0. Initialization

In [1]:
```
%matplotlib inline
```

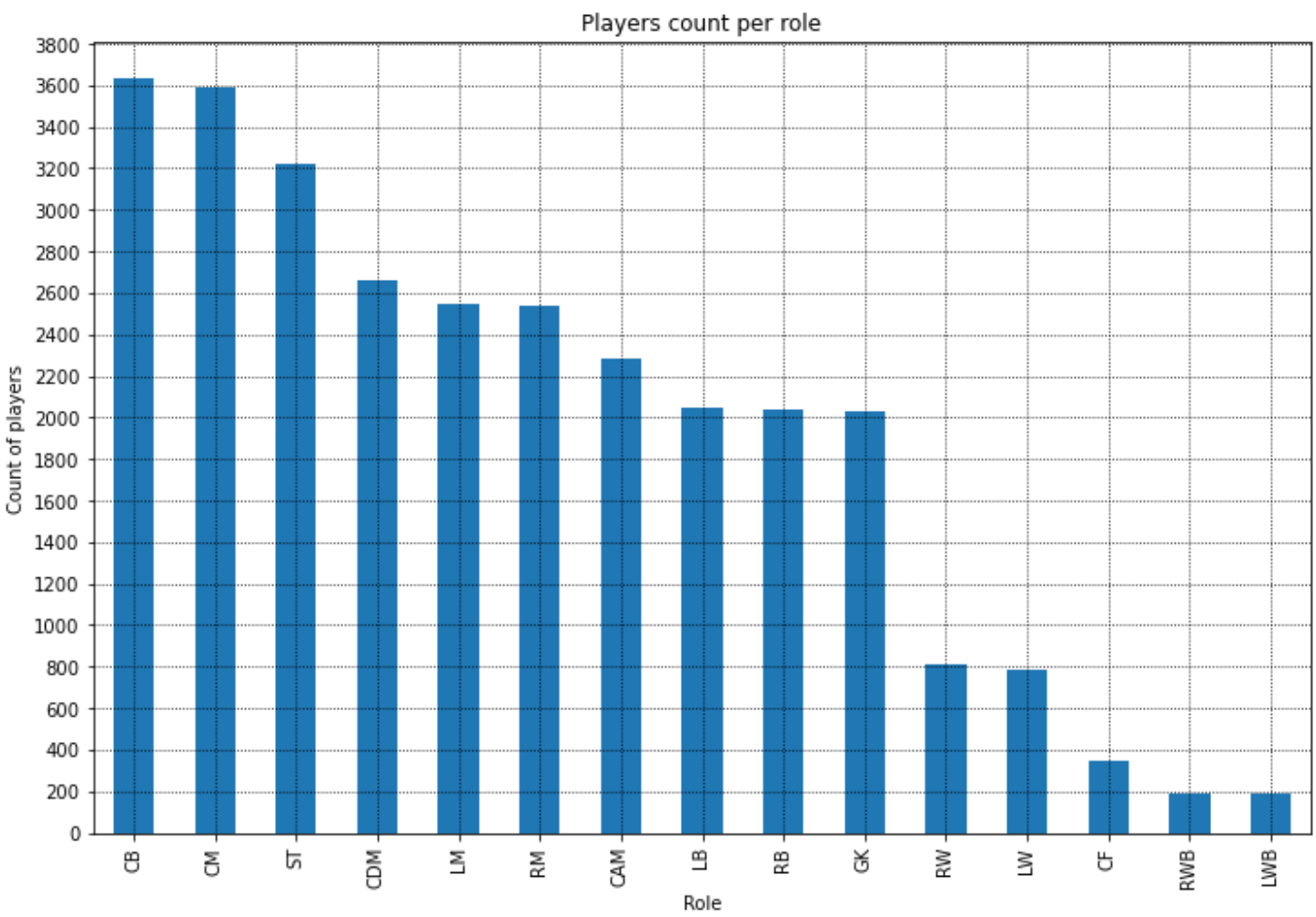
In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.offline as offline
import plotly.io as pio


pio.templates.default = "plotly_white"
offline.init_notebook_mode()

pd.set_option("display.max_columns", None)
plt.rcParams['figure.figsize'] = [12.0, 8.0]
plt.rcParams.update({
    "axes.grid" : True,
    "grid.color": "k",
    'grid.linestyle': ":",
})

complete_ds = pd.read_csv('datasets/CompleteDataset.csv', low_memory=False)
```

In [3]:
```
def convert_money_to_float(value):
    '''
    Converting a money string €100K, €50M into valid float value
    '''
    value = value.lstrip('€')
    magnitude = {'M': 1000000, 'K': 1000}
    num, index = value[:-1] or 0, value[-1]
    if index in magnitude:
        return float(num)*float(magnitude[index])
    return float(num)
```

In [4]:
```
ds = (complete_ds.assign(Roles = complete_ds["Preferred Positions"].str.strip(' ').str.spl
ds['Value'] = ds['Value'].map(convert_money_to_float)
ds['Wage'] = ds['Wage'].map(convert_money_to_float)
```

1. What roles are more popular than others?

In [5]:
```
ax = ds['Roles'].value_counts().plot(kind='bar')
ax.set_xlabel('Role')
ax.set_ylabel('Count of players')
ax.set_yticks(range(0, 4000, 200))
_ = ax.set_title("Players count per role")
```



Summary

Top-3 roles in contemporary football are:

- **CB** - central defender
- **CM** - central midfielder
- **ST** - striker

There are more than 3000 players who can do these roles

The rarest roles are:

- **LWB**, **RWB** - left and right wing-back
- **CF** - centre forward

2. How many players can change playing style?

Many footballers can play different roles. Let's build a table which will show how many players can combine playing styles. How many strkers can play as defenders? How many goalkeepers can do midfielder roles?

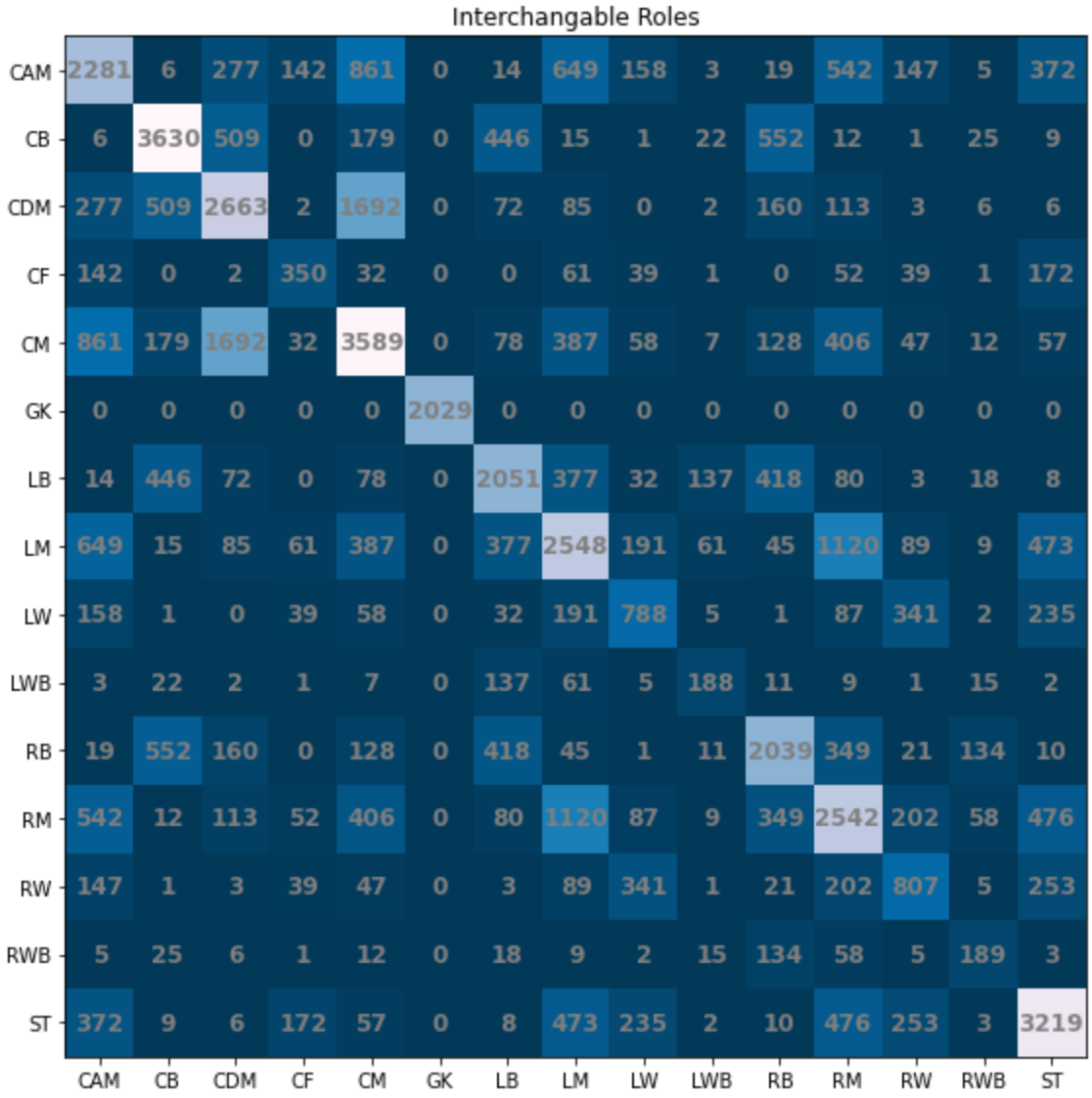
In [6]:

```
def get_interchangable_roles_count(first_role, second_role):
    players = ds.loc[ds["Roles"] == first_role]
    return ds.loc[ (ds["ID"].isin(players["ID"]))  & (ds["Roles"] == second_role) ]["ID"].

roles = sorted(ds["Roles"].unique())
interchangable_roles = list()
for first_role in roles:
    subresult = list()
    for second_role in roles:
```

```
        subresult.append(get_interchangable_roles_count(first_role, second_role))
    interchangable_roles.append(subresult)
```

In [7]:

```
fig, ax = plt.subplots()
im = ax.imshow(interchangable_roles, cmap='PuBu_r')


ax.grid(False)
ax.set_xticks(range(0,len(roles)))
ax.set_yticks(range(0,len(roles)))
ax.set_xticklabels(roles)
ax.set_yticklabels(roles)

for i in range(len(roles)):
    for j in range(len(roles)):
        text = ax.text(j, i, interchangable_roles[i][j],
                       ha="center", va="center", color="gray", fontsize="large", weight="h

ax.set_title("Interchangable Roles")
fig.tight_layout()
plt.show()
```



Summary

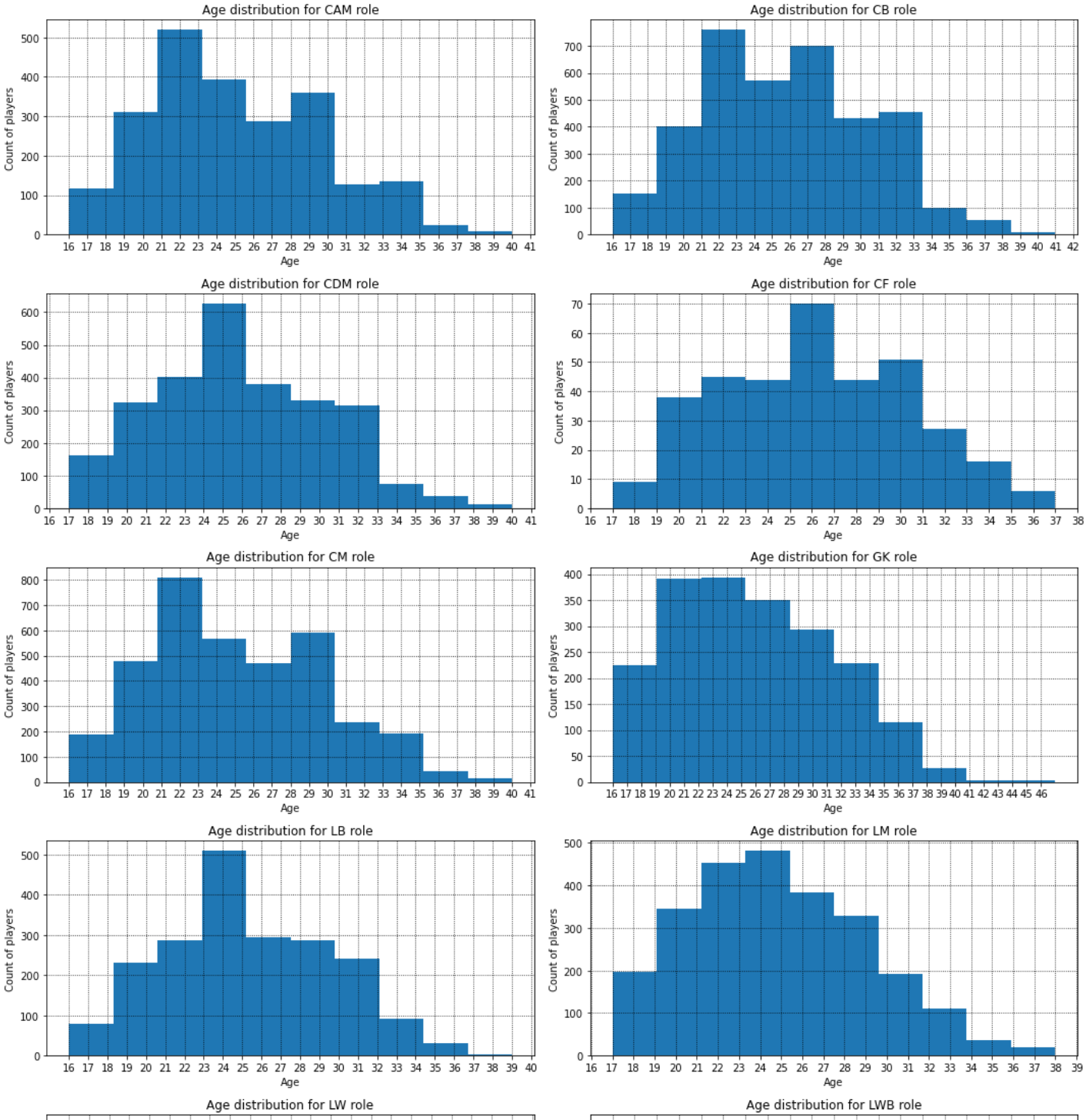
- There are no **Goalkeepers** (**GK**) who play as outfield players.
- and vice versa, outfield players don't do goalkeeper role.

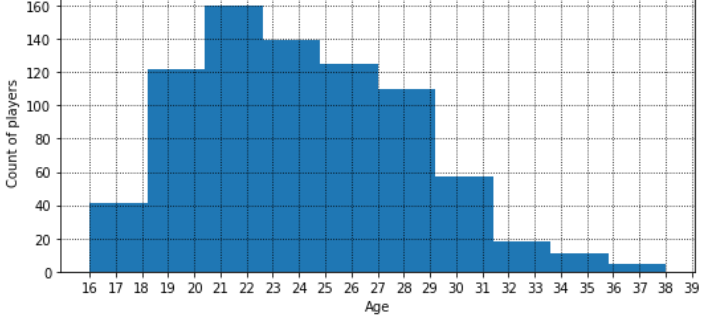
- Outfield players can change their roles in many cases. For instance many **Strikers** (**ST**) can do **Midfilder** roles (**LM**, **RM**, **CAM**)

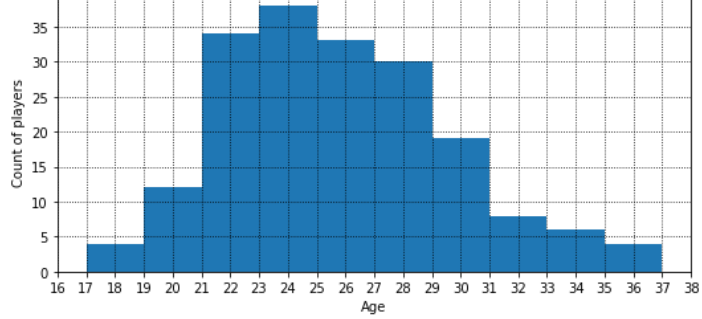
3. Players age per role

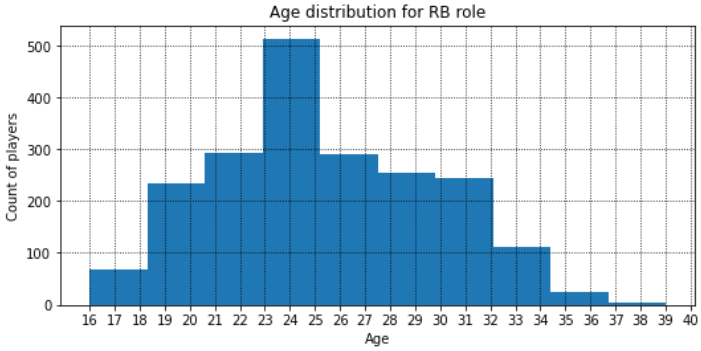
Let's build age distribution for every playing position.

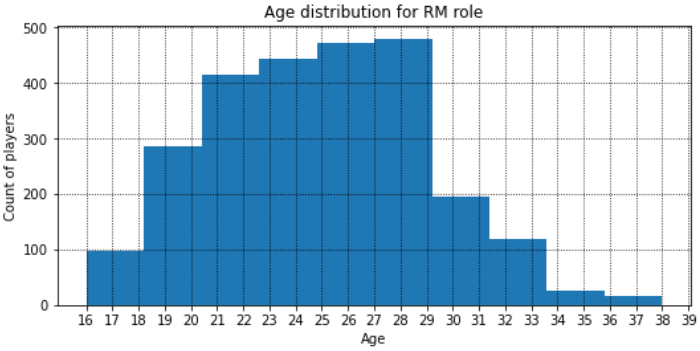
In [8]:
```
fig, axs = plt.subplots(nrows=8, ncols=2, figsize=(15,30))
axs[-1, -1].axis('off')
axes = axs.ravel()
for _ax, (label, df) in zip(axes, ds.groupby('Roles')):
    _ax.set_xticks(range(16,47))
    df['Age'].plot.hist(ax=_ax, label=label)
    _ax.title.set_text(f"Age distribution for {label} role")
    _ax.set_ylabel('Count of players')
    _ax.set_xlabel('Age')
plt.tight_layout()
```

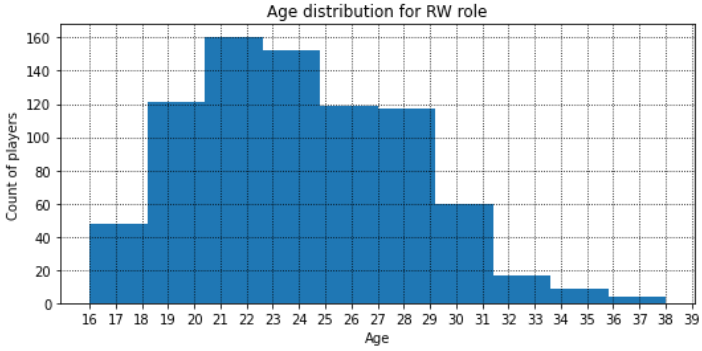


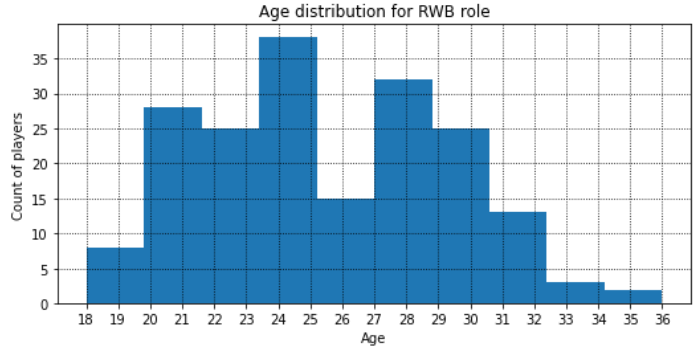


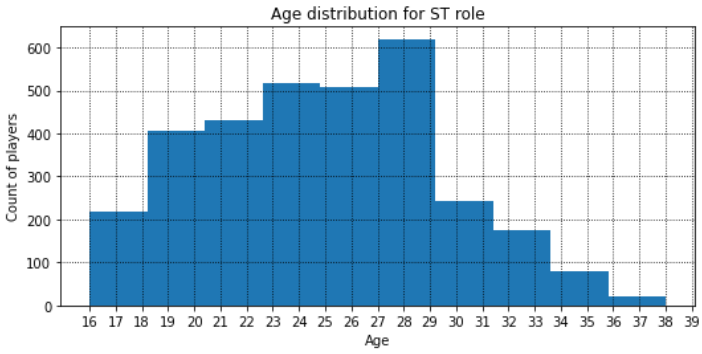












Summary

- The biggest group of footballers in all roles are 20 - 28 years old.
- **Forwarders** (**RW**, **RB**) a bit younger than other footballers.
- some **goalkeepers** (**GK**) can be good even after their forty.

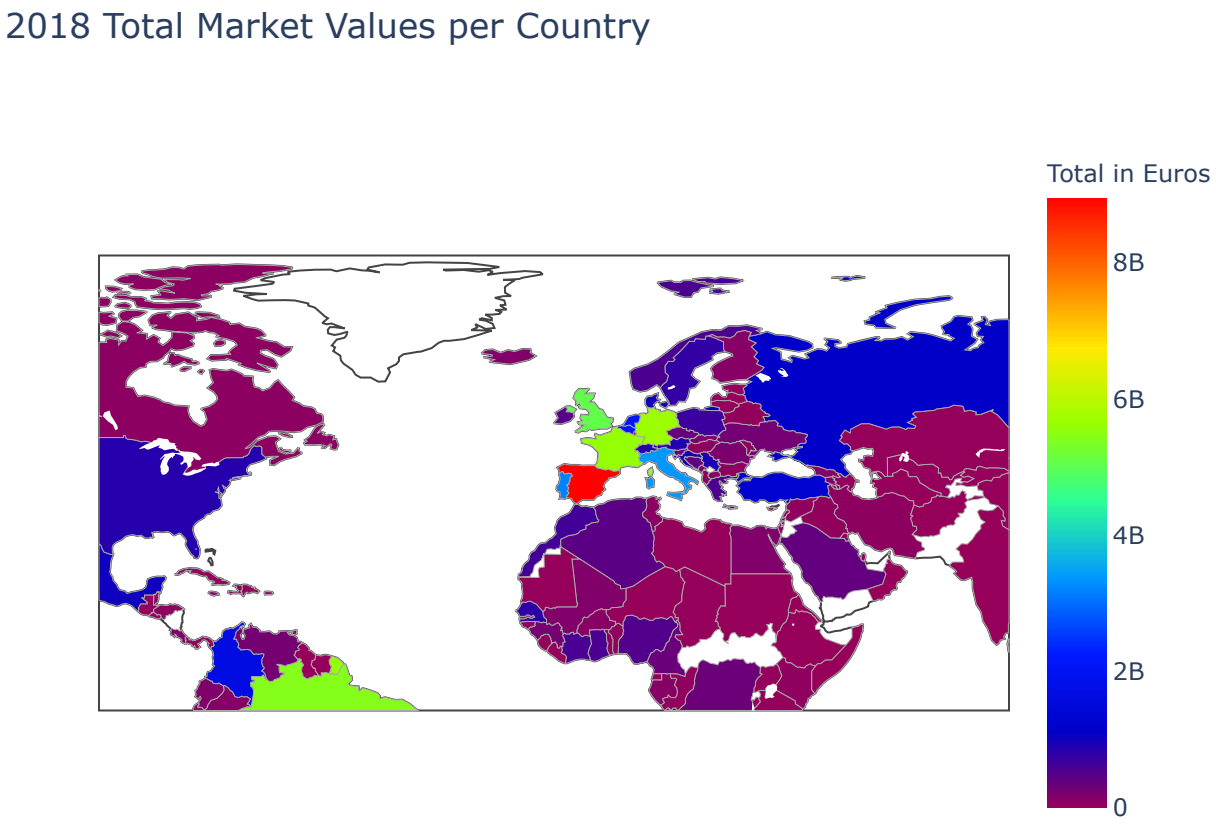
4. Total Market Value of Every Country

In [9]:

```
df_value = ds.copy()
# United Kingdom is a complicated union of several countries: Wales, Scotland, England...
# For some reason these countries have separate football teams.
# But we will summarize them all to the one 'United Kingdom'
df_value['Nationality'] = df_value['Nationality'].replace(['England'],'United Kingdom')
df_value['Nationality'] = df_value['Nationality'].replace(['Wales'],'United Kingdom')
df_value['Nationality'] = df_value['Nationality'].replace(['Scotland'],'United Kingdom')
df_value['Nationality'] = df_value['Nationality'].replace(['Northern Ireland'],'United Kir
df_value = df_value.groupby('Nationality').sum().reset_index().sort_values('Value', ascend
```

In [10]:

```
fig = go.Figure(data=go.Choropleth(
    locations=df_value['Nationality'],
    z = df_value['Value'],
    locationmode = 'country names',
    colorbar_title = "Total in Euros",
    colorscale = 'Rainbow',
    autocolorscale=False,
    reversescale=False,
    marker_line_color='darkgray',
    marker_line_width=0.5,
))

fig.update_geos(
    center=dict(lon=-10, lat=40),
    resolution = 110,
    projection=dict(scale=2),
)

fig.update_layout(
    title_text = '2018 Total Market Values per Country',
)

offline.iplot(fig)
```



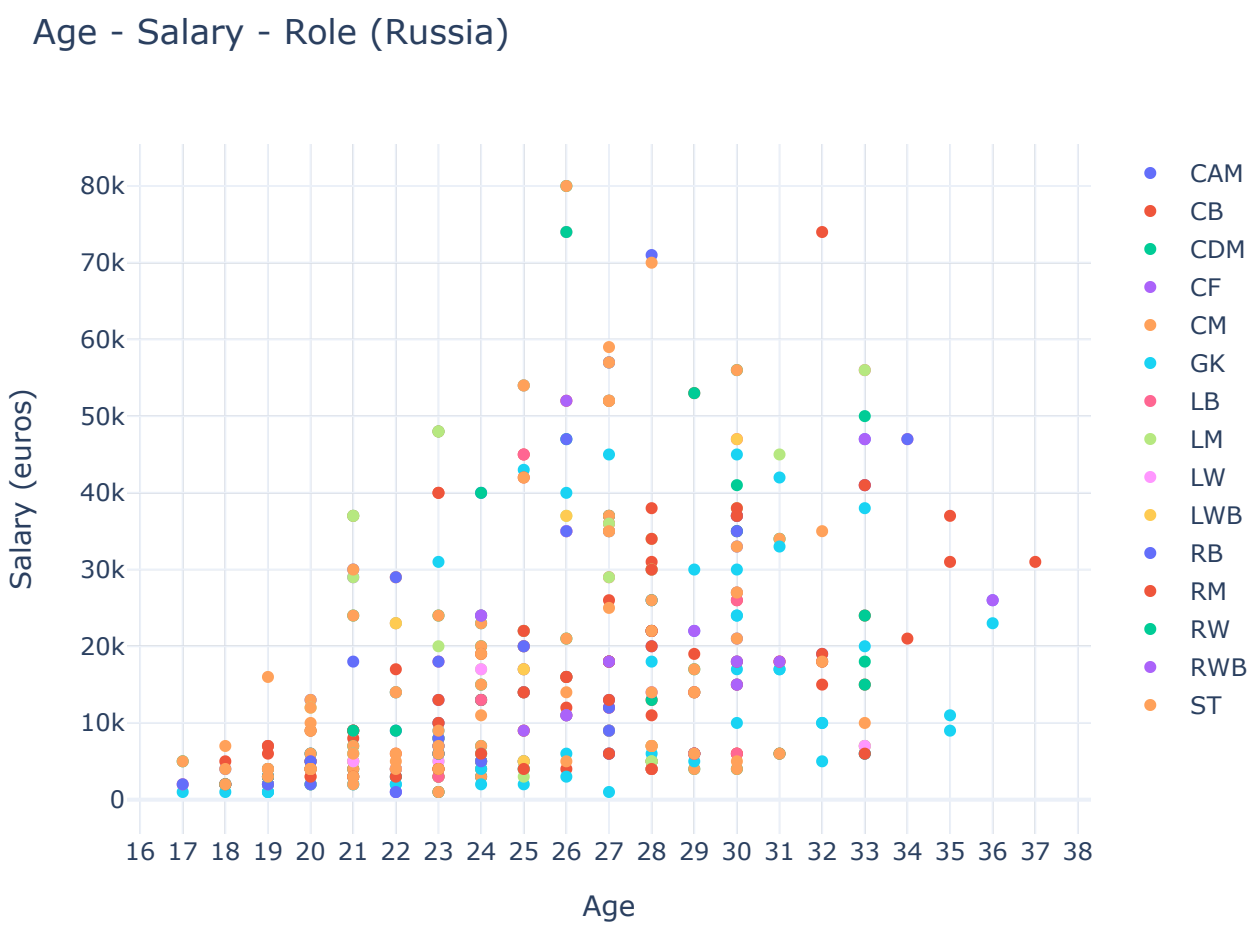
5. Age - Salary - Position scatter

Let's build a scatter for Russian players to get their salary, age and roles together.

In [11]:

```
partial_ds = ds.loc[ds['Nationality']=='Russia']
```

```
roles = sorted(partial_ds["Roles"].unique())

traces = list()
for role in roles:
    _ds = partial_ds.loc[partial_ds['Roles']==role]
    traces.append(
        go.Scatter(
            y=_ds['Wage'],
            x=_ds['Age'],
            name=role,
            mode='markers',
            text=_ds['Name'] + ' - ' + _ds['Club']
        )
    )

layout = go.Layout(
    title= 'Age - Salary - Role (Russia)',
    xaxis= dict(
        title= 'Age',
        dtick = 1
    ),
    yaxis=dict(
        title= 'Salary (euros)',
        dtick = 10000,
        tick0 = 0
    )
)
fig = go.Figure(data=traces, layout=layout)
offline.iplot(fig)
```

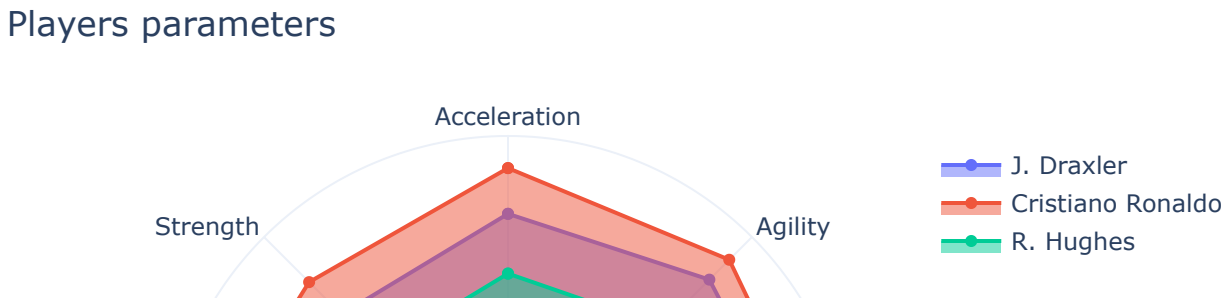


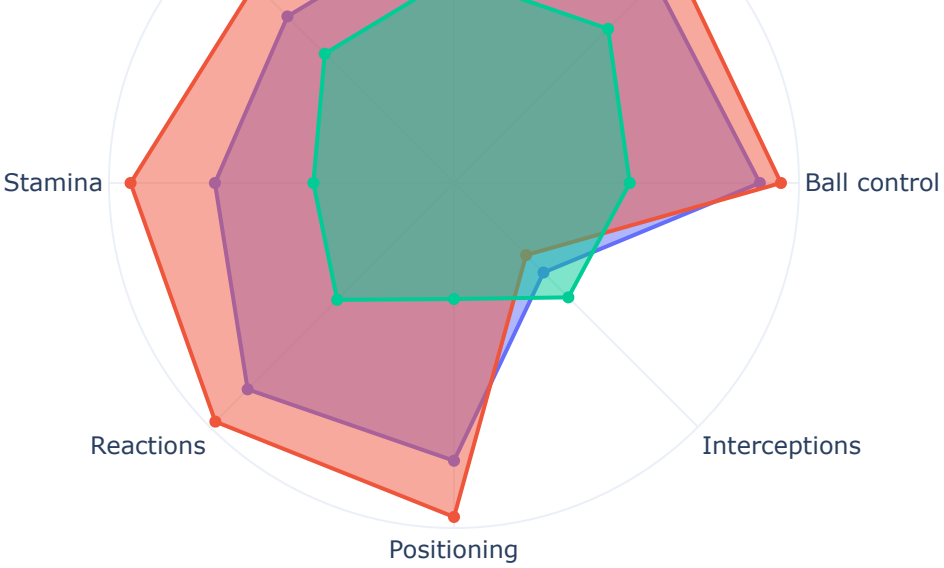
6. Comparing players

Let's take a couple of players and compare their characteristics

In [12]:
```
characteristics = [
    'Acceleration',
    'Agility',
    'Ball control',
    'Interceptions',
    'Positioning',
    'Reactions',
    'Stamina',
    'Strength'
]
# beware namesakes
players = [
    'J. Draxler',
    'Cristiano Ronaldo',
    'R. Hughes'
]
player_characteristics = complete_ds[['Name'] + characteristics].copy()
player_characteristics[characteristics] = (
    player_characteristics[characteristics].applymap(lambda x: int(x.split('+')[0].split('
```

In [13]:
```
traces = list()
for player in players:
    _ds = player_characteristics.loc[player_characteristics['Name']==player]
    traces.append(
        go.Scatterpolar(
              name = player,
              r = _ds[characteristics].values.tolist()[0] + [_ds[characteristics].values.t
              theta = characteristics + [characteristics[0]],
              hovertemplate='%{r}',
              hoverinfo='r',
            )
    )
fig = go.Figure(data=traces)
fig.update_traces(fill='toself')
fig.update_layout(
    title= 'Players parameters',
    polar = dict(
      angularaxis = dict(
        direction = "clockwise",
      ),
      radialaxis = dict(
        visible = False,
        showticklabels = False
      )
    )
)
offline.iplot(fig)
```




Stamina
Ball control
Reactions
Interceptions
Positioning