Docs

Status	Completed

Getting Started

Libraries used

```
import numpy as np
import pandas as pd
import requests
import json
import time
import ast
import streamlit as st
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from IPython.display import display
from pyairtable import Api, Base, Table
from PIL import Image
from wordcloud import WordCloud, STOPWORDS
from io import BytesIO
```

Main class: client

This is made to create instances of a client, with the attributes required to make for him:

- Calendar menu.
- Shopping list.
- Nutrition lists.

__ init __()

```
def __init__(self,name,surname,kcal,menudays,diet, exclude, address, client_id):
```

- Name: *String* with the name of our client
- Surname: *String* with the surname of our client
- Kcal: *String* with the desired calories for each day of the menu
- Menudays: *List of strings* with the days that the client wants to receive menu
- Diet: *String* with the dietary restrictions of our client. In case it has none, the string should be “ “
- Exclude: *String* with the ingredients the client wants to exclude from the diet, separated with *commas.* As with diet, if none, enter a *space*.
- Address: *String* with the address of the client.
- Client_id: *String* with our client ID. *Optional*. This is used in case that we want to upload our client to Airtable. In that case, the ID should be obtained based on the current maximum ID in the clients database.

Example:

```
myclient = client("Daniel",
                  "Tummler",
                  "2000",
                  ["Monday","Tuesday","Wednesday","Thursday","Friday"],
                  "vegan",
                  "peanuts, raisins",
                  "Somewhere in Madrid",
                  "159")
```

display_info()

Prints our client info (without the ID), and returns a *list* with each attribute.

```
myclient.display_info()
```

edit_name()

Replaces the client name with the one provided, and prints the change you've made.

```
myclient.edit_name("Dani")
```

edit_surname()

Replaces the client surname with the one provided, and prints the change you've made.

```
myclient.edit_surname("Rosales")
```

edit_kcal()

Replaces the client calories with the ones provided, and prints the change you've made.

```
myclient.edit_kcal("2500")
```

edit_menudays()

Replaces the client menu days of the week with the ones provided, and prints the change you've made.

```
myclient.edit_kcal(["Friday","Saturday","Sunday"])
```

edit_diet()

Replaces the client dietary restriction with the one provided, and prints the change you've made.

```
myclient.edit_diet("vegetarian")
```

edit_exclude()

Replaces the client excluded ingredients with the one-s provided, and prints the change you've made.

```
myclient.edit_exclude("chicken, meat")
```

edit_address()

Replaces the client address with the one provided, and prints the change you've made.

```
myclient.edit_address("Somewhere in A Coruña")
```

create_calendar_menu()

Makes a *request* to Spoonacular API and creates a *dataframe* with "Breakfast, Lunch,Dinner" as index, and the *menudays* of this client as columns.

When it’s done, it prints “Calendar menu created!”.

```
myclient.create_calendar_menu()
```

return_calendar_menu()

Returns the *dataframe* created in *create_calendar_menu().*

```
myclient.return_calendar_menu()
```

display_calendar_menu()

Displays the *dataframe* created in *create_calendar_menu().*

```
myclient.display_calendar_menu()
```

return_recipes_ids()

Returns a *list of strings* with the ids of each recipe in the calendar menu.

```
myclient.return_recipes_ids()
```

create_shopping_list()

Makes a request to Spoonacular API and creates a *list* with the information of each recipe in the menu.

After, it loops over the *list* to create:

- *List* with the ingredient names of every recipe.
- *List* with the ingredient amounts of every recipe.
- *List* with the ingredient units of every recipe.

In the end, creates a *dataframe* with *lists* as columns, and prints “Shopping list created!”.

```
myclient.create_shopping_list()
```

return_shopping_list()

Returns the *dataframe* created in *create_shopping_list().*

```
myclient.return_shopping_list()
```

display_shopping_list()

Displays the *dataframe* created in *create_shopping_list().*

```
myclient.display_shopping_list()
```

return_recipe_info()

Returns the *list* with all the recipes info.

```
myclient.return_recipe_info()
```

return_recipes_instructions()

Returns a *dictionary* with recipes as *keys* and a *list of strings* with each step for the recipe.

```
myclient.return_recipes_instructions()
```

display_recipes_instructions()

Prints each step of every recipe in the created menu.

```
myclient.display_recipes_instructions()
```

create_nutrition_lists()

Creates the following *lists* and *dataframes*:

- *Lists*:
  - Nutrient names.
  - Nutrient amounts.
  - Nutrient units.
  - Nutrient PDN (Percentage of Daily Needs).
  - Properties names.
  - Properties amounts.
  - Properties units.
- *Dataframe*:
  - Nutrients.
  - Properties.

When it finishes, prints: “Nutrition lists created!”.

```
myclient.create_nutrition_lists()
```

return_nutrients_list()

Returns nutrients *dataframe* created in *create_nutrition_lists()*

```
myclient.return_nutrients_list()
```

display_nutrients_list()

Displays nutrients *dataframe* created in *create_nutrition_lists()*

```
myclient.display_nutrients_list()
```

return_properties_list()

Returns properties *dataframe* created in *create_nutrition_lists()*

```
myclient.return_properties_list()
```

display_properties_list()

Displays properties *dataframe* created in *create_nutrition_lists()*

```
myclient.display_properties_list()
```

Airtable functions

These are made to interact with the Airtable clients database, using pyairtable.

Each one of them requires, at list, three attributes relative to the database we're using, so you will need to know them for inicialization.

```
api_key = "keyKwTRludsHj7GFw"
base_id = "appM62yZdhosjF9WU"
table_name = "Clients"
```

If you want to use another Airtable database to use this app, you need to have the following structure:

Name	Surname	Calories	Days of the menu	Diet	Excluded ingredients	Address	ID
STR	STR	STR	STR	STR	STR	STR	STR

load_list_of_clients_to_airtable()

Loads a list of clients to Airtable.

```
clients = [["Daniel",
            "Tummler",
            "2000",
            ["Monday","Tuesday","Wednesday","Thursday","Friday"],
            "vegan",
            "peanuts, raisins",
            "Somewhere in Madrid",
            "159"],["Jacob",
                    "Bamio",
                    "2500",
                    ["Monday","Tuesday","Wednesday","Thursday","Friday"],
                    " ",
                    "raisins",
                    "Somewhere in Pontevedra",
                    "160"]]

load_list_of_clients_to_airtable(clients,api_key, base_id, table_name):
```

load_client_to_airtable()

Loads one client to Airtable.

```
myclient = ["Daniel",
            "Tummler",
            "2000",
            ["Monday","Tuesday","Wednesday","Thursday","Friday"],
            "vegan",
            "peanuts, raisins",
            "Somewhere in Madrid",
            "159"]

load_client_to_airtable(myclient,api_key, base_id, table_name)
```

extract_all_clients_from_airtable()

Returns a *list* with all the clients in the Airtable database.

```
airtable_clients = extract_all_clients_from_airtable(api_key, base_id, table_name)
```

extract_client_from_airtable()

Returns one client from Airtable that matches the client_id passed to the function.

```
myclient = extract_client_from_airtable("159",api_key, base_id, table_name)
```

return_max_id()

Returns the higher ID in the entire Airtable database, in order to give the next to a new client.

```
max_id = return_max_id(api_key, base_id, table_name)
```

delete_all_records()

Deletes all the records in the Airtable database. Just used in case it's needed for any test.

```
delete_all_records(api_key, base_id, table_name)
```

Analyzed plots functions

These are made show our client interesting data allocated in the app.

return_pie_diets()

Returns a *figure* with a comparative between the dietary restrictions of our clients, made with *matplotlib*.

```
fig = return_pie_diets(api_key,base_id,table_name)
```

return_wordcloud_graph()

Returns a *figure* with a *wordcloud* that shows the main ingredients in our client menu. It requires a shopping list to extract the ingredients, that can be obtained with *return_shopping_list().*

```
fig = return_wordcloud_graph(myclient.return_shopping_list())
```

return_sunburst_graph()

Returns a *figure* with a *sunburst* from *plotly* that shows the mean PDN of the nutrients in our client menu. It requires a nutrients list to extract the values, that can be obtained with *return_nutrients_list().*

```
fig = return_sunburst_graph(myclient.return_nutrients_list())
```

return_macronutrients_graph()

Returns a *figure* with a *seaborn bar plot* that shows the mean PDN of the micronutrients in our client menu. It requires a nutrients list to extract the values, that can be obtained with *return_nutrients_list().*

```
fig = return_macronutrients_graph(myclient.return_nutrients_list())
```

return_micronutrients_graph()

Returns a *figure* with a *seaborn bar plot* that shows the mean PDN of the macronutrients in our client menu. It requires a nutrients list to extract the values, that can be obtained with *return_nutrients_list().*

```
fig = return_micronutrients_graph(myclient.return_nutrients_list())
```