Exercise: Building the Presentation Layer with Python and Streamlit

Campusbier Database

In this exercise, we create a simple graphical user interface (GUI) for our example application. The GUI accesses the data layer which we built in the previous exercise. Note that in our simple application, there is no need to implement an application layer yet. The application layer would normally act as a mediator between the presentation layer (GUI) and the data layer. We will build the application layer in the next exercise, when we actually need it.

> Note that you must have downloaded and opened the *campusbier.db* database before you start this exercise. Also, you must have completed the previous exercise *Building the Data Layer with Python*.

Task 1: Create an empty Streamlit app

We do not want to become experts in building graphical user interfaces, but nevertheless we want to employ a simple GUI for our example application. The Streamlit framework is a perfect solution for us, as is it built with a prototyping mindset. In this first part of the exercise, we create an empty Streamlit application that serves as our template for the rest of this exercise.

Start by installing Streamlit using the Python package manager `pip`:

```
pip install streamlit
```

Make sure activate your virtual Python environment first, if you use one. If not, the streamlit package will be installed globally.

Next, create a new Python script named `frontend.py` and copy the following code

```
import streamlit as st
st.title("Home")
st.write("This the simple GUI for our information management system.")
```

That's all we need at this point. Now start the Streamlit app by typing the following command in your terminal:

```
streamlit run frontend.py
```

This will start a new webserver on your local machine and open the start page of our GUI in your default browser. Voilà, we created a graphical user interface with three lines of code .

Task 2: Search and display orders

In this part, we use the GUI to let the user search for an order by its ID and display the result. For that, we reuse our function from the data layer we previously built.

1. Extend your `frontend.py` script to include a text field and a button. Prompt the user to enter a valid order ID in the text field and then press the button.
2. When the user pressed the button, the application should get the input from the text field and pass it to the data layer's function to fetch details for the order.
3. Save the result from the database query on a variable. Examine the variable by printing it with `st.write()`. Find a way to display the data in a user-friendly format.

Task 3: Show orders for a date range

In this part of the exercise, we add the functionality to show order for a given date range. We can leverage our data layer for easy access to the data.

1. To separate this feature of your application from the previous one, add a new divider to your page and add a heading "List orders for date range". Add a heading for the previous feature, too.
2. Add two date pickers to let the user pick a start and end date. Add a button and assign a useful label to it.
3. When the user pressed the button, the application should fetch all orders for the selected date range using the respective function from our data layer. Before submitting the start- and end date to the function, see if you add a validation to ensure the start date is smaller than the end date.
4. Finally, display the result in a suitable format. See if you can get a Streamlit table to work.

Task 4: Show unique customers for a date range

1. As before, add a divider and a speaking heading for this new feature.
2. Add two more date pickers to let the user pick the start and end date for the customer report. Add a button to execute the query.
3. After the user pressed the button, execute the function from the data layer and pass the start and end dates. Add a validation as before - can you reuse it?
4. Display the list of customers along with their number of orders and turnover from the date range. See if you can additionally find a suitable visual representation.

Good job! You created your first simple GUI! In the next part of this exercise, we are going to add some more functionality for which we need the application layer and some external API.