CODE SOURCE

Le script qui instancie et exécute l'application dash est dans le fichier **app.py**.

```
app = Dash(__name__)
if __name__ == "__main__":
       app.run_server(debug=True)
```

Dans le script on y trouve deux gros types de composants :
→HTML : chaque balise HTML a son composants

```
html.Div([
        html.Div(
            html.P("Bienvenue sur l'application AI Match X Easy Date",
style={'textAlign': 'center','color': 'pink','fontSize': 30})),
)])
```

→Dash Core Component : composants interactifs, générés avec JavaScript, HTML et CSS via la bibliothèque React.js.

```
dbc.Tabs(
        [    dbc.Tab(label="Accueil", tab_id="Accueil"),
            dbc.Tab(label="Statistique", tab_id="Statistique"),
            dbc.Tab(label="Modélisation", tab_id="Modelisation"),
            dbc.Tab(label="Prédiction", tab_id="Prediction")
        ],
        id="tabPanel",
        active_tab="Accueil",
    )])
```

Mise en page de l'application:
Barre de menu : Le TABPANEL propose quatre onglets cliquables qui amènent à quatre pages différentes.

```
TABPANEL = dbc.Container()
```

Corpus: Le contenu est divisé en quatre sections qui s'affichent en fonction de l'onglet sélectionné.

```
html.Div([...], id="id-onglet")
```

Les deux onglets "**accueil**" et "**modélisation**" proposent un contenu statique de cette forme.

```
html.Div([
   html.P(),
   html.Img(),
])
```

L'onglet "**statistique**" affiche des graphes grâce aux fonctions dcc.Graph()
Le premier graphique est interactif : il se charge en fonction des données du dataframe, que l'on spécifie dans les filtres (composant dcc.Dropdown) et les autres graphs sont créés appelés depuis fig.py.
Ce fichier propose une classe qui prend en entrée des dataframes. Plusieurs fonctions sont écrites, chaque fonction crée une figure à l'aide des librairies plotly : express et graph_objects et matplotlib.
Les traitements des dataframes passés en paramètres de la classe Fig sont faits dans une classe PrepDatas du fichier prepdatas.py.

L'onglet "**prédiction**" propose une API pour prédire les matchs grâce au modèle de prédiction model/model.pickle.dat sur le GitHub

```
html.Div([
            dcc.Dropdown(id="xInput", options=[{"label":name,"value":name} for
name in df.columns], value="age", style={"width":"100%", "padding":"5px"}),
            dcc.Dropdown(id="yInput", options=[{"label":name,"value":name} for
name in df.columns], value="income", style={"width":"100%", "padding":"5px"}),
            dcc.Dropdown(id="colorInput", options=[{"label":name,"value":name}
for name in df.columns], value="gender", style={"width":"100%",
"padding":"5px"})
      ], id="DivFilter", style={"display":"flex"}),
      dcc.Graph(id="GraphStat_1", style={"width":"100%", "margin":"5px"}),
```

Fonctions @callback:

Les fonctions callback sont des fonctions appelées en réaction à un événement ou une interaction de l'utilisateur. Par exemple en pressant un bouton.
output : id de l'élément qui change avec le callback
input : id et val de l'élément qui appel le callback
state : id des éléments utilisés dans le callback

```
render_tab_content()
#affiche le le contenue ("corps") en fonction de l'onglet sélectionné

update_graph()
#met à jour le graphique en fonction des filtres sélectionnés

predFromFile()
#fait la prédiction (.predict()) a partir du fichier renseigné, créer un fichier
avec les prédictions et renvoi ce fichier

predFromForm()
#fait la prédiction à partir des champs remplies dans le formulaire
```