In [1]:
```
# Import des librairies
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
import seaborn.objects as so
from datetime import datetime
import scipy.stats as stats
```

Customers

In [2]:
```
customers = pd.read_csv(r'customers.csv', sep=';')
print(customers.head())
```

```
  client_id sex  birth
0    c_4410   f   1967
1    c_7839   f   1975
2    c_1699   f   1984
3    c_5961   f   1962
4    c_5320   m   1943
```

In [3]:
```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8621 entries, 0 to 8620
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   client_id  8621 non-null   object
 1   sex        8621 non-null   object
 2   birth      8621 non-null   int64 
dtypes: int64(1), object(2)
memory usage: 202.2+ KB
```

In [4]:
```
customers.isna().sum()
```

Out[4]:
```
client_id    0
sex          0
birth        0
dtype: int64
```

In [5]:
```
customers.duplicated().sum()
```

Out[5]:
```
0
```

Products

In [6]:
```
products = pd.read_csv(r'products.csv', sep=';')
print(products.head())
```

```
  id_prod  price  categ
0  0_1421  19.99      0
1  0_1368   5.13      0
2   0_731  17.99      0
3   1_587   4.99      1
4  0_1507   3.99      0
```

In [7]:
```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3286 entries, 0 to 3285
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   id_prod  3286 non-null   object 
 1   price    3286 non-null   float64
 2   categ    3286 non-null   int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 77.1+ KB
```

In [8]:
```
products.isna().sum()
```

Out[8]:
```
id_prod    0
price      0
categ      0
dtype: int64
```

In [9]:
```
products.duplicated().sum()
```

Out[9]:
```
0
```

Transactions

In [10]:
```
transactions = pd.read_csv(r'transactions.csv', sep=';')
print(transactions.head())
```

```
  id_prod                       date session_id client_id
0  0_1259  2021-03-01 00:01:07.843138        s_1     c_329
1  0_1390  2021-03-01 00:02:26.047414        s_2     c_664
2  0_1352  2021-03-01 00:02:38.311413        s_3     c_580
3  0_1458  2021-03-01 00:04:54.559692        s_4    c_7912
4  0_1358  2021-03-01 00:05:18.801198        s_5    c_2033
C:\Users\maxim\AppData\Local\Temp\ipykernel_8252\161402444.py:1: DtypeWarning: Columns (0,1,2,3) have mixed types. Specify dtype option on import or set low_memory=False.
  transactions = pd.read_csv(r'transactions.csv', sep=';')
```

In [11]:
```
transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   id_prod     687534 non-null  object
 1   date        687534 non-null  object
 2   session_id  687534 non-null  object
 3   client_id   687534 non-null  object
dtypes: object(4)
memory usage: 32.0+ MB
```

In [12]:
```
transactions.isna().sum()
```

Out[12]:
```
id_prod       361041
date          361041
session_id    361041
client_id     361041
dtype: int64
```

In [13]:
```
# Exploration des valeurs nulles
transactions_na = transactions[transactions.isnull().any(axis=1)]
print(transactions_na.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 361041 entries, 687534 to 1048574
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   id_prod     0 non-null       object
 1   date        0 non-null       object
 2   session_id  0 non-null       object
 3   client_id   0 non-null       object
dtypes: object(4)
memory usage: 13.8+ MB
None
```

In [14]:
```
# Ces lignes sont totalement nulles, suppression de ces lignes
transactions_ok = ~transactions.isnull().any(axis=1)
transactions = transactions[transactions_ok]
print(transactions.isna().sum())
print(transactions.info())
```

```
id_prod       0
date          0
session_id    0
client_id     0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Index: 687534 entries, 0 to 687533
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   id_prod     687534 non-null  object
 1   date        687534 non-null  object
 2   session_id  687534 non-null  object
 3   client_id   687534 non-null  object
dtypes: object(4)
memory usage: 26.2+ MB
None
```

In [15]:
```
# Correction du format date
transactions['date'] = pd.to_datetime(transactions['date'])
transactions['date_day'] = transactions['date'].dt.date
print(transactions.head())
```

```
  id_prod                       date session_id client_id    date_day
0  0_1259 2021-03-01 00:01:07.843138        s_1     c_329  2021-03-01
1  0_1390 2021-03-01 00:02:26.047414        s_2     c_664  2021-03-01
2  0_1352 2021-03-01 00:02:38.311413        s_3     c_580  2021-03-01
3  0_1458 2021-03-01 00:04:54.559692        s_4    c_7912  2021-03-01
4  0_1358 2021-03-01 00:05:18.801198        s_5    c_2033  2021-03-01
```

Fusion des fichiers

In [16]:
```
#Fusion à gauche pour avoir chaque infos produits liées à la transaction
merged = pd.merge(transactions, products, on='id_prod', how='left')
```

In [17]:
```
# Vérification
print(merged.shape)
print(merged.head())
print(merged.isna().sum())
```

```
(687534, 7)
  id_prod                       date session_id client_id    date_day  price  \
0  0_1259 2021-03-01 00:01:07.843138        s_1     c_329  2021-03-01  11.99
1  0_1390 2021-03-01 00:02:26.047414        s_2     c_664  2021-03-01  19.37
2  0_1352 2021-03-01 00:02:38.311413        s_3     c_580  2021-03-01   4.50
3  0_1458 2021-03-01 00:04:54.559692        s_4    c_7912  2021-03-01   6.55
4  0_1358 2021-03-01 00:05:18.801198        s_5    c_2033  2021-03-01  16.49

   categ
0      0
1      0
2      0
3      0
4      0
id_prod       0
date          0
session_id    0
client_id     0
date_day      0
price         0
categ         0
dtype: int64
id_prod       0
date          0
session_id    0
client_id     0
date_day      0
price         0
categ         0
dtype: int64
```

In [18]:
```
# Fusion du fichier customers
merged = pd.merge(merged, customers, on='client_id', how='left')
```

In [19]:
```
# Vérification
print(merged.shape)
print(merged.head())
print(merged.isna().sum())
```

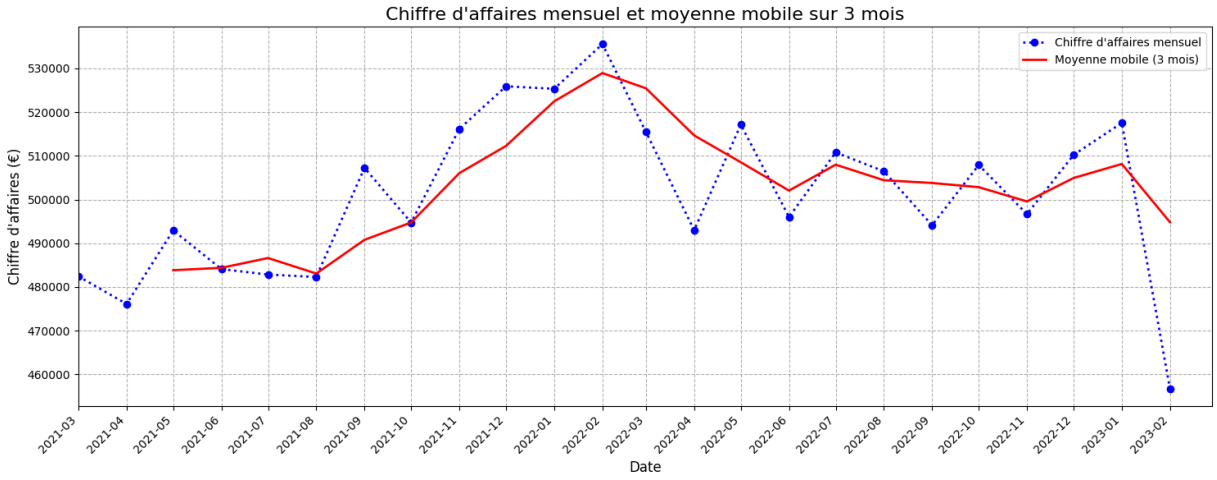
```
(687534, 9)
  id_prod                       date session_id client_id    date_day  price  \
0  0_1259 2021-03-01 00:01:07.843138        s_1     c_329  2021-03-01  11.99
1  0_1390 2021-03-01 00:02:26.047414        s_2     c_664  2021-03-01  19.37
2  0_1352 2021-03-01 00:02:38.311413        s_3     c_580  2021-03-01   4.50
3  0_1458 2021-03-01 00:04:54.559692        s_4    c_7912  2021-03-01   6.55
4  0_1358 2021-03-01 00:05:18.801198        s_5    c_2033  2021-03-01  16.49

   categ sex  birth
0      0   f   1967
1      0   m   1960
2      0   m   1988
3      0   f   1989
4      0   f   1956
id_prod       0
date          0
session_id    0
client_id     0
date_day      0
price         0
categ         0
sex           0
birth         0
dtype: int64
id_prod       0
date          0
session_id    0
client_id     0
date_day      0
price         0
categ         0
sex           0
birth         0
dtype: int64
```

Moyenne mobile du CA

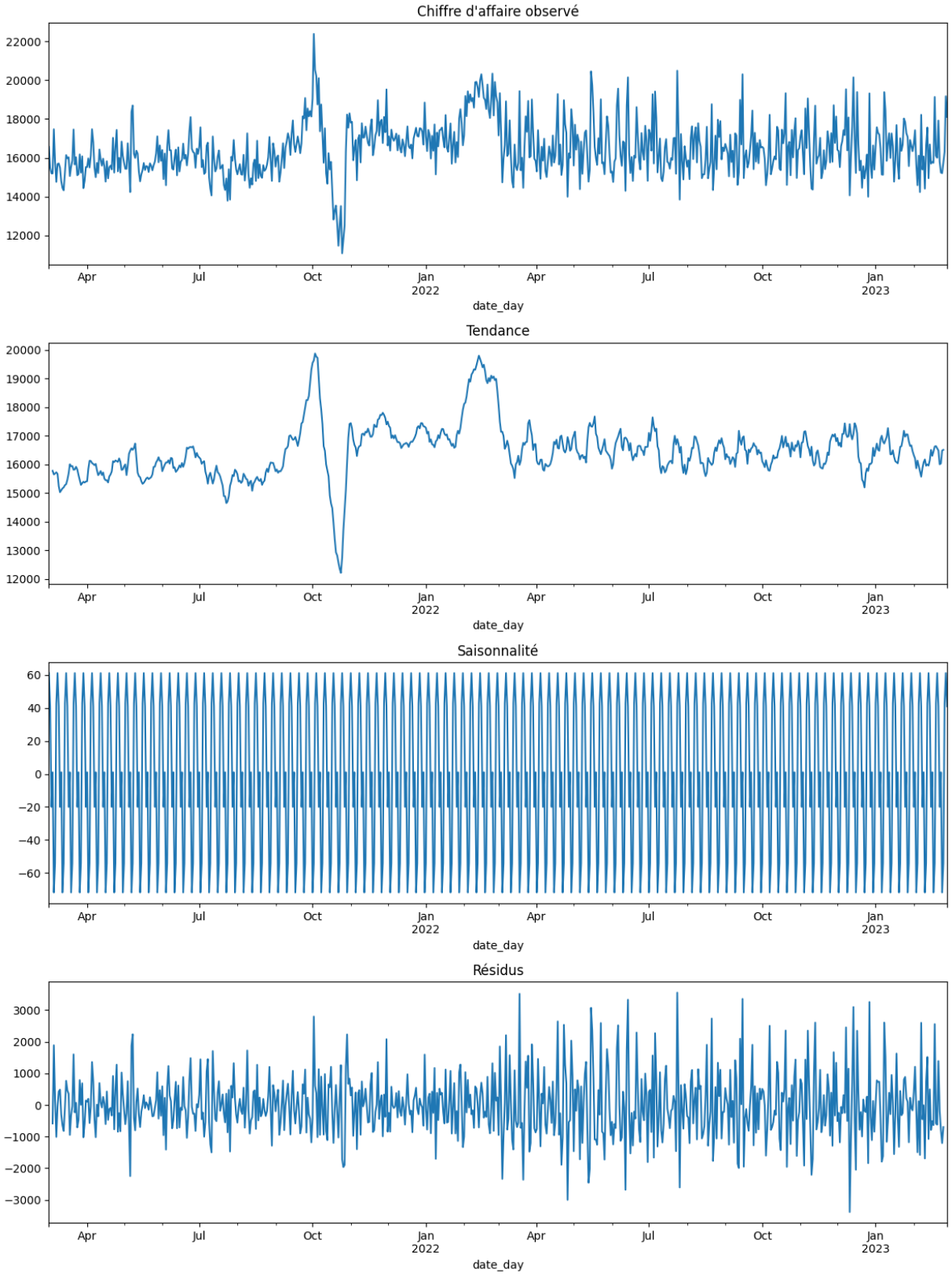
In [20]:
```python
# Date placé en index, mise au bon format, calcul du CA par mois avec resample('MS'
merged.set_index('date_day', inplace=True)
merged.index = pd.to_datetime(merged.index)
monthly_revenue = merged['price'].resample('MS').sum()
rolling_avg_revenue = monthly_revenue.rolling(window=3).mean()
```

In [21]:
```python
# Visualisation
# Représentation du CA et de la moyenne mobile pour comparer les 2 valeurs, besoin
# évite de toucher la colonne date pour l'instant

fig, ax = plt.subplots(figsize=(15, 6))

ax.plot(monthly_revenue.index, monthly_revenue.values, label='Chiffre d\'affaires m
ax.plot(rolling_avg_revenue.index, rolling_avg_revenue.values, label='Moyenne mobil
ax.set_title('Chiffre d\'affaires mensuel et moyenne mobile sur 3 mois', fontsize=1
ax.set_xlabel('Date', fontsize=12)
ax.set_ylabel('Chiffre d\'affaires (€)', fontsize=12)
```

```
ax.legend(fontsize=10)

ax.xaxis.set_major_locator(mdates.MonthLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))

plt.xticks(rotation=45, ha='right')
ax.set_xlim(pd.to_datetime('2021-03-01'), pd.to_datetime('2023-02-28'))
ax.grid(True, linestyle='--', alpha=1)
plt.tight_layout()
plt.show()
```

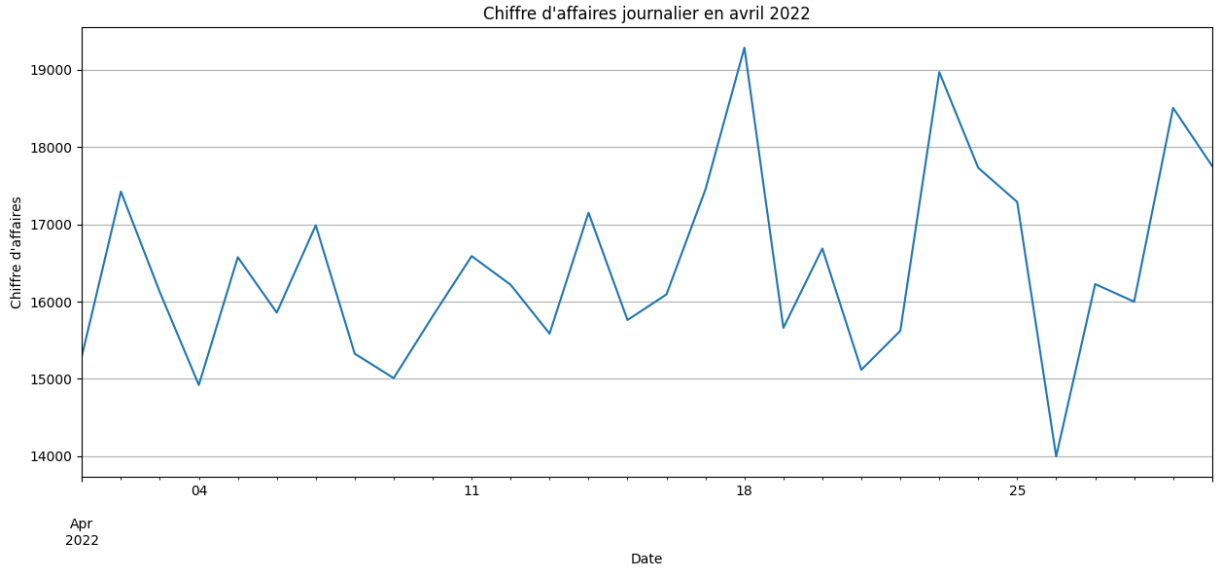


In [23]:
```
# Décomposition en utilisant le CA journalier plutôt que mensuel
daily_revenue = merged['price'].resample('D').sum()
decomposition = seasonal_decompose(daily_revenue, model='additive', period=7)
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(12, 16))

#Données réelles observées
decomposition.observed.plot(ax=ax1)
ax1.set_title('Chiffre d\'affaire observé')

#Tendance
decomposition.trend.plot(ax=ax2)
ax2.set_title('Tendance')

#Saisonnalité
decomposition.seasonal.plot(ax=ax3)
ax3.set_title('Saisonnalité')

#Résidus
decomposition.resid.plot(ax=ax4)
ax4.set_title('Résidus')

plt.tight_layout()
plt.show()
```

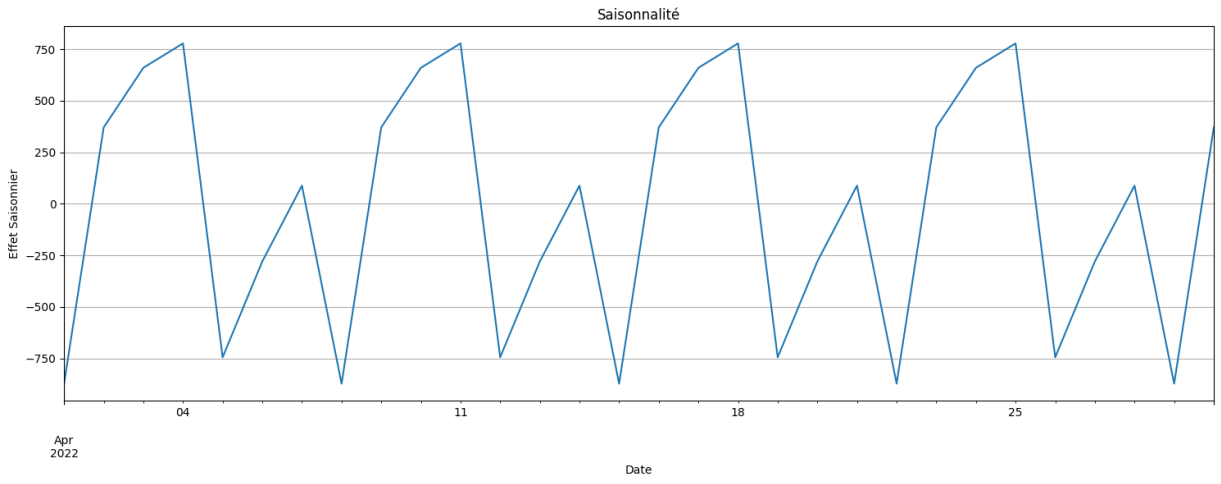


In [24]:
```
# Selection d'un mois au hasard pour étudier la saisonnalité par jour plus en détai
daily_revenue.index = pd.to_datetime(daily_revenue.index)
april22_data = daily_revenue['2022-04-01':'2022-04-30']
plt.figure(figsize=(15, 6))
april22_data.plot()
plt.title("Chiffre d'affaires journalier en avril 2022")
plt.xlabel("Date")
```

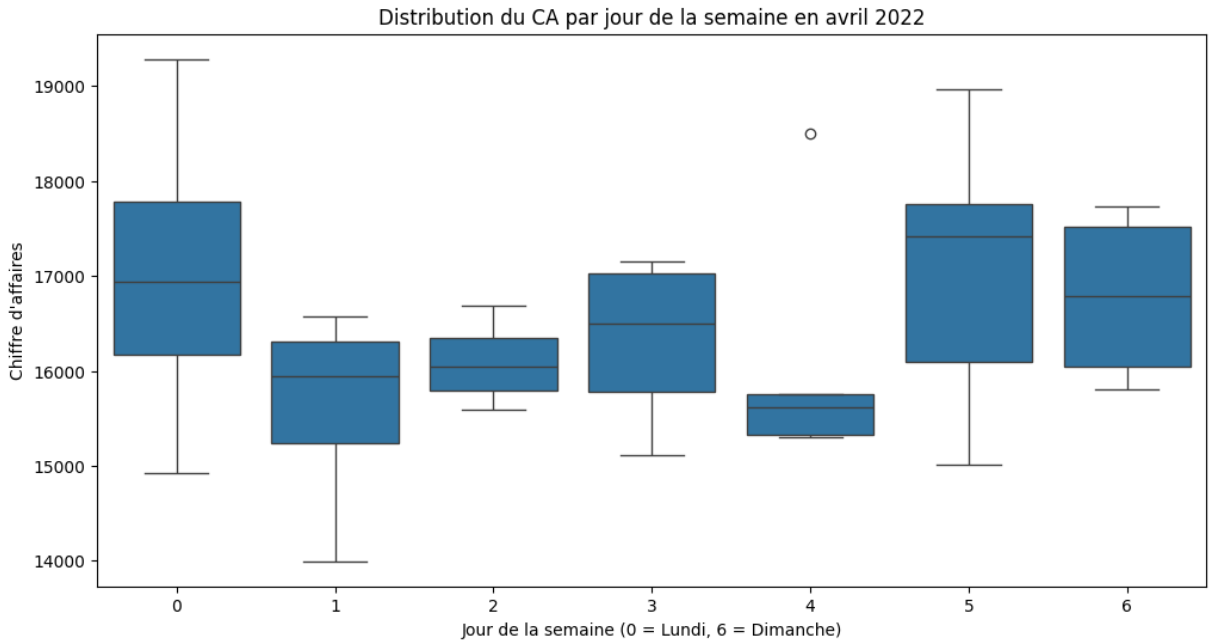
```
plt.ylabel("Chiffre d'affaires")
plt.grid(True)
plt.show()
```



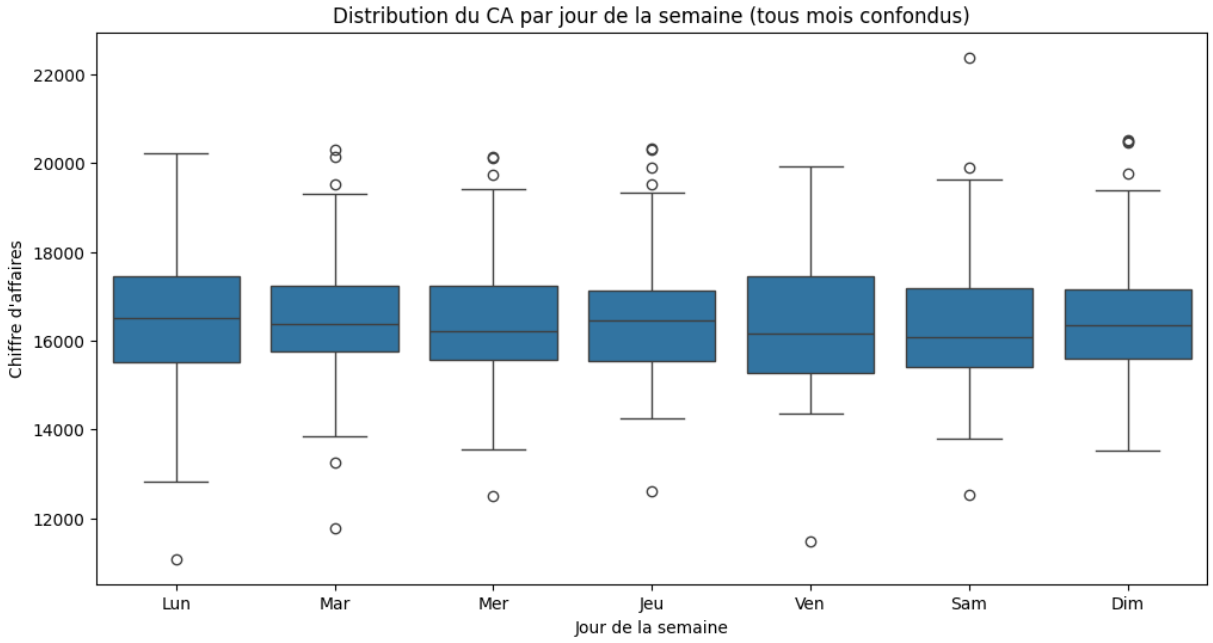
In [25]:
```
decomposition = seasonal_decompose(april22_data, model='additive', period=7)
plt.figure(figsize=(15, 6))
decomposition.seasonal.plot()
plt.title('Saisonnalité')
plt.xlabel('Date')
plt.ylabel('Effet Saisonnier')
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [26]:
```
april22_df = april22_data.to_frame(name='CA')
april22_df['day_of_week'] = april22_df.index.to_series().dt.dayofweek
plt.figure(figsize=(12, 6))
sns.boxplot(x='day_of_week', y='CA', data=april22_df)
plt.title("Distribution du CA par jour de la semaine en avril 2022")
plt.xlabel("Jour de la semaine (0 = Lundi, 6 = Dimanche)")
plt.ylabel("Chiffre d'affaires")
plt.show()
```

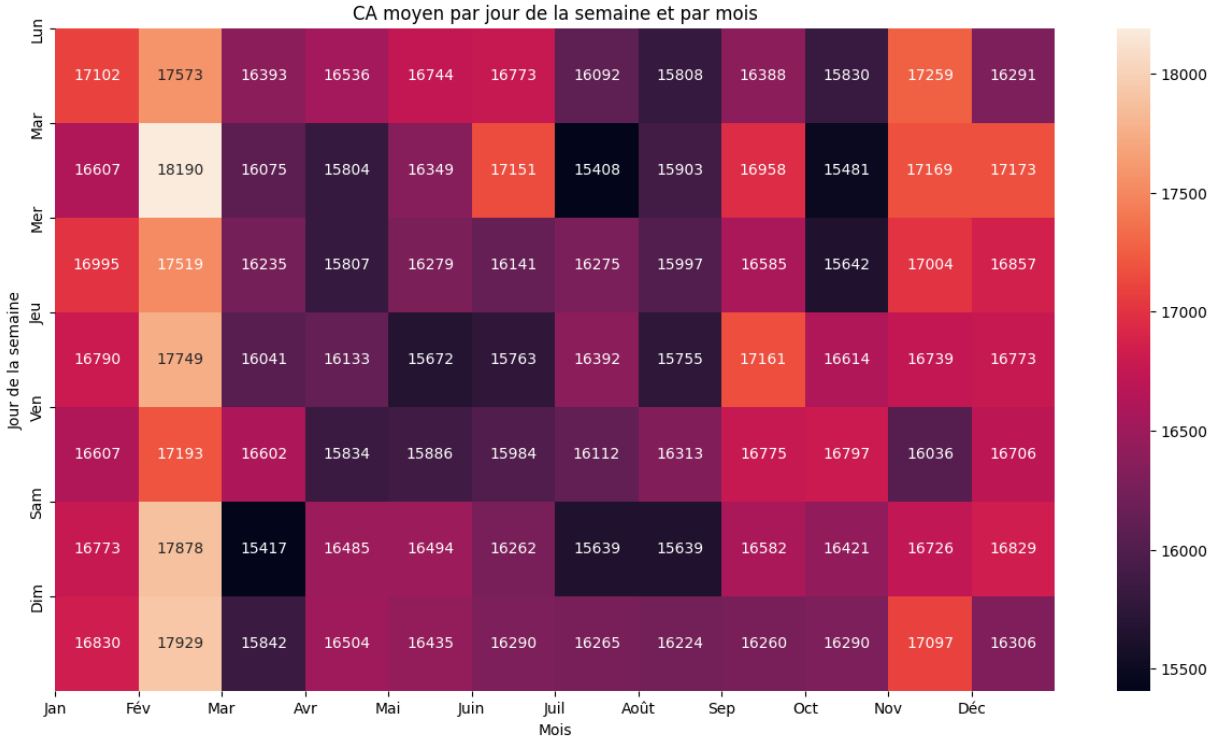


In [27]:
```
# L'analyse n'est que sur le mois d'avril 22 qui servait de point de départ de réfl
all_data = daily_revenue.copy()
all_data = all_data.to_frame(name='CA')
all_data['month'] = all_data.index.month
all_data['day_of_week'] = all_data.index.dayofweek
plt.figure(figsize=(12, 6))
sns.boxplot(x='day_of_week', y='CA', data=all_data)
plt.title("Distribution du CA par jour de la semaine (tous mois confondus)")
plt.xlabel("Jour de la semaine")
plt.ylabel("Chiffre d'affaires")
plt.xticks(range(7), ['Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam', 'Dim'])
plt.show()
```



In [28]:
```
pivot_data = all_data.pivot_table(values='CA', index='day_of_week', columns='month'

plt.figure(figsize=(15, 8))
```

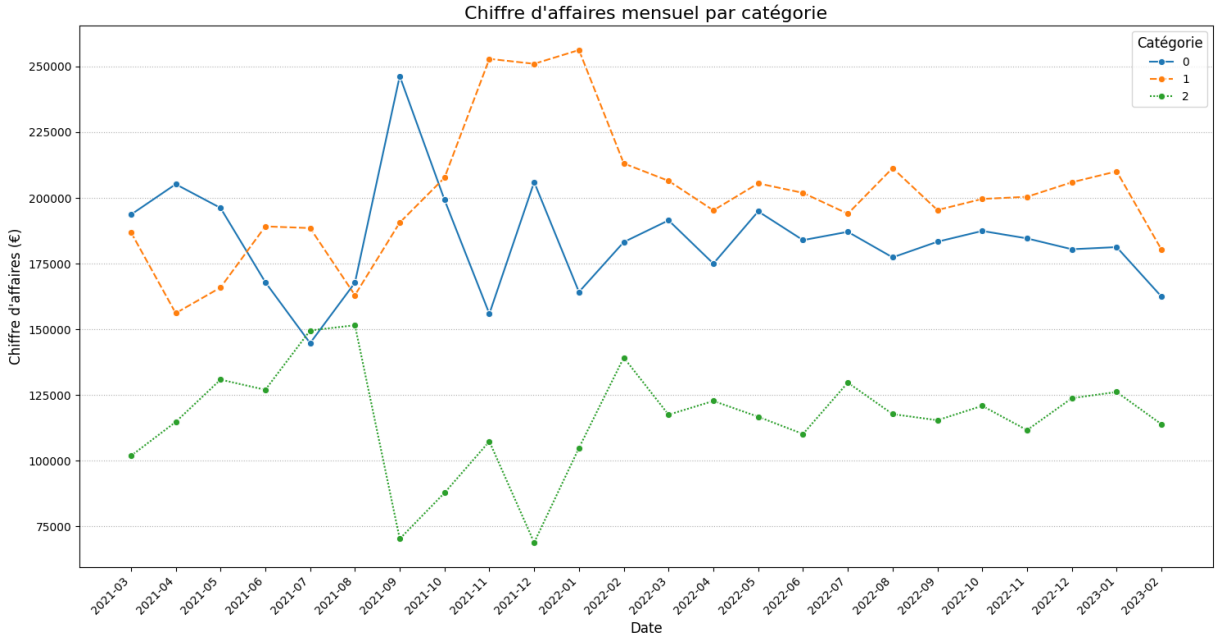
```
sns.heatmap(pivot_data, annot=True, fmt='.0f')
plt.title("CA moyen par jour de la semaine et par mois")
plt.xlabel("Mois")
plt.ylabel("Jour de la semaine")
plt.yticks(range(7), ['Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam', 'Dim'])
plt.xticks(range(12), ['Jan', 'Fév', 'Mar', 'Avr', 'Mai', 'Juin', 'Juil', 'Août', '
plt.show()
```



Chiffre d'affaire par catégorie

In [29]:

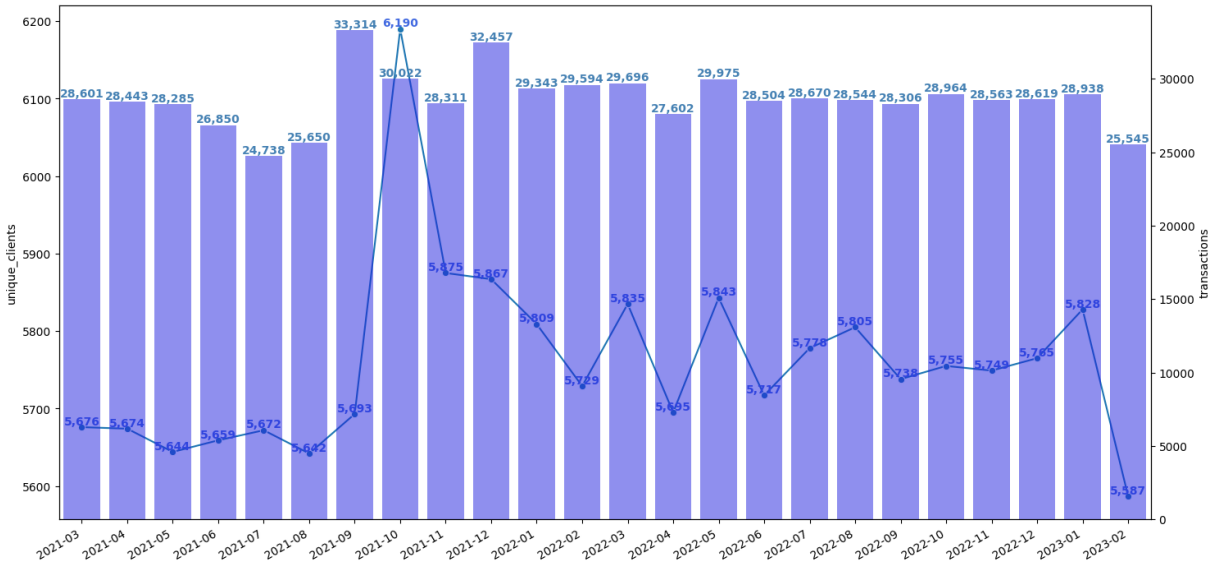
```
#Création de périodes mensuelles pout calculer le CA/mois
merged['date_month'] = merged['date'].dt.to_period('M').astype(str)

#Calcul
monthly_revenue = merged.groupby(['date_month', 'categ'])['price'].sum().reset_inde

#Graphique
plt.figure(figsize=(15, 8))
sns.lineplot(data=monthly_revenue, x='date_month', y='price', hue='categ', marker='

#Esthétique
plt.title('Chiffre d\'affaires mensuel par catégorie', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Chiffre d\'affaires (€)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Catégorie', title_fontsize='12', fontsize='10', loc='upper right'
plt.grid(linestyle=':', axis='y')
plt.tight_layout()
plt.show()
```



Clients par mois / Clients et transactions par mois

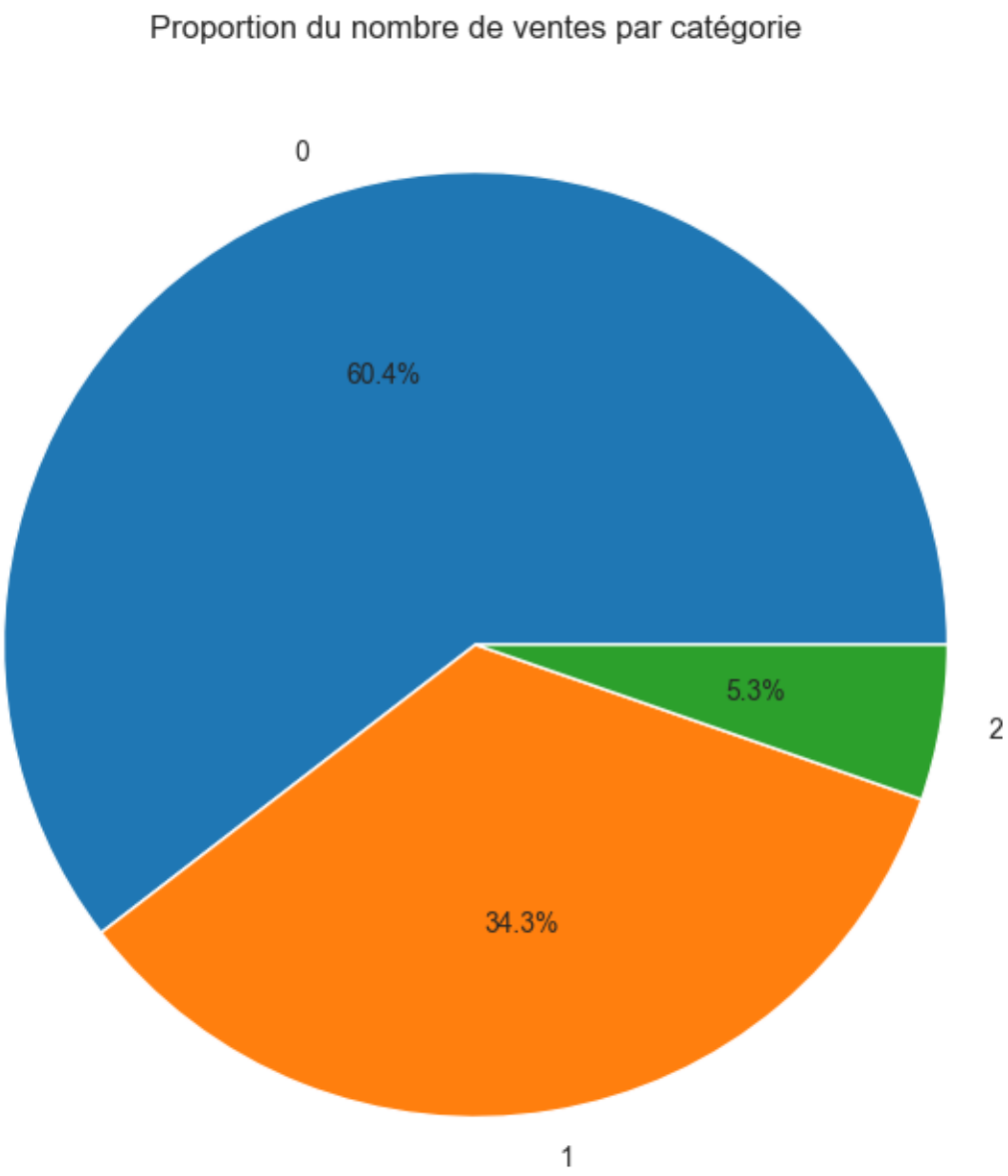
In [30]:

```
# Calcul du nombre de clients uniques et du total de transactions par mois
clients_count = merged.groupby('date_month')['client_id'].nunique()
transactions_count = merged.groupby('date_month').size()
monthly_transacclient = pd.DataFrame({'unique_clients': clients_count,'transactions

# Graphique
ax1 = sns.set_style(style=None, rc=None )
fig, ax1 = plt.subplots(figsize=(15,8))
sns.lineplot(data = monthly_transacclient['unique_clients'], marker='o', sort=False
ax2 = ax1.twinx()
sns.barplot(data= monthly_transacclient, x='date_month', y='transactions', color="b
for idx, row in monthly_transacclient.iterrows():
    ax1.text(idx, row['unique_clients'], f'{row["unique_clients"]:,.0f}',
             ha='center', va='bottom', color='royalblue', fontweight='bold')
for i, v in enumerate(monthly_transacclient['transactions']):
    ax2.text(i, v, f'{v:,.0f}', ha='center', va='bottom', color='steelblue', fontwe
plt.tight_layout()
fig.autofmt_xdate() #car xticks(rotation=45) ne fonctionne pas je ne sais pas pourq
plt.show()
```



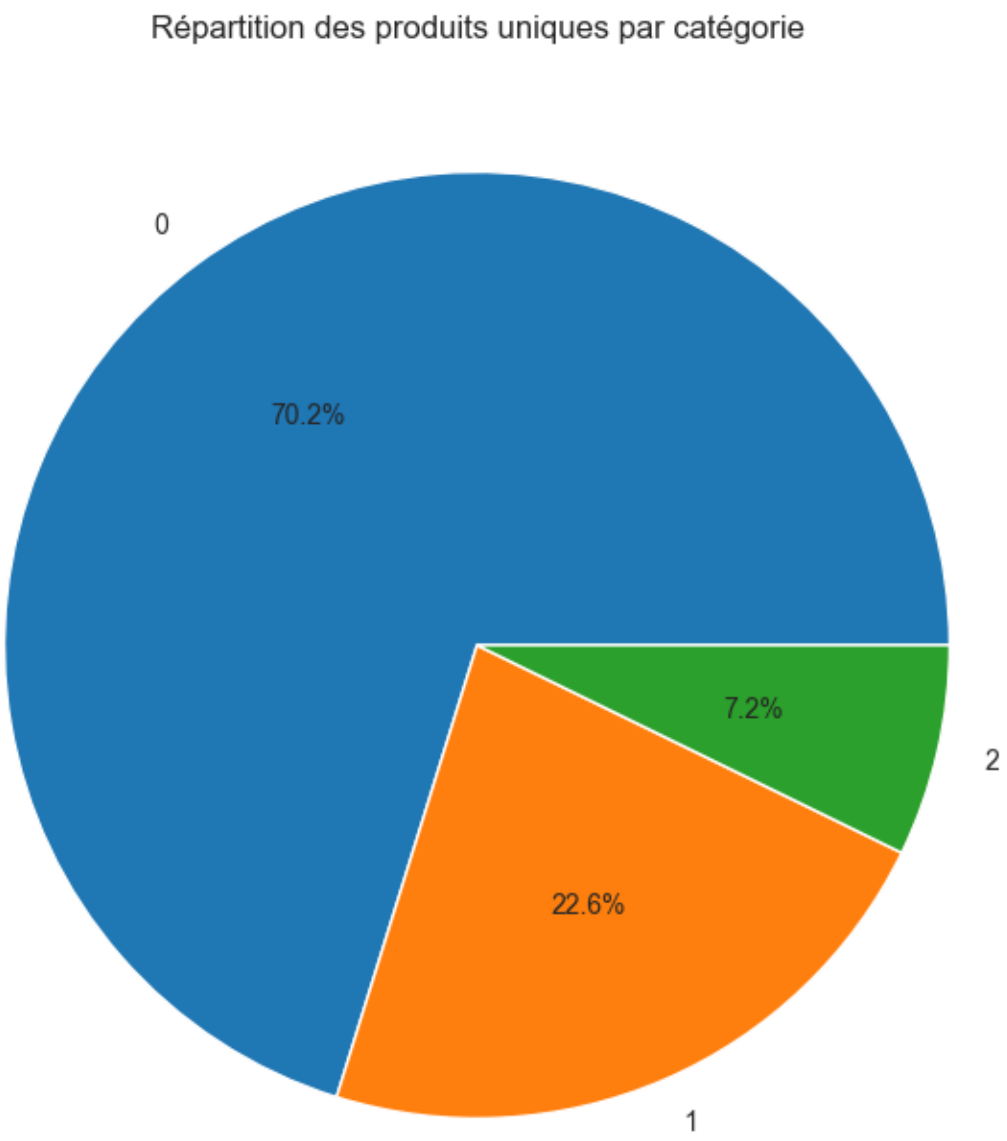
Représentation des catégories vendues

In [31]:
```
category_counts = merged['categ'].value_counts()
sns.set_style("whitegrid")
plt.figure(figsize=(8,8))
plt.pie(category_counts.values, labels=category_counts.index, autopct='%1.1f%%')
plt.title('Proportion du nombre de ventes par catégorie')
plt.show()
```



Références par catégories

In [32]:
```
products_by_category = merged.groupby('categ')['id_prod'].nunique()
plt.figure(figsize=(8, 8))
plt.pie(products_by_category.values, labels=products_by_category.index, autopct='%1
plt.title('Répartition des produits uniques par catégorie')
plt.show()
```



Répartition du chiffre d'affaire selon la catégorie

In [33]:
```
# Calcul du CA par article
total_sales_by_id = merged.groupby(['id_prod'])['price'].sum().reset_index(name='CA
print(total_sales_by_id['CA'].sum())
```

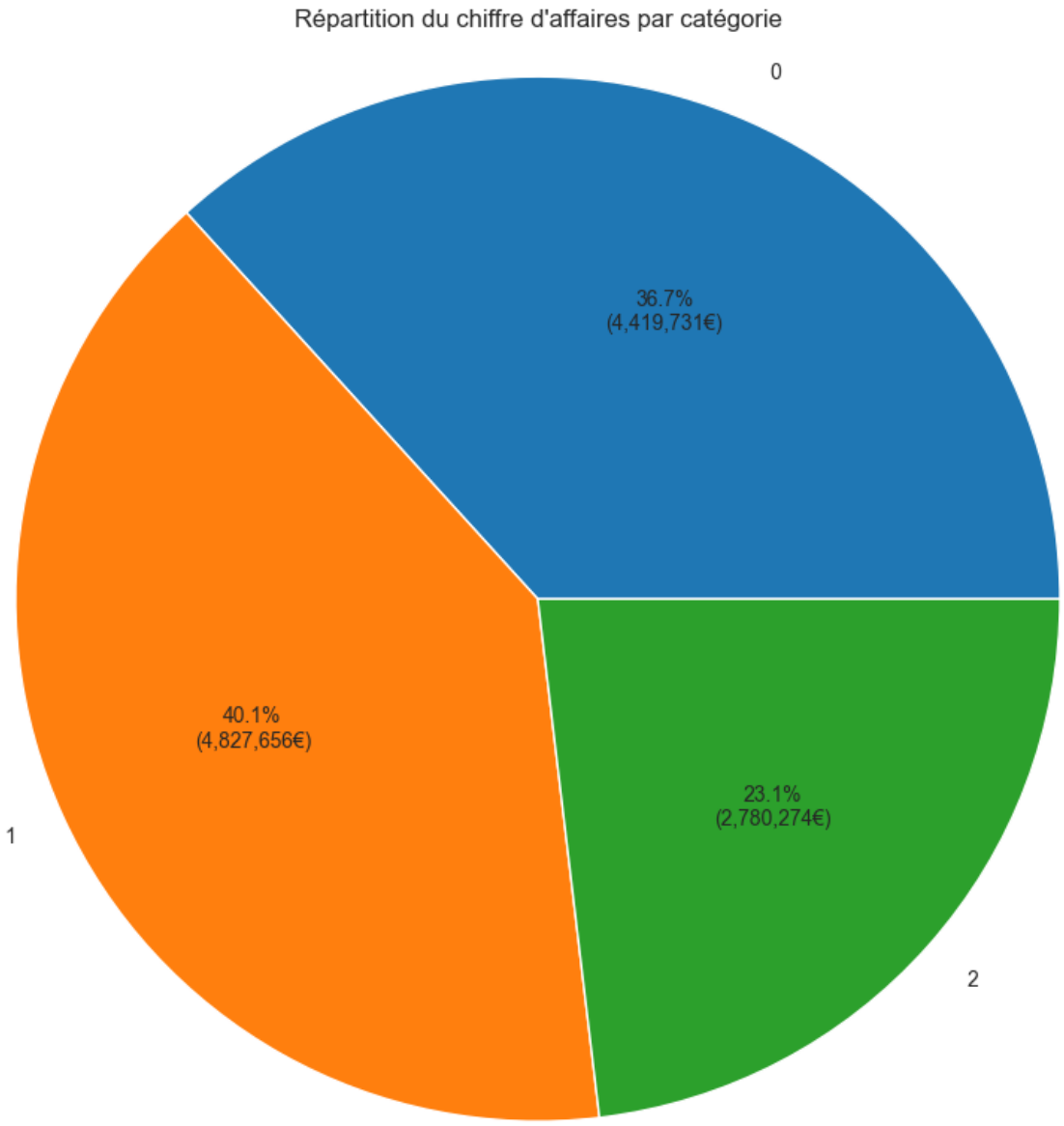
```
12027663.099999998
```

In [34]:
```
# Calcul du CA par catégorie + contrôle de la valeur
total_sales_by_category = merged.groupby('categ')['price'].sum().reset_index(name='
print(total_sales_by_category['CA'].sum())
```

```
12027663.099999998
```

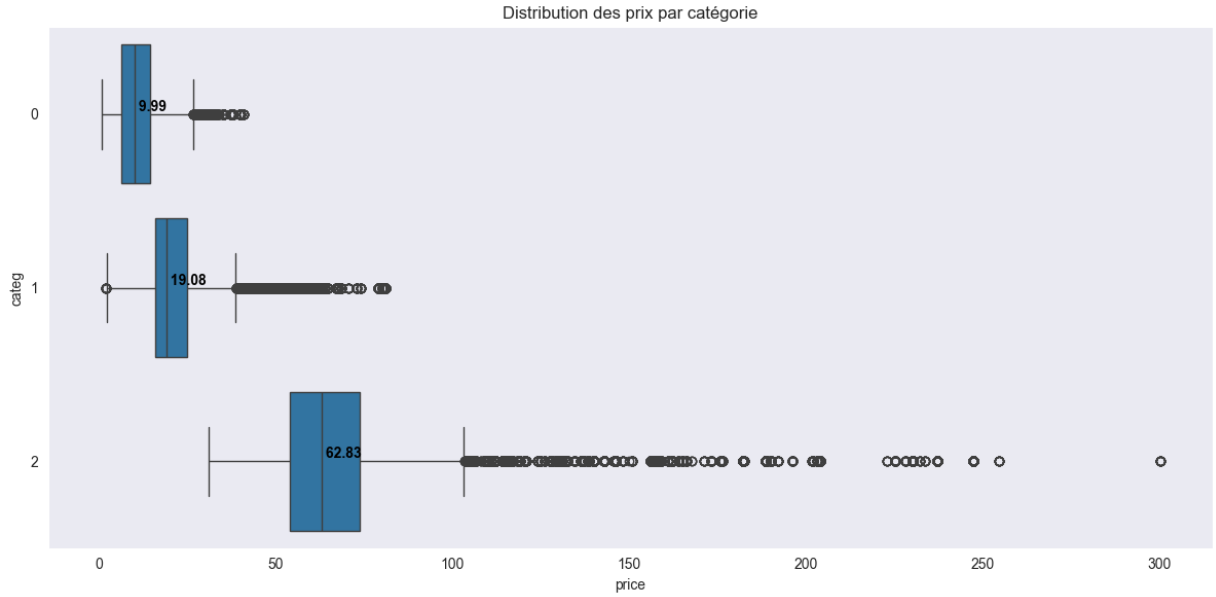
In [35]:
```
plt.figure(figsize=(8, 8))
```

```
plt.pie(total_sales_by_category['CA'], labels=total_sales_by_category['categ'], aut
plt.title('Répartition du chiffre d\'affaires par catégorie')
plt.axis('equal')
plt.tight_layout()
plt.show()
```



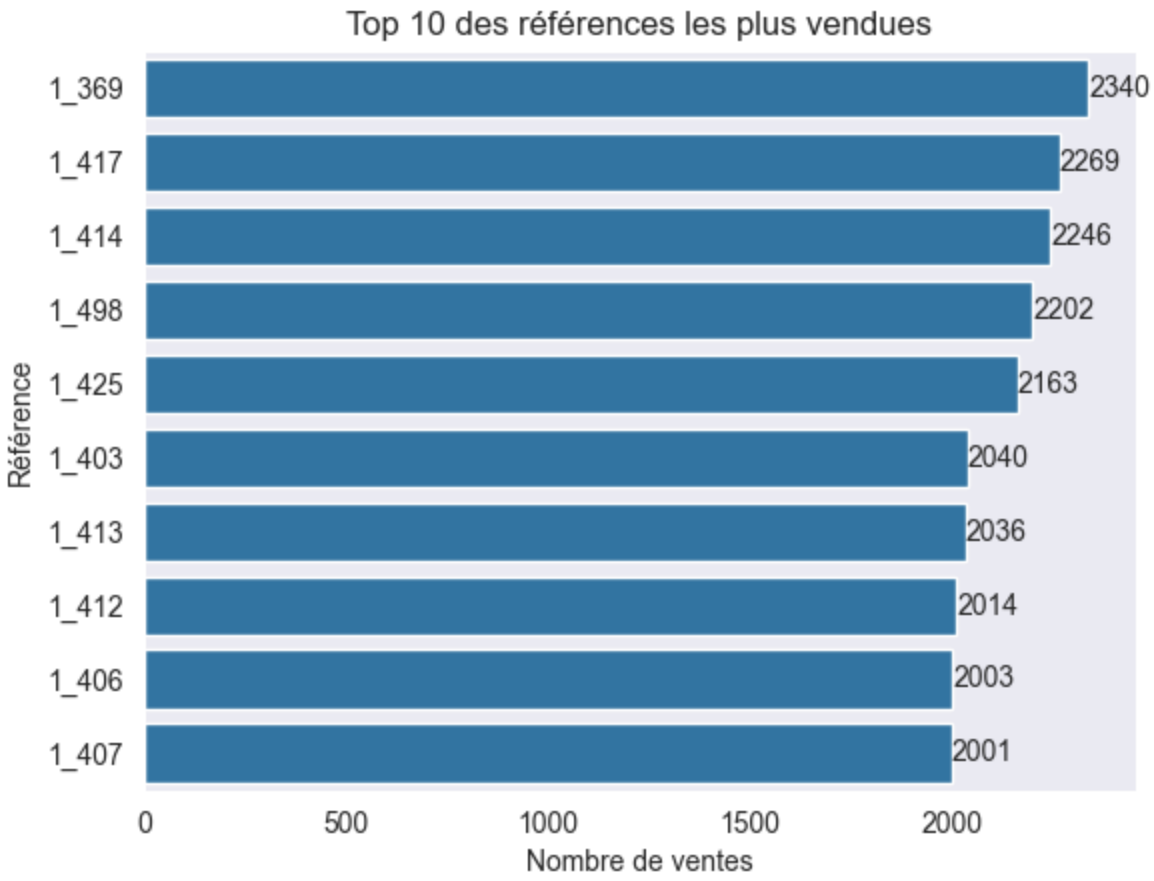
Distribution des prix par catégorie

In [36]:
```
medians = merged.groupby('categ')['price'].median()
plt.figure(figsize=(12,6))
sns.set_style("dark")
ax = sns.boxplot(data=merged, y='categ', x='price', orient='h')
for i, median in enumerate(medians):
    ax.text(median, i, f' {median:.2f}', va='bottom', color='black', fontweight='se
plt.title('Distribution des prix par catégorie')
plt.tight_layout()
plt.show()
```

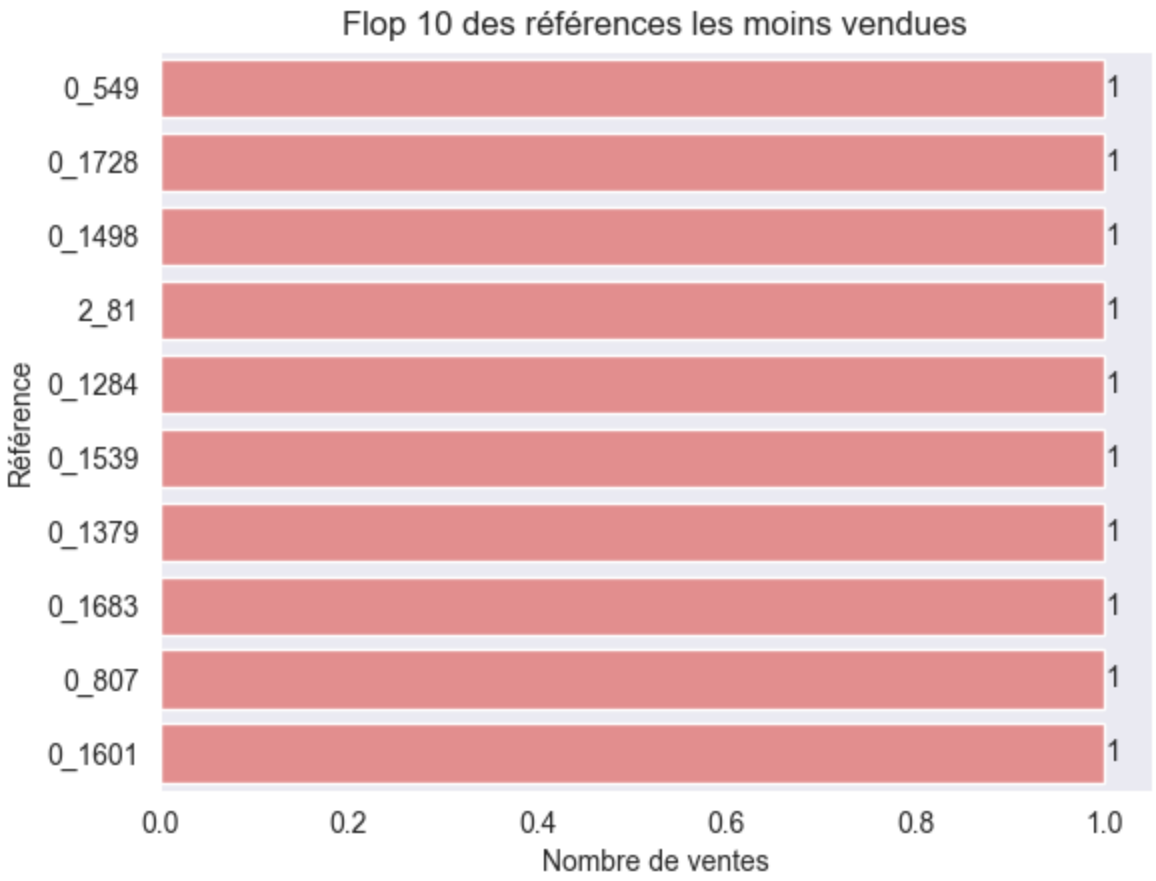


Top et Flop 10 des références

In [37]:
```
top_id = merged['id_prod'].value_counts().nlargest(10)
ax = sns.barplot(data=top_id, orient='h')
ax.bar_label(ax.containers[0])
plt.xlabel("Nombre de ventes")
plt.ylabel('Référence')
plt.title('Top 10 des références les plus vendues')
plt.show()
```



In [38]:
```
flop_id = merged['id_prod'].value_counts().nsmallest(10)
ax = sns.barplot(data=flop_id, orient='h', color='lightcoral')
ax.bar_label(ax.containers[0])
plt.xlabel("Nombre de ventes")
plt.ylabel('Référence')
plt.title('Flop 10 des références les moins vendues')
plt.show()
```



In [39]:
```
# Nombre de références n'ayant qu'une vente
id_count = merged['id_prod'].value_counts()
single_sales = (id_count == 1).sum()
print(f'Nombre de références avec une seule vente : {single_sales}')
```

```
Nombre de références avec une seule vente : 18
```

Clients

In [40]:
```
#Calcul du CA par client
total_sales_by_client = merged.groupby(['client_id'])['price'].sum().reset_index(na
total_sales_by_client.head(10)
```

Out[40]:

	client_id	CA
0	c_1	629.02
1	c_10	1353.60
2	c_100	254.85
3	c_1000	2291.88
4	c_1001	1823.85
5	c_1002	436.54
6	c_1003	1230.83
7	c_1004	1077.83
8	c_1005	844.93
9	c_1006	4029.97

In [41]:
```
#Top 10 des clients
total_sales_by_client = total_sales_by_client.sort_values('CA', ascending=False).re
total_sales_by_client.head(10)
```

Out[41]:

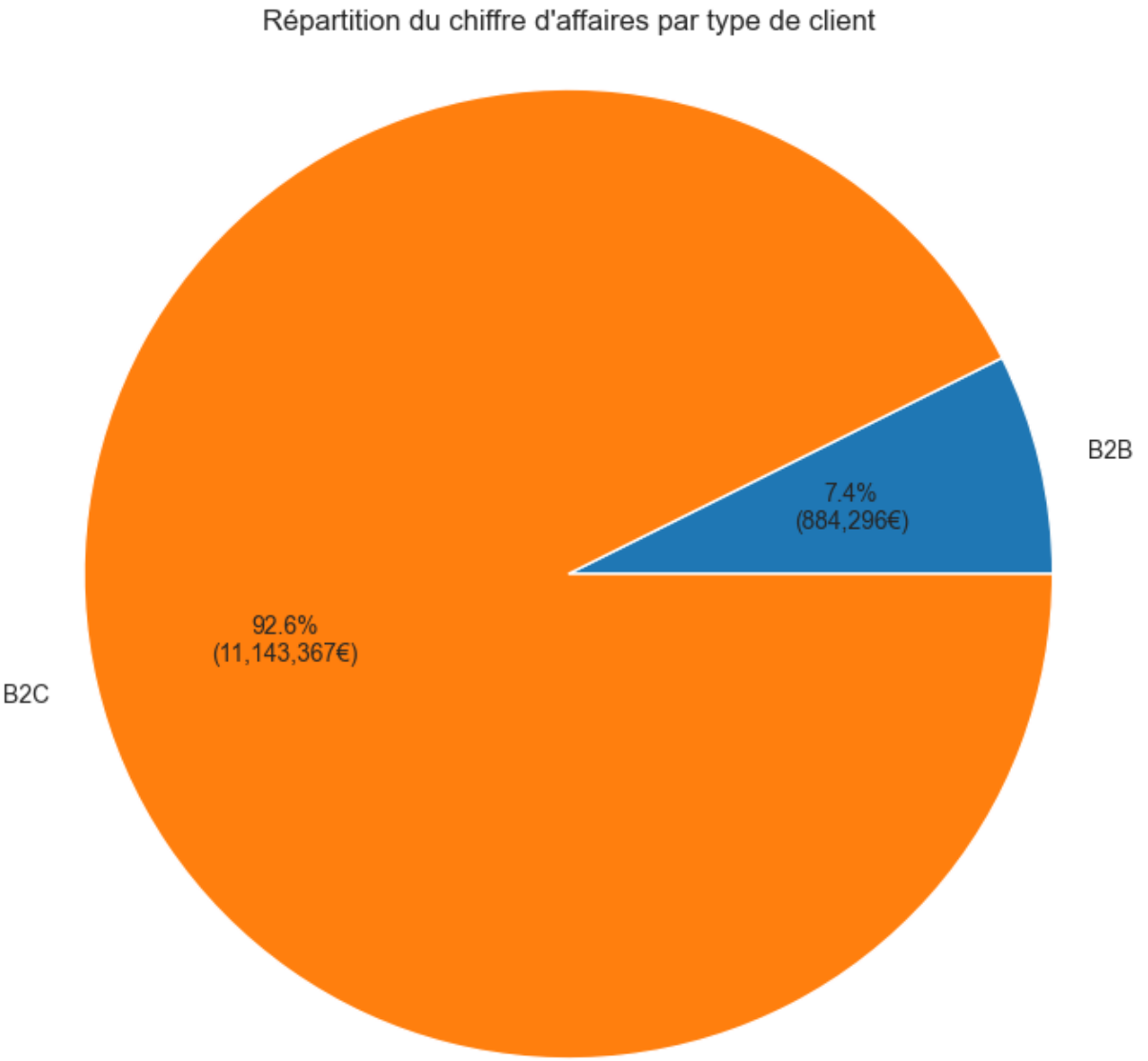
	client_id	CA
0	c_1609	326039.89
1	c_4958	290227.03
2	c_6714	153918.60
3	c_3454	114110.57
4	c_1570	5285.82
5	c_3263	5276.87
6	c_2140	5260.18
7	c_2899	5214.05
8	c_7319	5155.77
9	c_7959	5135.75

On retrouve un CA bien plus élévé pour les 4 premiers clients, on peut considérer que ce sont les clients B2B.

In [42]:
```
total_sales_by_client['client_type'] = 'B2C'
total_sales_by_client.loc[:3, 'client_type'] = 'B2B'

# Afficher les 10 premiers clients
print(total_sales_by_client.head(10))
```

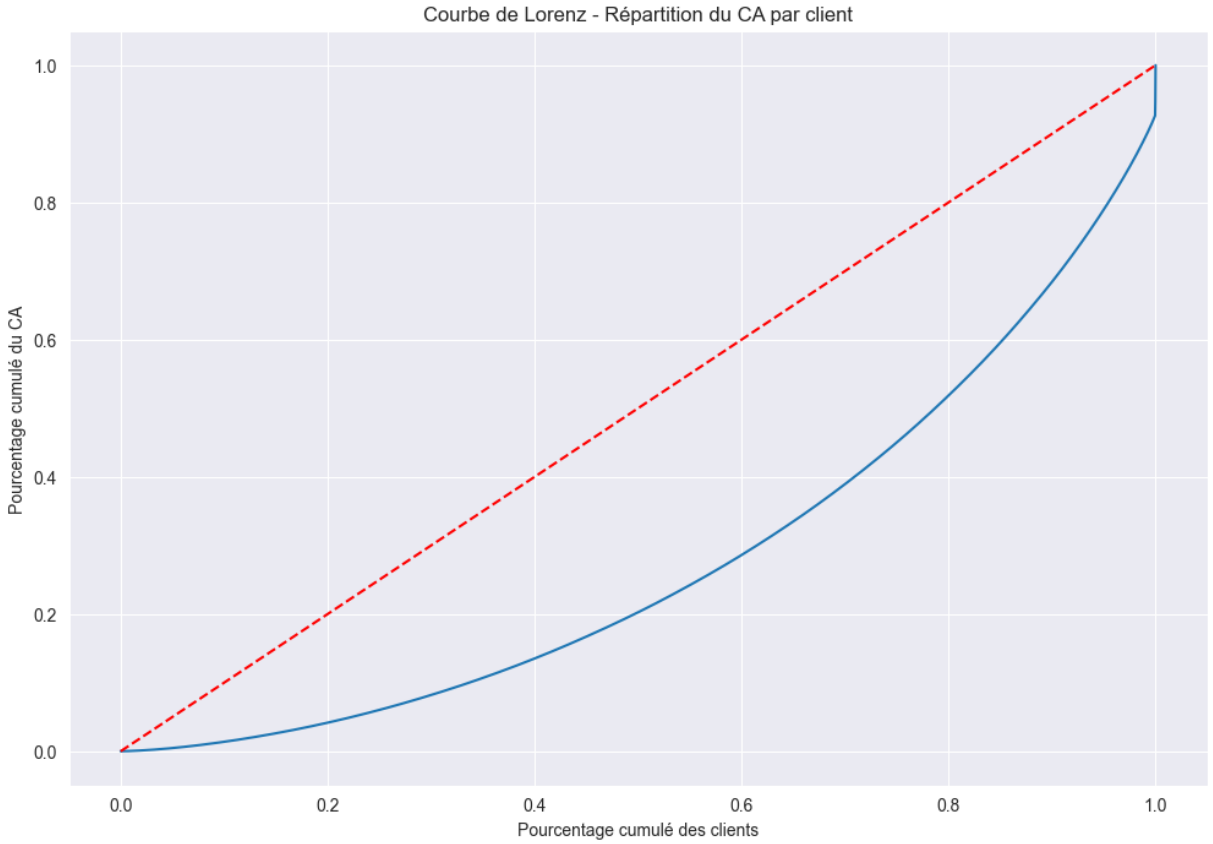
	client_id	CA	client_type
0	c_1609	326039.89	B2B
1	c_4958	290227.03	B2B
2	c_6714	153918.60	B2B
3	c_3454	114110.57	B2B
4	c_1570	5285.82	B2C
5	c_3263	5276.87	B2C
6	c_2140	5260.18	B2C
7	c_2899	5214.05	B2C
8	c_7319	5155.77	B2C
9	c_7959	5135.75	B2C

In [43]:
```
total_sales_by_type = total_sales_by_client.groupby('client_type')['CA'].sum().rese
plt.figure(figsize=(8, 8))
plt.pie(total_sales_by_type['CA'], labels=total_sales_by_type['client_type'], autop
plt.title('Répartition du chiffre d\'affaires par type de client')
plt.axis('equal')
plt.show()
```



In [44]:
```
total_sales_by_client = total_sales_by_client.sort_values('CA')
total_sales_by_client['cum_percent_clients'] = np.arange(1, len(total_sales_by_clie
```

```
total_sales_by_client['cum_percent_ca'] = total_sales_by_client['CA'].cumsum() / to

plt.figure(figsize=(12, 8))
plt.plot([0] + list(total_sales_by_client['cum_percent_clients']), [0] + list(total
plt.plot([0, 1], [0, 1], 'r--')
plt.xlabel('Pourcentage cumulé des clients')
plt.ylabel('Pourcentage cumulé du CA')
plt.title('Courbe de Lorenz - Répartition du CA par client')
plt.grid(True)
plt.show()
```

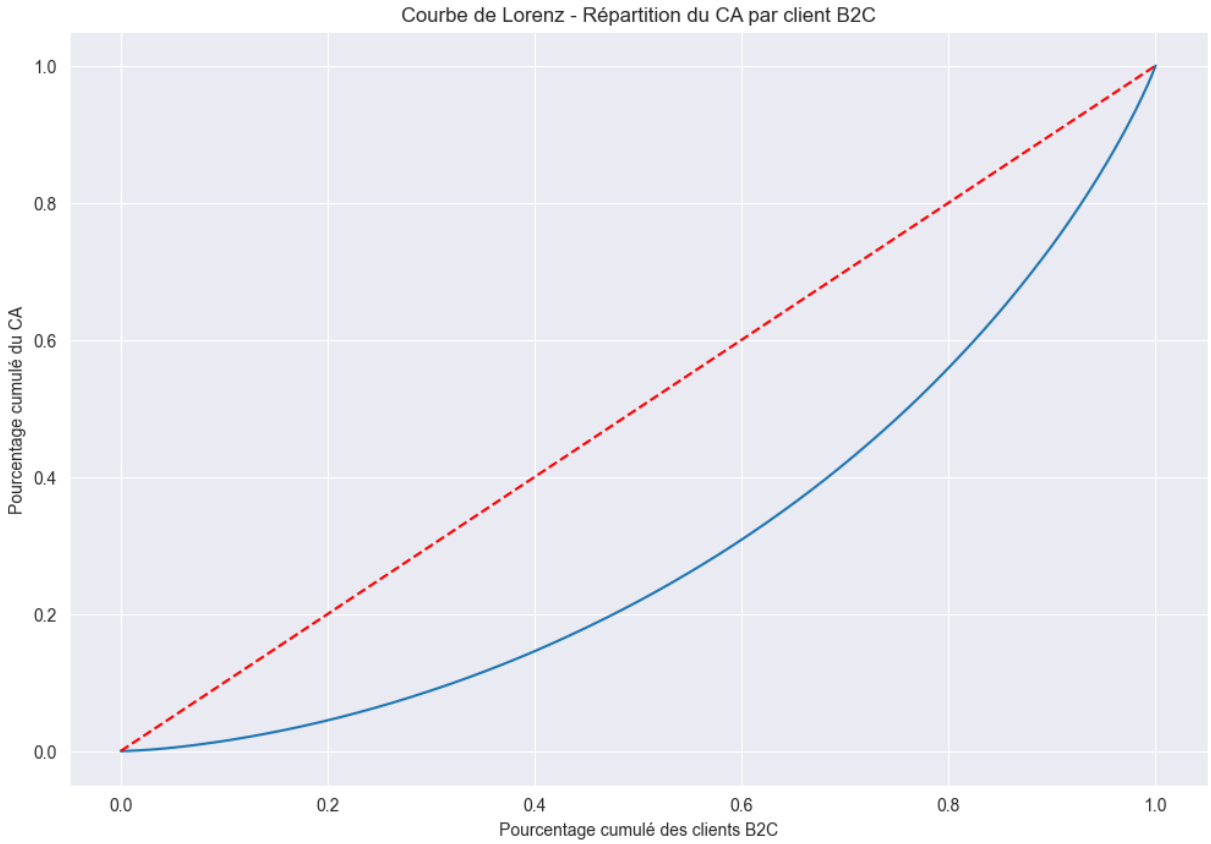


In [45]:
```
#Formule pour le calcul de l'indice de Gini
def gini(x):
    total = 0
    for i, xi in enumerate(x[:-1], 1):
        total += np.sum(np.abs(xi - x[i:]))
    return total / (len(x)**2 * np.mean(x))

#Indice de Gini pour la répartition du CA par client
gini_index = gini(total_sales_by_client['CA'].values)

print(f"Indice de Gini : {gini_index:.4f}")
```

```
Indice de Gini : 0.4419
```

In [46]:
```
# Même exploration en ne prenant en compte que les clients B2C
b2c_sales = total_sales_by_client[total_sales_by_client['client_type'] == 'B2C']
b2c_sales = b2c_sales.sort_values('CA')
b2c_sales['cum_percent_clients'] = np.arange(1, len(b2c_sales) + 1) / len(b2c_sales
b2c_sales['cum_percent_ca'] = b2c_sales['CA'].cumsum() / b2c_sales['CA'].sum()
```

```
plt.figure(figsize=(12, 8))
plt.plot([0] + list(b2c_sales['cum_percent_clients']), [0] + list(b2c_sales['cum_pe
plt.plot([0, 1], [0, 1], 'r--')
plt.xlabel('Pourcentage cumulé des clients B2C')
plt.ylabel('Pourcentage cumulé du CA')
plt.title('Courbe de Lorenz - Répartition du CA par client B2C')
plt.grid(True)
plt.show()
```



In [47]:
```
#Indice de Gini pour la répartition du CA par client B2C
gini_index = gini(b2c_sales['CA'].values)

print(f"Indice de Gini pour les clients B2C : {gini_index:.4f}")
```

```
Indice de Gini pour les clients B2C : 0.3983
```

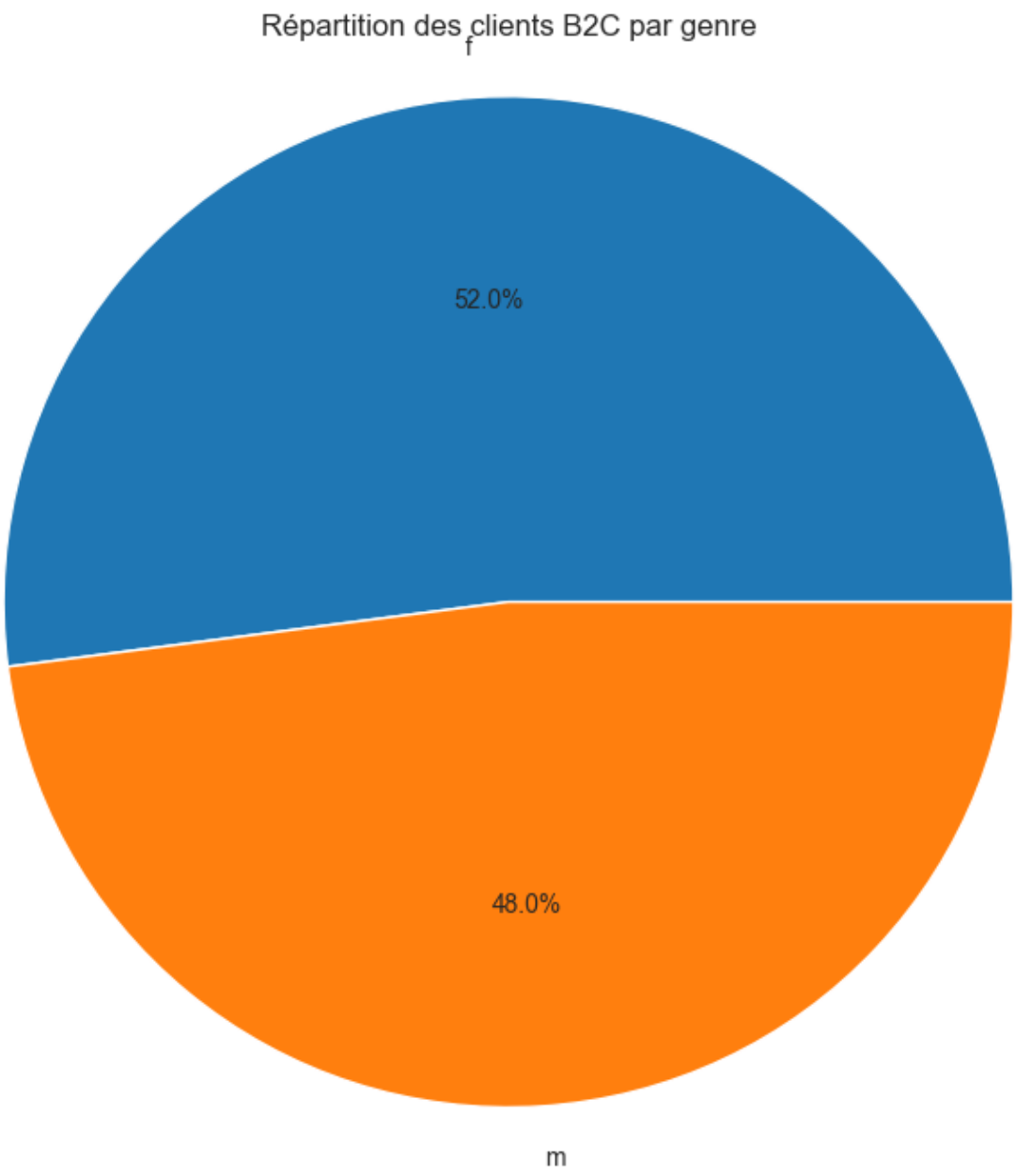
L'indice de Gini indique un niveau d'inégalité modéré dans la distribution des achats.

In [48]:
```
b2c_profiles = b2c_sales.merge(merged[['client_id', 'sex', 'birth']], on='client_id
b2c_profiles.head(10)
```

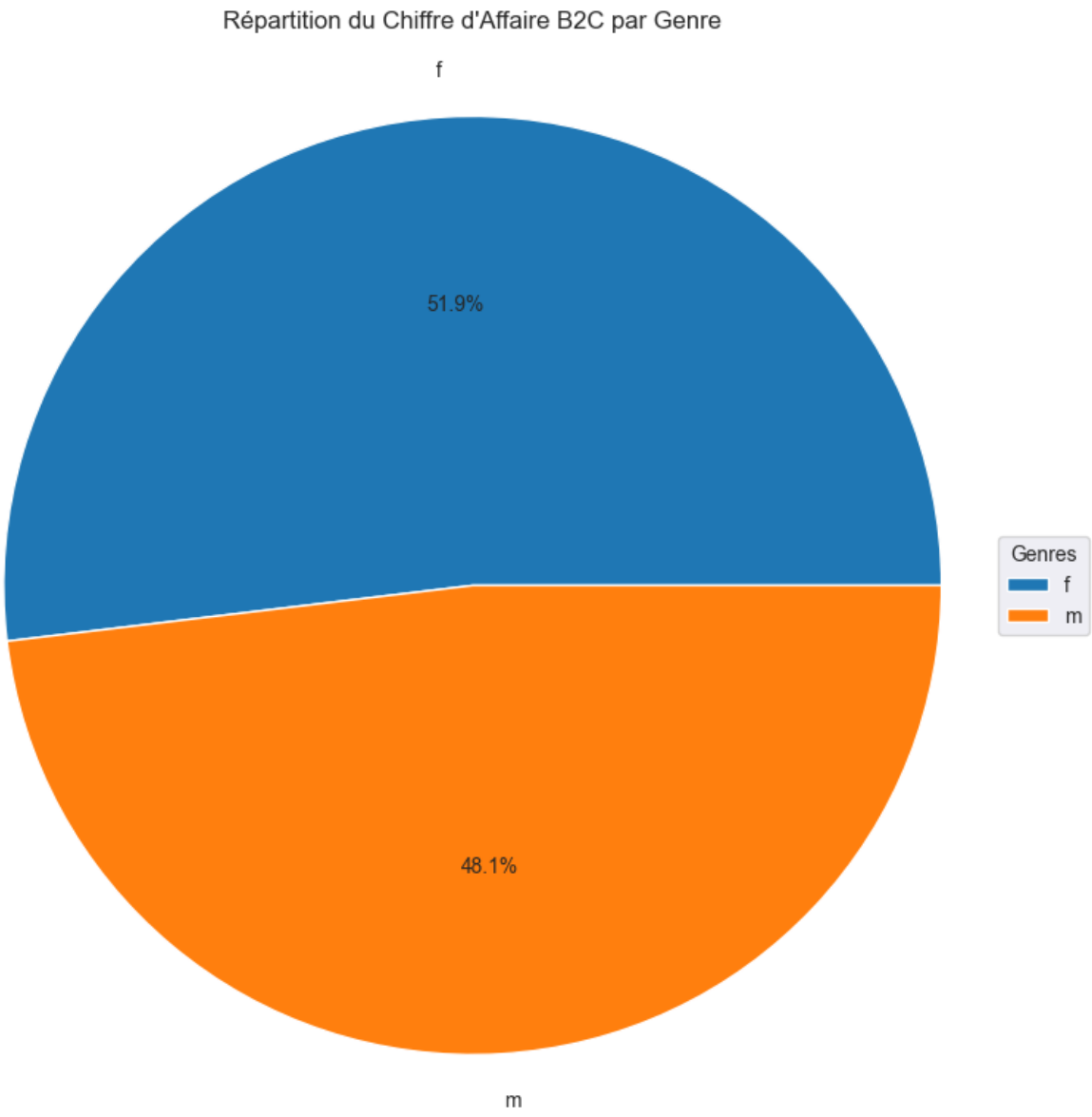
Out[48]:

	client_id	CA	client_type	cum_percent_clients	cum_percent_ca	sex	birth
0	c_8351	6.31	B2C	0.000116	5.662561e-07	f	1968
1	c_8140	8.30	B2C	0.000233	1.311094e-06	m	1971
2	c_8140	8.30	B2C	0.000233	1.311094e-06	m	1971
3	c_8114	9.98	B2C	0.000349	2.206694e-06	m	1962
4	c_8114	9.98	B2C	0.000349	2.206694e-06	m	1962
5	c_4648	11.20	B2C	0.000465	3.211776e-06	m	2004
6	c_4478	13.36	B2C	0.000582	4.410696e-06	f	1970
7	c_6040	15.72	B2C	0.000698	5.821400e-06	f	1974
8	c_6040	15.72	B2C	0.000698	5.821400e-06	f	1974
9	c_5919	15.98	B2C	0.000814	7.255437e-06	f	1955

In [49]:
```
# Répartition des clients par genre
gender_counts = b2c_profiles['sex'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(gender_counts.values, labels=gender_counts.index, autopct='%1.1f%%')
plt.title('Répartition des clients B2C par genre')
plt.axis('equal')
plt.show()
```



In [50]:
```
#Répartition du CA par genre
sales_by_gender = b2c_profiles.groupby('sex')['CA'].sum()

plt.figure(figsize=(8, 8))
plt.pie(sales_by_gender.values, labels=sales_by_gender.index, autopct='%1.1f%%')
plt.title('Répartition du Chiffre d\'Affaire B2C par Genre')
plt.legend(title="Genres", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))
plt.axis('equal')
plt.tight_layout()
plt.show()
```



Lien genre/catégorie

In [51]:
```
# Retrait des clients B2B du fichier merged
b2b_client_ids = total_sales_by_client.loc[total_sales_by_client['client_type'] ==
merged_b2c = merged.loc[~merged['client_id'].isin(b2b_client_ids)].copy()

# Ajout de la colonne âge
current_year = datetime.now().year
merged_b2c['age'] = current_year - merged_b2c['birth']
```

In [52]:
```
# Lien entre le genre du client et la catégorie de livre acheté
# 2 variables catégorielles, on va donc utiliser le test Khi2
from scipy.stats import chi2_contingency as chi2_contingency
contingency_table = pd.crosstab(merged_b2c['sex'], merged_b2c['categ'])
print(contingency_table)
```

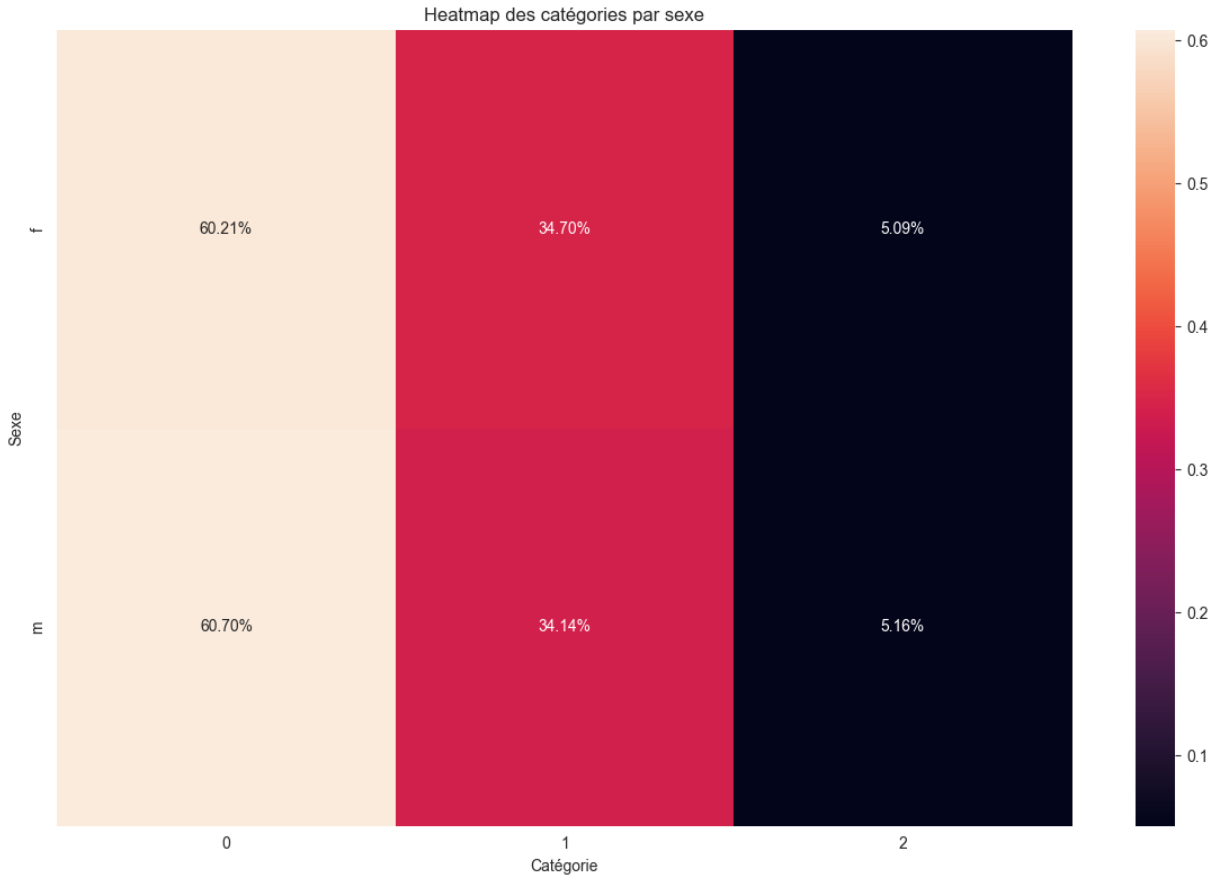
```
categ       0       1      2
sex
f      200793  115721  16980
m      186488  104884  15868
```

In [53]:
```
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

print(f"Statistique du chi-carré : {chi2}")
print(f"Degrés de liberte : {dof}")
print(f"Valeur p : {p_value}")
```

```
Statistique du chi-carré : 22.66856665178056
Degrés de liberte : 2
Valeur p : 1.1955928116587024e-05
```

La p value est nettement infé&rieure à 0.05, l'hypothèse nulle (pas de dépendance entre le genre du client et la catégorie achetée) est rejetée. On peut donc affirmer qu'il y un lien significatif entre le genre du client et la catégorie de livre acheté.

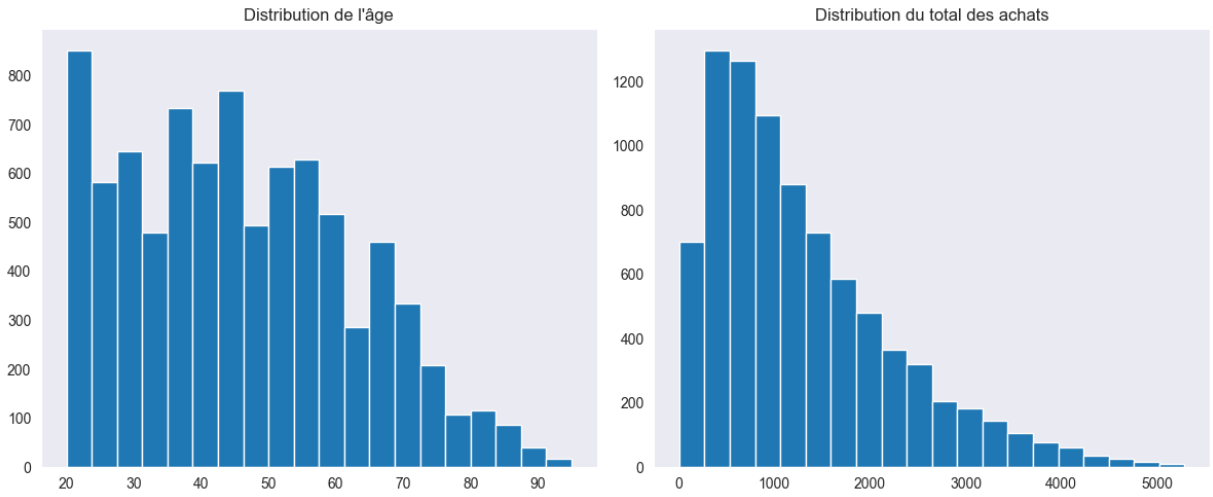
In [54]:
```
freq_table = pd.crosstab(merged_b2c['sex'], merged_b2c['categ'], normalize='index')
plt.figure(figsize=(12, 8))
sns.heatmap(freq_table, annot=True, fmt='.2%', )
plt.title(f'Heatmap des catégories par sexe')
plt.xlabel('Catégorie')
plt.ylabel('Sexe')
plt.tight_layout()
plt.show()
```



Lien âge/montant dépensé

In [55]:
```
# Nouveau df avec le montant total par client et l'âge du client
total_sales_age = merged_b2c.groupby('client_id').agg({'price': 'sum','age': 'first
total_sales_age.columns = ['client_id', 'total_achats', 'age']
```

In [56]:
```
# Vérification de la distribution des valeurs
plt.figure(figsize=(12, 5))
plt.subplot(121)
plt.hist(total_sales_age['age'], bins=20)
plt.title("Distribution de l'âge")
plt.subplot(122)
plt.hist(total_sales_age['total_achats'], bins=20)
plt.title("Distribution du total des achats")
plt.tight_layout()
plt.show()
```

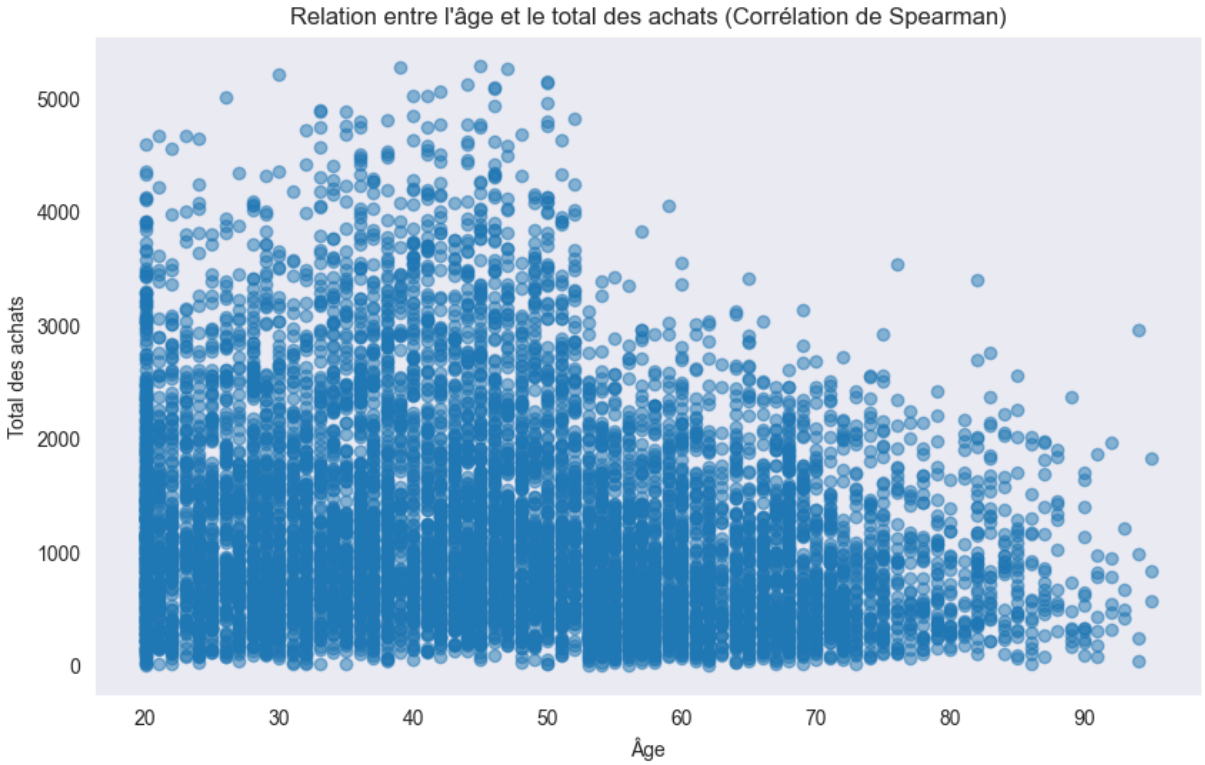


La distribution des valeurs ne suit pas la loi normale, on va donc partir sur un test de Spearman pour étudier le lien entre ces 2 variables.

In [57]:
```
# Test
corr, p_value = stats.spearmanr(total_sales_age['age'], total_sales_age['total_acha

print(f"Coefficient de corrélation de Spearman : {corr}")
print(f"p-value : {p_value}")
```

```
Coefficient de corrélation de Spearman : -0.18453804793783096
p-value : 1.0212910436382683e-66
```

In [58]:
```
# Visualisation
plt.figure(figsize=(10, 6))
plt.scatter(total_sales_age['age'], total_sales_age['total_achats'], alpha=0.5)
plt.xlabel("Âge")
plt.ylabel("Total des achats")
plt.title(f"Relation entre l'âge et le total des achats (Corrélation de Spearman)")
plt.show()
```



Le coefficient de Spearman de -0.18 indique une faible corrélation entre l'âge du client et le montant dépensé et que le montant dépensé diminuerait quand l'âge augmente. La pvalue faible nous montre que l'on peut rejeter l'hypothèse nulle (pas de lien), mais la corrélation est faible (-0.18).

Lien âge/fréquence d'achat

In [59]:
```
# Pearson ou Spearman selon distribution (nb achat/mois -> var continue)
# spearman
# corr, p_value = stats.spearmanr(x, y)
```

In [60]:
```
session_count = merged_b2c.groupby(['client_id', 'age', 'session_id', 'date_month']
frequence_age = session_count.groupby(['client_id', 'age']).size().reset_index(name
nb_month = session_count['date_month'].nunique()
frequence_age['monthly_buy_freq'] =  frequence_age['nb_sessions']/nb_month
frequence_age.head(5)
```

Out[60]:

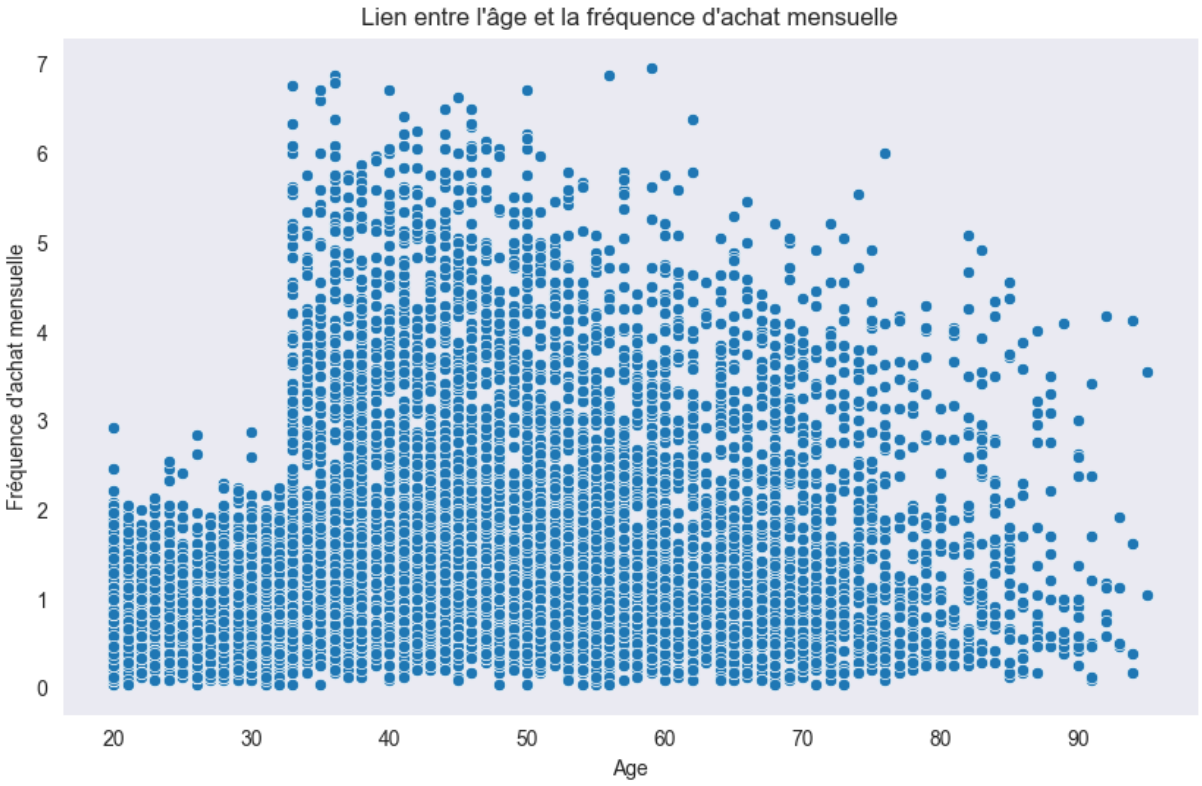
	client_id	age	nb_sessions	monthly_buy_freq
0	c_1	69	34	1.416667
1	c_10	68	34	1.416667
2	c_100	32	5	0.208333
3	c_1000	58	94	3.916667
4	c_1001	42	47	1.958333

In [61]:
```
corr, p_value = stats.spearmanr(frequence_age['age'], frequence_age['monthly_buy_fr
print(f"Coefficient de corrélation de Spearman : {corr}")
print(f"p-value : {p_value}")
```

```
Coefficient de corrélation de Spearman : 0.21193495184251068
p-value : 7.004159481412222e-88
```

La pvalue est extrémement faible, on peut rejeter avec certitude l'hypothèse H0 qui dit qui l'y a pas de relation entre l'âge et la fréquence d'achat. Le coefficient de corrélation est faible et positif, on peut dire quel'âge influe modérément de manière positive sur la fréquence d'achat, il existe certainement d'autres paramètres qui influencent cette fréquence.

In [62]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot( data=frequence_age, x='age', y='monthly_buy_freq',)
plt.title("Lien entre l'âge et la fréquence d'achat mensuelle")
plt.xlabel("Age")
plt.ylabel("Fréquence d'achat mensuelle")
plt.show()
```



Lien âge/panier moyen

In [63]:
```
# Pearson ou Spearman selon distribution (var continues)
#spearman
# corr, p_value = stats.spearmanr(x, y)
```

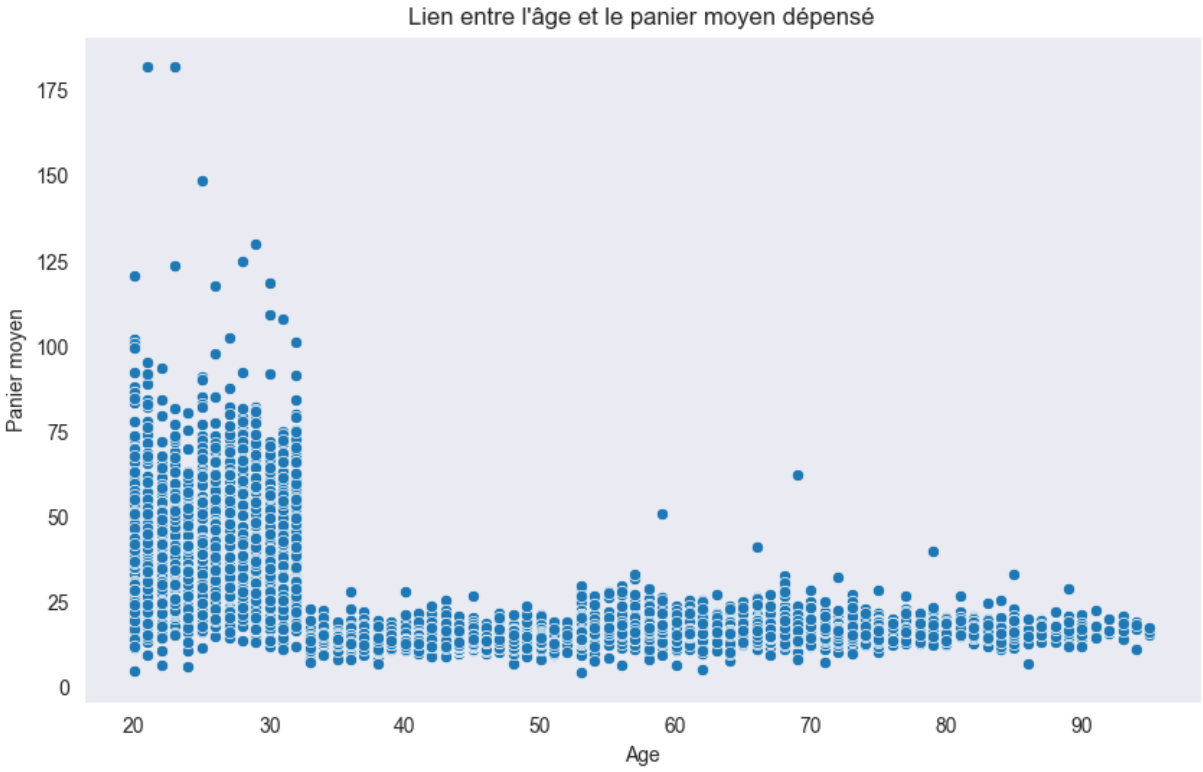
In [64]:
```
avg_price_age = merged_b2c.groupby(['client_id', 'age']).agg({'price': 'mean'}).res
corr, p_value = stats.spearmanr(avg_price_age['age'], avg_price_age['price'])
print(f"Coefficient de corrélation de Spearman : {corr}")
```

```
print(f"p-value : {p_value}")
```

```
Coefficient de corrélation de Spearman : -0.32586821258009735
p-value : 8.352869998649883e-212
```

L'hypothèse H0 est rejetée par la pvalue extrémement faible, l'hypothèse h1 qui dit qu'il existe un lien entre l'âge et la panier moyen dépensé est vraie. Le coefficient de corrélation est faible et négatif, ce qui signifie que l'âge a une influence modérée sur le panier moyen, et que plus l'âge avance plus le panier moyen diminue.

In [65]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot( data=avg_price_age, x='age', y='price',)
plt.title("Lien entre l'âge et le panier moyen dépensé")
plt.xlabel("Age")
plt.ylabel("Panier moyen")
plt.show()
```



Lien âge/catégorie achetée

In [66]:
```
# Anova ou Kruskal selon distribution (age var continue et categ var catégorielle)
# kruskal kruskal(*samples, nan_policy='propagate', axis=0, keepdims=False)
# vis boxplot
```

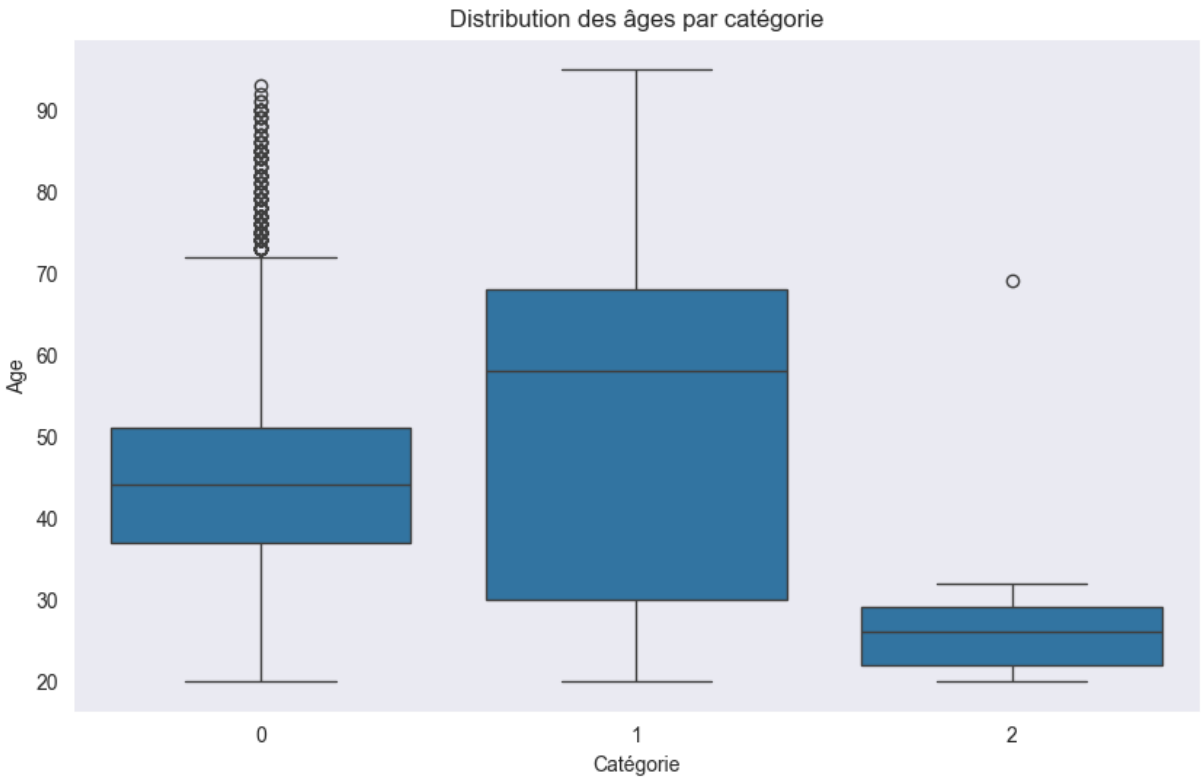
In [67]:
```
#On attribue la catégorie la plus achetée à chaque client avec lambda
age_categ = merged_b2c.groupby('client_id').agg({'age': 'first','categ': lambda x:

categ_1 = age_categ[age_categ['categ'] == 0]['age']
categ_2 = age_categ[age_categ['categ'] == 1]['age']
categ_3 = age_categ[age_categ['categ'] == 2]['age']
```

```
h_statistic, pvalue = stats.kruskal(categ_1, categ_2, categ_3)
print(f"Statistique H de Kruskal-Wallis : {h_statistic}")
print(f"p-value : {pvalue}")
```

```
Statistique H de Kruskal-Wallis : 2149.235687688523
p-value : 0.0
```

L'hypothèse nulle est que la médiane est la même pour tous les groupes, l'hypothèse alternative est que la médiane n'est pas égale. La pvalue semble être très proche de 0 car le résultat affiché est 0, on rejette donc l'hypthèse nulle. L'âge serait donc associé à la préférence d'achat des client de manière significative. Cependant ce test ne permet pas de déterminer la relation entre l'âge et la catégorie achetée.

In [68]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=age_categ, x='categ', y='age')
plt.title("Distribution des âges par catégorie")
plt.xlabel("Catégorie")
plt.ylabel("Age")
plt.show()
```



La visualisation permet d'avoir une idée plus claire quant à la relation entre l'âge et la catégorie, la catégorie 2 semble avoir plus de succès chez les plus jeunes, la catégorie 0 entre 40 et 50 ans, la catégorie 1 semble elle attirer une tranche d'âge assez large. On se rappelle que la catégorie 2 a une médiane de prix plus élevée, ce qui explique également la relation précédente entre l'âge et le panier moyen (plus jeune = PM plus élevé).

Genre/CA (exploration supplémentaire)

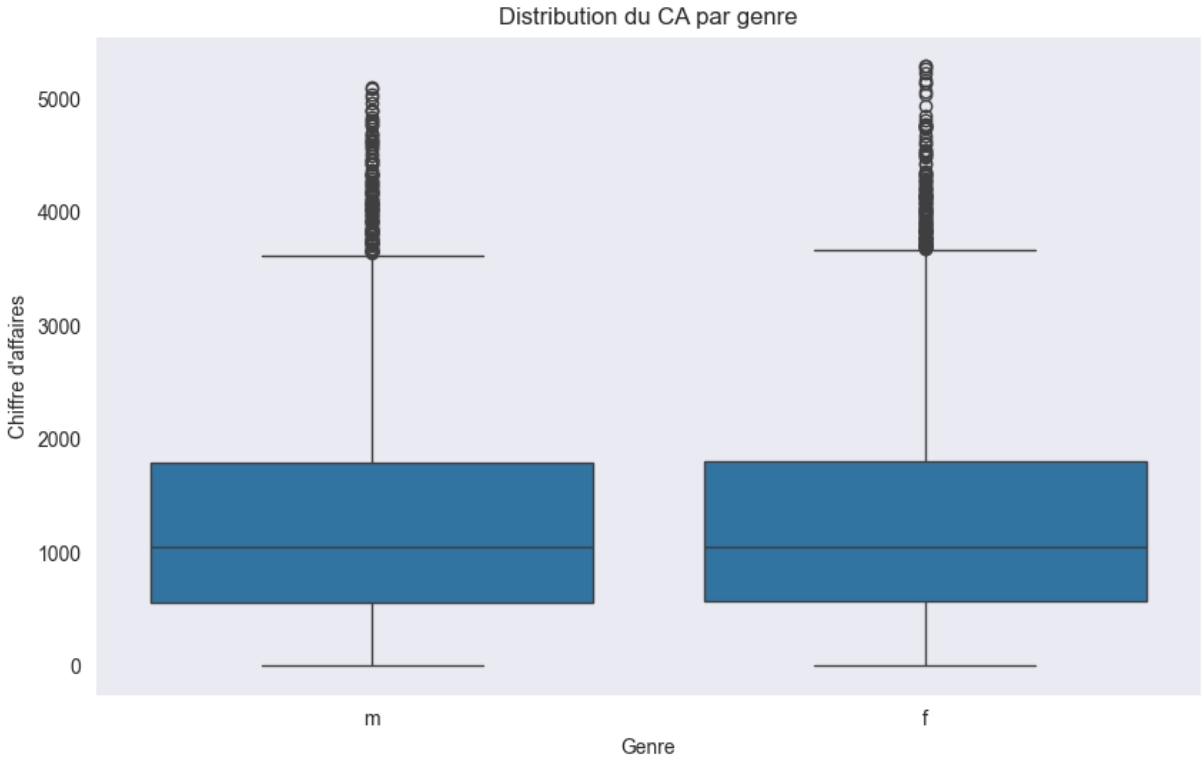
On va cette fois ci étudier le lien entre le genre et le montant dépensé. On mesure dans ce cas l'écart entre 2 moyennes, ce qui implique l'utlisation du test de Student.

In [69]:
```
#Calcul du CA par client et par genre
total_gender = merged_b2c.groupby(['client_id', 'sex'])['price'].sum().reset_index(
total_f= total_gender[total_gender['sex'] == 'f']['price']
total_m= total_gender[total_gender['sex'] == 'm']['price']

t_statistic, p_value = stats.ttest_ind(total_f, total_m)

print(f"t statistic : {t_statistic}")
print(f"p-value : {p_value}")
```

```
t statistic : -0.18246789566060223
p-value : 0.8552198508106684
```

H0: il n'y a pas de différence significative de CA entre les femmes et les hommes H1: il existe une différence significative de CA entre les femmes et les hommes La pvalue est nettement supérieure au seuil standard de 0.05, on ne rejette pas l'hypothèse nulle : pas de preuve statistique entre le genre et le montant dépensé. La valeur t proche de 0 suggère une différence très faible entre les moyennes des deux groupes, la visualisation suivante permet d'arriver au même constat sur la distribution du CA.

In [70]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=total_gender, x='sex', y='price')
plt.title("Distribution du CA par genre")
plt.xlabel("Genre")
plt.ylabel("Chiffre d'affaires")
plt.show()
```



Un lien avait été fait entre le genre et la catégorie achetée, cependant cela n'a pas d'impact sur la répartition du CA. Cela ne semble pas pertinent d'essayer de viser un genrage des articles.

Stratégies à envisager

Nous avons remarqué le lien entre l'âge et le panier moyen ainsi que le lien entre l'âge et la fréquence d'achat. Les catégories 0 et 1 semblent plébiscitées par la population plus âgée alors que la population plus jeune préfère la catégorie 2. Il serait interessant de tester une augmentation du prix moyen des articles de catégorie 0 et 1 dans le but d'augmenter le montant dépensé par la population de plus de 30 ans qui a une fréquence d'achat plus importante, et ainsi augmenter le chiffre d'affaire global. Il serait également intéressant de tester l'augmentation de l'offre des articles catégorie 2, qui représente une faible proportion des articles vendus mais qui a un prix médian plus élevé et qui touche une clientèle qui dépense plus.