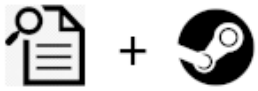
# Progetto



Denise Cilia
X81000791
Social Media Management
A.A 21/22

---

# Introduzione

In uno studio di profilazione condotto sulla piattaforma di Steam su un profilo utente in particolare, è stato costruito un dataset contenente 297 record inerenti ai giochi posseduti dall'utente. Troviamo al suo interno informazioni come:

- l'**id** univoco del gioco;
- il **nome** del gioco;
- le **ore di gioco**;
- se è stato giocato nelle ultime **due settimane**;
- ed infine il **genere** del gioco.

Su questo dataset verrà applicata una fase di filtering in maniera tale da concentrarci solamente sui giochi effettivamente giocati dall'utente evitando così un'elaborazione troppo ampia.
In un secondo momento verranno aggiunte al dataset in studio le colonne **review** e **label**, contenenti rispettivamente tutte le recensioni ottenute attraverso scraping della pagina web del gioco in particolare, e le varie etichette ad indicare se si tratta di una recensione positiva o negativa.

Lo scopo di questo progetto è costruire un piccolo sistema di raccomandazione per l'utente. Questo lavoro verrà svolto ottenendo le recensioni per un determinato gioco, ad esempio il gioco più giocato dall'utente, e procedendo con una rappresentazione delle recensioni attraverso il modello **Bag of Words**. Successivamente si ricerca sullo Steam Store un gioco dello stesso genere del gioco selezionato, e allo stesso modo si ricavano dalla pagina web tutte le recensioni disponibili che vengono rappresentate mediante lo stesso modello sopracitato. Il risultato di entrambe le elaborazioni sarà contenuto in due vettori separati sulla quale verrà applicato il calcolo della **similarità del Coseno**, che darà in output un risultato mediante il quale sarà possibile consigliare o meno all'utente il gioco in questione.

---

Inizialmente importiamo il dataset ed effettuiamo delle semplici operazioni di filtering.

In [2]:
```
import pandas as pd

data = pd.read_csv('./games.csv')
data
```

Out[2]:

| | id | name | hours_of_game | recently_played | genres |
|---|---|---|---|---|---|
| **0** | 20 | Team Fortress Classic | 0 | no | Action |
| **1** | 15710 | Oddworld: Abe's Exoddus | 0 | no | Adventure |
| **2** | 9480 | Saints Row 2 | 0 | no | Action |
| **3** | 1250 | Killing Floor | 1 | no | Action |
| **4** | 35420 | Killing Floor Mod: Defence Alliance 2 | 0 | no | Action |
| **...** | ... | ... | ... | ... | ... |
| **291** | 46510 | Syberia 2 | 0 | no | Adventure |
| **292** | 204450 | Call of Juarez Gunslinger | 0 | no | Action |
| **293** | 520720 | Dear Esther: Landmark Edition | 0 | no | Adventure Casual Indie |
| **294** | 1577120 | The Quarry | 0 | no | Adventure |
| **295** | 1515950 | Capcom Arcade Stadium | 0 | no | Action Free to Play |

296 rows × 5 columns

In [3]:

```
new_data = data[data['hours_of_game'] >= 20] # filter

new_data.index = range(len(new_data)) # re-impostiamo gli indici delle righ
new_data
```

Out[3]:

| | id | name | hours_of_game | recently_played | genres |
|---|---|---|---|---|---|
| **0** | 105600 | Terraria | 623 | no | Action Adventure Indie RPG |
| **1** | 113200 | The Binding of Isaac | 99 | yes | Action Adventure Indie RPG |
| **2** | 205100 | Dishonored | 21 | no | Action Adventure |
| **3** | 221040 | Resident Evil 6 | 62 | no | Action Adventure |
| **4** | 222480 | Resident Evil Revelations | 109 | no | Action Adventure |
| **5** | 211820 | Starbound | 79 | no | Action Adventure Casual Indie RPG |
| **6** | 235460 | METAL GEAR RISING: REVENGEANCE | 20 | no | Action |
| **7** | 254700 | Resident Evil 4 | 38 | no | Action Adventure |
| **8** | 335300 | DARK SOULS™ II: Scholar of the First Sin | 74 | no | Action RPG |
| **9** | 250900 | The Binding of Isaac: Rebirth | 392 | yes | Action |
| **10** | 287290 | Resident Evil Revelations 2 | 71 | no | Action Adventure |
| **11** | 21690 | Resident Evil 5 | 30 | no | Action Adventure |
| **12** | 367520 | Hollow Knight | 109 | no | Action Adventure Indie |
| **13** | 322330 | Don't Starve Together | 942 | yes | Adventure Indie Simulation |
| **14** | 374320 | DARK SOULS™ III | 78 | no | Action |
| **15** | 389730 | TEKKEN 7 | 37 | no | Action Sports |
| **16** | 236110 | Dungeon Defenders II | 131 | no | Action Free to Play Indie RPG Strategy |
| **17** | 413150 | Stardew Valley | 114 | no | Indie RPG Simulation |
| **18** | 339340 | Resident Evil 0 | 38 | no | Action Adventure |
| **19** | 403640 | Dishonored 2 | 38 | no | Action |
| **20** | 292030 | The Witcher 3: Wild Hunt | 171 | no | RPG |
| **21** | 418370 | Resident Evil 7 Biohazard | 50 | yes | Action Adventure |
| **22** | 601510 | Yu-Gi-Oh! Duel Links | 96 | no | Free to Play Strategy |
| **23** | 614570 | Dishonored®: Death of the Outsider™ | 23 | no | Action |
| **24** | 692850 | Bloodstained: Ritual of the Night | 23 | no | Action Adventure RPG |
| **25** | 774361 | Blasphemous | 28 | no | Action Adventure Indie |
| **26** | 731490 | Crash Bandicoot™ N. Sane Trilogy | 27 | no | Action |
| **27** | 973760 | Thronebreaker: The Witcher Tales | 25 | yes | Adventure RPG |
| **28** | 883710 | Resident Evil 2 | 62 | yes | Action |
| **29** | 1147560 | Skul: The Hero Slayer | 50 | yes | Action Indie |

| | id | name | hours_of_game | recently_played | genres |
|---|---|---|---|---|---|
| **30** | 1201240 | BLEACH Brave Souls - 3D Action | 636 | yes | Action Casual Free to Play RPG |
| **31** | 1145360 | Hades | 82 | no | Action Indie RPG |
| **32** | 1449850 | Yu-Gi-Oh! Master Duel | 113 | yes | Free to Play Simulation Strategy |
| **33** | 1325200 | Nioh 2 – The Complete Edition | 135 | no | Action RPG |
| **34** | 1196590 | Resident Evil Village | 54 | no | Action |

Passiamo adesso all'operazione di web scraping che ci permetterà di ottenere le informazioni ricercate, ovvero recensione e tipo di recensione: se positiva o negativa.

In [4]:
```
import requests

def get_reviews(appid, params = {'json':1}):
        url = 'https://store.steampowered.com/appreviews/' + appid
        response = requests.get(url = url, params = params, headers = {'Use

        return response.json()
```

In [5]:
```
def get_n_reviews(appid, n = 100):
    reviews = []
    cursor = '*'
    params = {
            'json' : 1,
            'filter' : 'all',
            'language' : 'english',
            'day_range' : 9223372036854775807,
            'review_type' : 'all',
            'purchase_type' : 'all'
            }

    while n > 0:
        params['cursor'] = cursor.encode()
        params['num_per_page'] = min(100, n)
        n -= 100

        response = get_reviews(appid, params)
        cursor = response['cursor']
        reviews += response['reviews']

    return reviews
```

La funzione `get_n_reviews` (1) dà in output *n* recensioni su un determinato gioco descritto dal suo **id** univoco.
Adesso definiamo una serie di funzioni utili alla pulizia del testo. In particolare rimuoveremo dal testo di ciascuna delle recensioni: i tag del tipo **[tag]**, le emoji, le end line **\n** e i numeri.

In [6]:
```
import re

def removeSquareBracketsTag(data):
    pattern = r'\[.*?\]'
    return re.sub(pattern, '', data)

def remove_emojis(data):
    emoj = re.compile("["
```

```
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)

def removeEndLines(data):
    return re.sub(r'\n', '', data)

def remove_number(data):
    data = data.lower()
    data = re.sub(r'\d+', '', data)
    return data
```

In [8]:

```
def addToDataset(data_param):
    reviews = []
    dictListForReview = []
    dictListForLabels = []

    appids = data_param['id'].to_numpy()
    for i in range(len(appids)):
        reviews = get_n_reviews(str(appids[i]), 100)
        df = pd.DataFrame(reviews)[['review', 'voted_up']]

        for j in range(len(df['review'])):
            df.at[j, 'review'] = remove_emojis(df.iloc[j]['review'])
            df.at[j, 'review'] = removeSquareBracketsTag(df.iloc[j]['review
            df.at[j, 'review'] = removeEndLines(df.iloc[j]['review'])
            df.at[j, 'review'] = remove_number(df.iloc[j]['review'])

        label_dict = df['voted_up'].to_dict()
        review_dict = df['review'].to_dict()
        dictListForReview.append(review_dict)
        dictListForLabels.append(label_dict)
    copy = data_param
    copy['review'] = dictListForReview
    copy['label'] = dictListForLabels

    return copy
```

Definendo la funzione `addToDataset` prendiamo tutti gli id nel dataset ed estraiamo per ciascun record presente tutte le recensioni e le etichette del gioco in questione, aggiungendo queste nuove informazioni all'interno di due nuove colonne **review** e **label** che si troveranno in un nuovo dataframe di copia.
Più formalmente prendendo in considerazione la riga *i*-esima del nuovo dataset, avremo in output nella colonna review una lista di recensioni e una lista di etichette nella colonna label

con una corrispondenza 1-a-1. Nella *j*-esima posizione faremo riferimento alla *j*-esima etichetta

In [9]:
```
new_data_2 = addToDataset(new_data)
```

In [10]:
```
new_data_2
```

Out[10]:

| | id | name | hours_of_game | recently_played | genres | review | la |
|---|---|---|---|---|---|---|---|
| **0** | 105600 | Terraria | 623 | no | Action Adventure Indie RPG | {0: 'terraria’s final major update, journey’s ... | Tı Tı Tı Tı Tı |
| **1** | 113200 | The Binding of Isaac | 99 | yes | Action Adventure Indie RPG | {0: 'this is definitely one of the greatrest g... | Tı Tı Tı Tı Tı |
| **2** | 205100 | Dishonored | 21 | no | Action Adventure | {0: '/pros you really fall in love with th... | Tı Tı Tı Tı Tı |
| **3** | 221040 | Resident Evil 6 | 62 | no | Action Adventure | {0: 'leon. this is chris. i have an urgent mis... | Tı Tı Tı Tı Tı |
| **4** | 222480 | Resident Evil Revelations | 109 | no | Action Adventure | {0: 'resident evil revelations is capcom's att... | Tı Tı Tı Tı Tı |
| **5** | 211820 | Starbound | 79 | no | Action Adventure Casual Indie RPG | {0: 'this review is for version . starbound is... | Tı Tı Tı Tı Tı |

| | id | name | hours_of_game | recently_played | genres | review | la |
|---|---|---|---|---|---|---|---|
| **6** | 235460 | METAL GEAR RISING: REVENGEANCE | 20 | no | Action | {0: 'senator armstrong is so cool, i wish amer... | Tr<br>Tr<br>Tr<br>Tr<br>Tr |
| **7** | 254700 | Resident Evil 4 | 38 | no | Action Adventure | {0: 'don't worry you are not only one who play... | Tr<br>Tr<br>Tr<br>Tr<br>Tr |
| **8** | 335300 | DARK SOULS™ II: Scholar of the First Sin | 74 | no | Action RPG | {0: 'tl;drbuy it if you prefer the individual ... | Tr<br>Tr<br>Tr<br>Tr<br>Tr |
| **9** | 250900 | The Binding of Isaac: Rebirth | 392 | yes | Action | {0: 'rebirth is an amazing game! it improved o... | Tr<br>Tr<br>Tr<br>Tr<br>Tr |
| **10** | 287290 | Resident Evil Revelations 2 | 71 | no | Action Adventure | {0: ' beware, if you're buying this game to pl... | Tr<br>Tr<br>Tr<br>Tr<br>Tr |
| **11** | 21690 | Resident Evil 5 | 30 | no | Action Adventure | {0: 'if you have problems opening the game:) o... | Tr<br>Tr<br>Tr<br>Fa<br>Fals |

| | id | name | hours_of_game | recently_played | genres | review | la |
|---|---|---|---|---|---|---|---|
| **12** | 367520 | Hollow Knight | 109 | no | Action Adventure Indie | {0: 'loved it so much i showed my wife.she lov... | Tı Tı Tı Tı Tı |
| **13** | 322330 | Don't Starve Together | 942 | yes | Adventure Indie Simulation | {0: 'awesome game.my ex gave me this game. we ... | Tı Tı Tı Tı Tı |
| **14** | 374320 | DARK SOULS™ III | 78 | no | Action | {0: 'i mean, what's there to say. it's dark so... | Tı Tı Tı Tı Tı |
| **15** | 389730 | TEKKEN 7 | 37 | no | Action Sports | {0: 'i can't explain this without getting emot... | Tı Tı Tı Tı Tı |
| **16** | 236110 | Dungeon Defenders II | 131 | no | Action Free to Play Indie RPG Strategy | {0: 'purchase a gem pack to expand bags, recei... | Fa Tı Fa Tı Fa |
| **17** | 413150 | Stardew Valley | 114 | no | Indie RPG Simulation | {0: 'fun and relaxing game. always something t... | Tı Tı Tı Tı Tı |

| | id | name | hours_of_game | recently_played | genres | review | la |
|---|---|---|---|---|---|---|---|
| **18** | 339340 | Resident Evil 0 | 38 | no | Action Adventure | {0: 'resident evil was in development hell si... | Tı Tı Tı Tı Tı |
| **19** | 403640 | Dishonored 2 | 38 | no | Action | {0: 'improves on every single aspect of the fi... | Tı Tı Tı Tı Tı |
| **20** | 292030 | The Witcher 3: Wild Hunt | 171 | no | RPG | {0: 'imagine you're working in a - corporate j... | Tı Tı Tı Tı Tı |
| **21** | 418370 | Resident Evil 7 Biohazard | 50 | yes | Action Adventure | {0: 'pros:-good balance between dangerous area... | Tı Tı Tı Tı Tı |
| **22** | 601510 | Yu-Gi-Oh! Duel Links | 96 | no | Free to Play Strategy | {0: 'and remember, the most powerful card is y... | Tı Tı Tı Tı Tı |
| **23** | 614570 | Dishonored®: Death of the Outsider™ | 23 | no | Action | {0: 'dishonored one more timedishonored: death... | Tı Tı Tı Tı Tı |

| | id | name | hours_of_game | recently_played | genres | review | la |
|---|---|---|---|---|---|---|---|
| **24** | 692850 | Bloodstained: Ritual of the Night | 23 | no | Action Adventure RPG | {0: 'did you like sotn? or aria/dawn of sorrow... | Tı Tı Tı Tı Tı |
| **25** | 774361 | Blasphemous | 28 | no | Action Adventure Indie | {0: 'an important thing when approaching blasp... | Tı Tı Tı Tı Tı |
| **26** | 731490 | Crash Bandicoot™ N. Sane Trilogy | 27 | no | Action | {0: 'i was when i first played crash bandicoo... | Tı Tı Tı Tı Tı |
| **27** | 973760 | Thronebreaker: The Witcher Tales | 25 | yes | Adventure RPG | {0: 'i like to think of this game as what a te... | Tı Tı Tı Tı Tı |
| **28** | 883710 | Resident Evil 2 | 62 | yes | Action | {0: 'resident evil that we deserve. capcom. i ... | Tı Tı Tı Tı Tı |
| **29** | 1147560 | Skul: The Hero Slayer | 50 | yes | Action Indie | {0: 'this is my opinion on this game. i havent... | Tı Tı Tı Tı Tı |

| | id | name | hours_of_game | recently_played | genres | review | la |
|---|---|---|---|---|---|---|---|
| **30** | 1201240 | BLEACH Brave Souls - 3D Action | 636 | yes | Action Casual Free to Play RPG | {0: 'bankai deez nuts', 1: 'as a disclaimer, t... | Tı |
| **31** | 1145360 | Hades | 82 | no | Action Indie RPG | {0: 'i don't usually play action rogue-likes..... | Tı |
| **32** | 1449850 | Yu-Gi-Oh! Master Duel | 113 | yes | Free to Play Simulation Strategy | {0: 'how master duel *play ranked*opponent is... | Tı |
| **33** | 1325200 | Nioh 2 – The Complete Edition | 135 | no | Action RPG | {0: 'you can return to character customization... | Tı |
| **34** | 1196590 | Resident Evil Village | 54 | no | Action | {0: 'warning, this review will contain spoiler... | Tı |

Concentriamo il nostro studio sui giochi più recentemente giocati dall'utente nelle ultime due settimane e con le più alte ore di gioco, così da ottenere:

In [12]:
```
mostPlayed = new_data_2[(new_data_2['hours_of_game'] >= 600) & (new_data_2[
mostPlayed
```

Out[12]:

| | id | name | hours_of_game | recently_played | genres | review | label |
|---|---|---|---|---|---|---|---|
| **13** | 322330 | Don't Starve Together | 942 | yes | Adventure Indie Simulation | {0: 'awesome game.my ex gave me this game. we ... | {0: True, 1: True, 2: True, 3: True, 4: True, ... |
| **30** | 1201240 | BLEACH Brave Souls - 3D Action | 636 | yes | Action Casual Free to Play RPG | {0: 'bankai deez nuts', 1: 'as a disclaimer, t... | {0: True, 1: True, 2: True, 3: True, 4: True, ... |

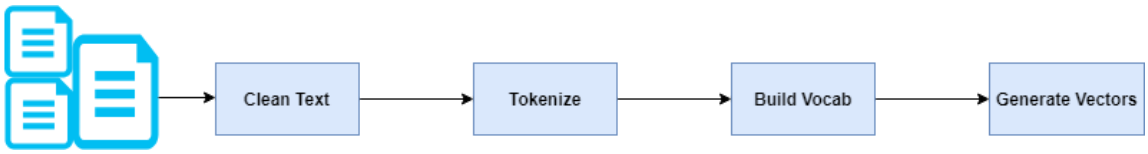
In [13]:
```
firstGameReview = mostPlayed.iloc[0]['review']
```

In [15]:
```
docs = []
for i in range(len(firstGameReview)):
    docs.append((firstGameReview[i]))
```

Estraiamo le recensioni del primo gioco e utilizziamo un ciclo *for* per inserire ciascuna recensione come singolo elemento in un array di stringhe. Quindi trasformiamo il dizionario in un array. Questa operazione ci servirà per poter usufruire del metodo `CountVectorizer`.

In [31]:
```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words = 'english')
cv_fit = cv.fit_transform(docs)
wordList = cv.get_feature_names()
countList = cv_fit.toarray().sum(axis=0)
result = dict(zip(wordList, countList))
```




Il metodo `CountVectorizer` viene utilizzato per convertire una collezione di documenti di testo in un vettore conteggio di termini o token. È possibile effettuare un ulteriore pre-processing sui dati, il quale permette di eliminare la punteggiatura e altre cosiddette **stop words**: parole non significative per il processo di **NLP** (Natural Language Processing). Visualizziamo il risultato:

In [34]:
```
unique_string = (" ").join(wordList)

import matplotlib.pyplot as plt
from wordcloud import WordCloud

word_cloud = WordCloud(collocations = False, background_color = 'white').ger

plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


Adesso utilizziamo un'ulteriore tecnica di web scraping([2]) per poter usufruire della funzione **ricerca** nello Store ufficiale di Steam. Tramite questo riusciremo ad individuare un numero *n* di giochi con il loro id univoco e specificandone il genere.

In [36]:
```
from bs4 import BeautifulSoup

headers = {'User-Agent':'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
def searchStore(n, term):
    game = []
    id = []
    while len(game) < n:
        url = f'https://store.steampowered.com/search/?term={term}&category1
        html = requests.get(url, headers=headers).text
        soup = BeautifulSoup(html, 'html.parser')
        search_results = soup.find_all(class_='search_result_row')

        if not search_results:
            break

        title = [result.find('span', class_='title').text for result in sea
        appid = [result['data-ds-appid'] for result in search_results]
        game.extend(title)
        id.extend(appid)
    num = min(n, len(game))
    return pd.DataFrame({
        'game':game[:num],
        'id':id[:num]
    })
```

In [39]:
```
suggested = searchStore(15, term = 'Adventure+Indie+Simulation')
suggested
```

Out[39]:

| | game | id |
|---|---|---|
| 0 | The Mortuary Assistant | 1295920 |
| 1 | BONELAB | 1592190 |
| 2 | Noita | 881100 |
| 3 | Escape Simulator | 1435790 |
| 4 | Clanfolk | 1700870 |
| 5 | MADiSON | 1670870 |
| 6 | Thief Simulator | 704850 |
| 7 | Totally Accurate Battle Simulator | 508440 |
| 8 | This War of Mine | 282070 |
| 9 | House Party | 611790 |
| 10 | BONEWORKS | 823500 |
| 11 | Keplerth | 747200 |
| 12 | Mr. Prepper | 761830 |
| 13 | Ancestors: The Humankind Odyssey | 536270 |
| 14 | CUSTOM ORDER MAID 3D2 It's a Night Magic | 1097580 |

Estraiamo in maniera randomica solo due campioni dal dataframe **suggested** per rendere l'elaborazione più snella e veloce.

In [43]:
```
random_suggest = suggested.sample(n = 2)
random_suggest
```

Out[43]:

| | game | id |
|---|---|---|
| 4 | Clanfolk | 1700870 |
| 13 | Ancestors: The Humankind Odyssey | 536270 |

In [44]:
```
new_suggested = addToDataset(random_suggest)
```

Andremo ad aggiungere anche qui recensioni ed etichette.

In [45]:
```
new_suggested
```

Out[45]:

| | game | id | review | label |
|---|---|---|---|---|
| 4 | Clanfolk | 1700870 | {0: 'several people have asked me to edit my r... | {0: False, 1: True, 2: True, 3: True, 4: True,... |
| 13 | Ancestors: The Humankind Odyssey | 536270 | {0: 'this is a very hesitant thumbs up, and ev... | {0: True, 1: True, 2: True, 3: True, 4: True, ... |

In [46]:
```
firstGameReview = new_suggested.iloc[0]['review']

docs2 = []
for i in range(len(firstGameReview)):
    docs2.append((firstGameReview[i]))

cv2 = CountVectorizer(stop_words = 'english')
cv2_fit = cv2.fit_transform(docs2)
wordList2 = cv2.get_feature_names()
```

```
countList2 = cv2_fit.toarray().sum(axis=0)
result2 = dict(zip(wordList2, countList2))
```

In [48]:
```
unique_string2 = (" ").join(wordList2)

word_cloud = WordCloud(collocations = False, background_color = 'white').ger

plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


Calcoliamo infinite la **similarità del Coseno**. La similarità del coseno è una tecnica euristica per la misurazione della similitudine tra due vettori effettuata calcolando il coseno tra di loro, usata generalmente per il confronto di testi nell'estrazione di dati e nell'analisi testuale.
Nel caso dell'analisi dei testi, poiché le frequenze dei termini sono sempre valori positivi, si otterranno valori che vanno da 0 a +1, dove +1 indica che le parole contenute nei due testi sono le stesse (ma non necessariamente nello stesso ordine) e 0 che non c'è nessuna parola che appare in entrambi.([3])

In [61]:
```
from collections import Counter
from sklearn.metrics.pairwise import cosine_similarity
from decimal import Decimal

def cosineSimilarity(par1, par2):


    # count word occurrences
    a_vals = Counter(par1)
    b_vals = Counter(par2)

    # convert to word-vectors
    words  = list(a_vals.keys() | b_vals.keys())
    a_vect = [a_vals.get(word, 0) for word in words]
    b_vect = [b_vals.get(word, 0) for word in words]

    # find cosine
    len_a  = sum(av*av for av in a_vect) ** 0.5
    len_b  = sum(bv*bv for bv in b_vect) ** 0.5
    dot    = sum(av*bv for av,bv in zip(a_vect, b_vect))
    cosine = dot / (len_a * len_b)

    print(round(Decimal(cosine), 2))
```

In [62]:
```
cosineSimilarity(wordList, wordList2) # Don't Starve Together / Clanfolk
```

```
0.40
```

In questo caso la somiglianza è del **40%**, il gioco potrebbe essere consigliato all'utente, ma l'utente potrebbe non gradirlo. Proviamo con il secondo gioco nella lista:

In [56]:
```
secondGameReview = new_suggested.iloc[1]['review']

docs3 = []
for i in range(len(secondGameReview)):
    docs3.append((secondGameReview[i]))

cv3 = CountVectorizer(stop_words = 'english')
cv3_fit = cv3.fit_transform(docs3)
wordList3 = cv3.get_feature_names()
countList3 = cv3_fit.toarray().sum(axis=0)
result3 = dict(zip(wordList3, countList3))
```

In [58]:
```
unique_string3 = (" ").join(wordList3)

word_cloud = WordCloud(collocations = False, background_color = 'white').ger

plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


In [63]:
```
cosineSimilarity(wordList, wordList3) # Don't Starve Together / Ancestors:
```

```
0.41
```

Anche qui otteniamo un grado di accuratezza del **41%**. Anche questo gioco potrebbe essere consigliato all'utente ma potrebbe non apprezzarlo.

# Conclusioni

Lo studio porta alla formulazione di un sistema di raccomandazione molto semplice che permette di suggerire dei titoli, ma difficilmente permette di ottenere risultati sopra il **50%** di accuratezza in quanto il sistema di recensione è fortemente legato al gioco che viene studiato. Ad esempio per il titolo *Don't Starve Together* appaiono frequentemente nei termini i nomi dei protagonisti del gioco, i quali quasi certamente non si troveranno nelle recensioni di altri titoli. Ogni gioco viene recensito da ciascun utente in maniera diversa e in molti casi le recensioni non contengono abbastanza parole. Bisognerebbe dunque, al fine di ottenere risultati più concreti, ampliare le recensioni raccolte, così da lavorare su più dati, migliorare il sistema di filtering, così da tener conto delle sole parole rappresentative per un determinato gioco ed infine applicare la similarità del coseno su tali risultati.

---

Fonti:

1. https://andrew-muller.medium.com/scraping-steam-user-reviews-9a43f9e38c92
2. https://pypi.org/project/steam-review-scraper/#description (search_game_id, leggermente modificato ed adattato allo studio in questione)
3. https://it.wikipedia.org/wiki/Coseno_di_similitudine#:~:text=La%20similarit%C3%A0%20del