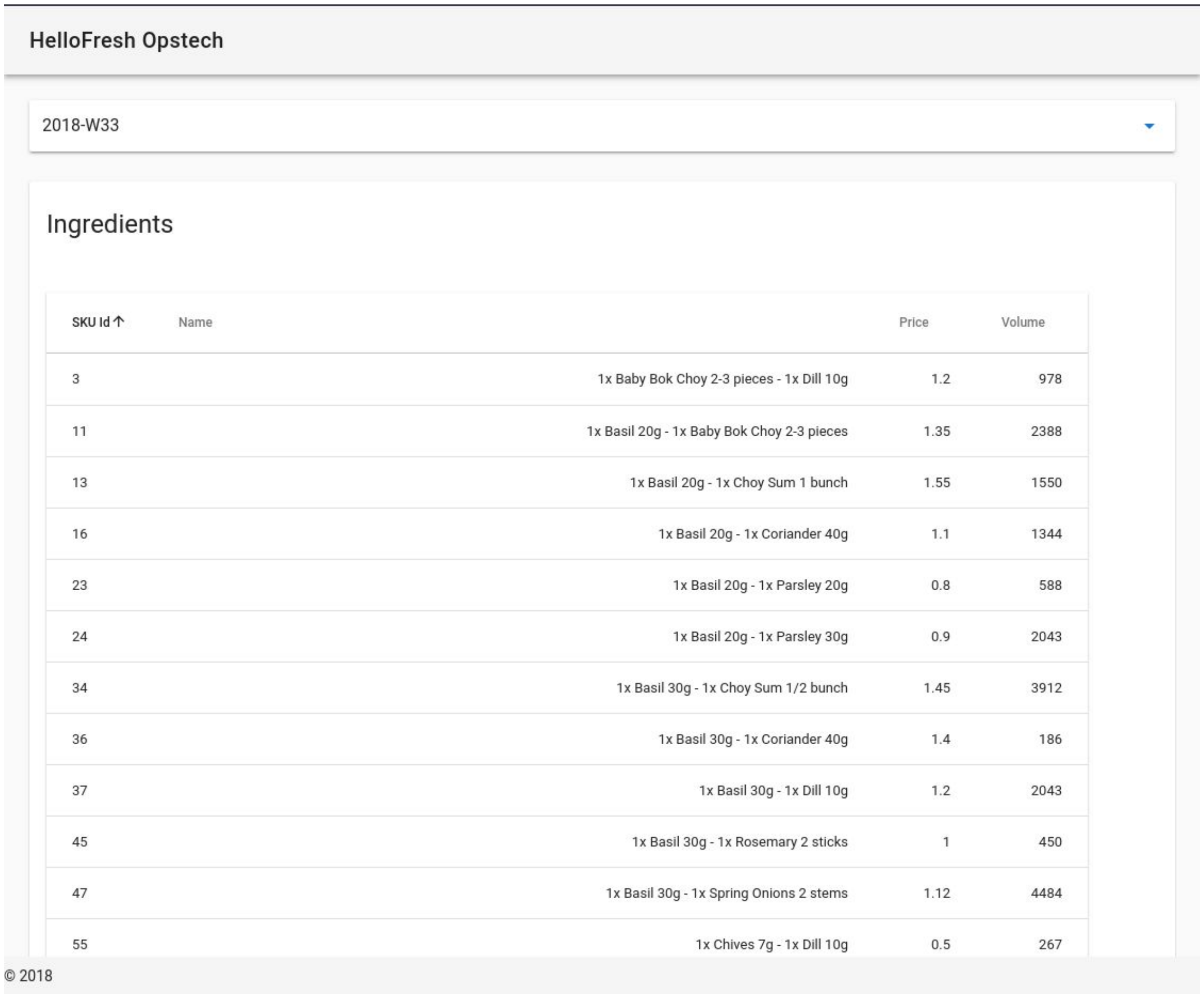
For a week, you have multiple menu, but we are only interested in the menu that have a **version** equal to 2. For the menu corresponding, we have 20 mealkits, and each of them has ingredients. Each **mealkit ingredient** is linked to a real ingredient that we have in our stock, with its price and name.

For a week, we also have the orders from our customers, called the **forecast**. Each order is function of a slot (the recipe number in the menu), that can be linked to the mealkit slot attribute in a menu.

Goal

Your goal is to complete the Python back-end and the VueJS front-end to display all the ingredients needed for a given week, with their price and their volume.

Example : We have 3 mealkits (slots 1, 2 and 3) with a *Carrot* inside. 2 of them have 1 carrot, and the last one has 2 carrots. The forecast predicts for slot 1 300 orders, and for the slot 2 and 3 100 orders each. This gives **600 carrots for this week**.



Ingredients

SKU Id ↑	Name	Price	Volume
3	1x Baby Bok Choy 2-3 pieces - 1x Dill 10g	1.2	978
11	1x Basil 20g - 1x Baby Bok Choy 2-3 pieces	1.35	2388
13	1x Basil 20g - 1x Choy Sum 1 bunch	1.55	1550
16	1x Basil 20g - 1x Coriander 40g	1.1	1344
23	1x Basil 20g - 1x Parsley 20g	0.8	588
24	1x Basil 20g - 1x Parsley 30g	0.9	2043
34	1x Basil 30g - 1x Choy Sum 1/2 bunch	1.45	3912
36	1x Basil 30g - 1x Coriander 40g	1.4	186
37	1x Basil 30g - 1x Dill 10g	1.2	2043
45	1x Basil 30g - 1x Rosemary 2 sticks	1	450
47	1x Basil 30g - 1x Spring Onions 2 stems	1.12	4484
55	1x Chives 7g - 1x Dill 10g	0.5	267

© 2018

Ressources

You have in back/models.py all the **Peewee** (a Python ORM) models needed for the **Recipes database**. You also have the connections to the **Forecast DB and Recipes DB**. All of this is managed by Peewee, so you need to search for the documentation and see how you can query the databases with it. The requirements.txt contains all the packages that you may need.

> *Caution* : you may need to set up Flask CORS with *CORS(app)* to communicate with your browser.

We expect an SQL query (with execute_sql) for the Forecast DB, and the usage of the models for the Recipes DB.

You need to set up a Web server with Python, and we clearly encourage **Flask**. But you can do whatever you want, if the result is consistent and working.

For the front-end request, we advice the usage of **Axios**, a web library for browsers. You need to complete the *getIngredients* function to get from the backend the ingredients and then display them in a card in the HTML. We encourage the usage of **Vuetify** to add components like cards or data tables. The headers have been provided and the weeks as well. Each time you change a week, it will call the *getIngredients* function.

Obviously, the endpoint in the back-end should take the week in parameters, and should return a JSON with the list of ingredients. But as I said, I don't care if the result works.

Clues

- My main.py working is 35 lines long. If yours is more like 100 long, there may be a trick that you missed.
- Don't hesitate to call the back references of the ForeignKey (if you don't, we will ask the question ahah).
- A cross-origin error ? Checkout Flask CORS. We will ask you the question what is CORS so check it out anyway.
- Stuck ? We are sleeping sorry :(

Good luck !