TRABAJO PRACTICO FINAL

Coderhouse 7/4/2022

Integrantes:Mariano Medela, Francisco del Pozo

STEAM DATA ANALYTICS

*Introducción*

En el presente informe se intentará demostrar los distintos factores que pueden llegar a formar parte de la razón por la cual un juego termina siendo o no exitoso. Esto incluye precios, ofertas y opiniones de usuarios.

El mismo fue desarrollado por Francisco del Pozo y Mariano Tomas Medela bajo la tutoría de Santiago Majorani.

Se implementará un tablero de indicadores modelando la cantidad de jugadores en los juegos más populares, sus precios y el nivel de satisfacción de los usuarios de este producto. Otros datos que se harán presentes son la cantidad de likes, dislikes, máxima cantidad de personas jugando en un mismo momento y como el uso de promociones especiales en determinados momentos clave permitió aumentar la popularidad de un juego o evitar su caída, dependiendo del caso.

*Objetivo*

El objetivo del mismo es poder ver que impacto tuvieron ciertas estrategias económicas y que compañías pueden llegar a ser las más ideales para invertir guiándonos en el éxito de los juegos que produjeron.

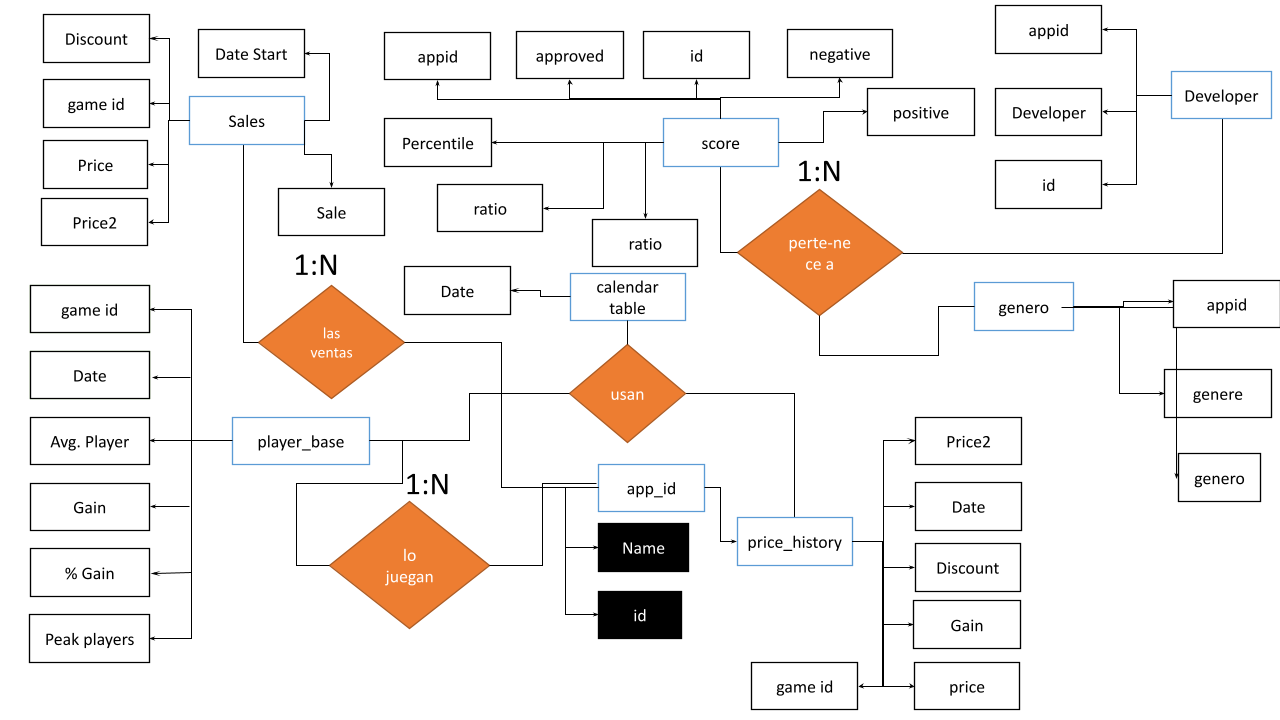
*Alcance*

El alcance del proyecto es de nivel estratégico ya que se puede tomar como referencia como afecta a la cantidad de jugadores activos la variación del precio a los distintos juegos y su rating.

Este tablero realiza un análisis descriptivo de los distintos juegos.

*Gráfico de entidad relación.*

En el siguiente gráfico se muestra, como el título indica, la relación de las distintas variables que serán protagonistas en este estudio. La representación de esta manera permite diferenciar rápidamente cada una de las entidades y evitar duplicados o información que se superponen entre sí.



*Tablas*

|  |  |  |
| --- | --- | --- |
| Calendar table | | |
| Clave | Campo | Tipo de dato |
| - | Date | Fecha |

|  |  |  |
| --- | --- | --- |
| price\_history | | |
| Clave | Campo | Tipo de dato |
| PK,FK | Date | Fecha |
| - | Discount | Float |
| - | Gain | Float |
| FK | game\_id | Int |
| - | Price | Float |
| - | Price2 | Float |

|  |  |  |
| --- | --- | --- |
| player\_base | | |
| Clave | Campo | Tipo de dato |
| FK | Date | Fecha |
| - | %Gain | Float |
| - | Gain | Float |
| PK,FK | game\_id | Int |
| - | Avg. Players | Float |
| - | Peak Players | Int |

|  |  |  |
| --- | --- | --- |
| app\_id | | |
| Clave | Campo | Tipo de dato |
| PK | id | Int |
| - | Name | Varchar(255) |

|  |  |  |
| --- | --- | --- |
| Sales | | |
| Clave | Campo | Tipo de dato |
| PK,FK | Date Start | Fecha |
| - | Discount | Float |
| - | sale | Varchar(255) |
| FK | game\_id | Int |
| - | Price | Float |
| - | Price2 | Float |

|  |  |  |
| --- | --- | --- |
| genre | | |
| Clave | Campo | Tipo de dato |
| PK | id | Int |
| - | genre | Varchar(255) |
| FK | appid | Int |

|  |  |  |
| --- | --- | --- |
| Developer | | |
| Clave | Campo | Tipo de dato |
| PK | id | Int |
| - | developer | Varchar(255) |
| FK | appid | Int |

|  |  |  |
| --- | --- | --- |
| Score | | |
| Clave | Campo | Tipo de dato |
| FK | app\_id | Int |
| - | Aproved | Bool |
| PK | id | Int |
| - | negative | Int |
| - | positive | Int |
| - | percentile | Float |
| - | ratio | Float |
| - | reviews | Int |

*Modelo relacional* Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

*transformaciones en las tablas*

Los cambios a las tablas fueron implementados el 28/2/2022

***Tablas:***

Se cargan las tablas por import de csv y se procede a los cambios de denominación de cada una de ellas.

Una captura de pantalla de un celular

Descripción generada automáticamente

**Tabla Sales:**

Crear origen (importa la tabla)



Armar la navegacion



Promover encabezados



**Tabla price\_history:**

Crear origen (importa la tabla)



Armar la navegación



Promover encabezados



Cambiar tipos:

date a type date,

price a currency.type,

gain a text,

discount a any,

game\_id a int64.



Renombrar columnas de Price a price



**Tabla app\_id:**

Crear origen (importa la tabla)



Armar la navegación



Promover encabezados



Cambiar tipos:

Id a int64,

Nombre a text.



Cambiar nombre columnas nombre a name



**Table genre:**

Crear origen (importa la tabla)



Armar la navegación



Promover encabezados



Cambiar tipos:

Id a int64,

Genre a text.



**Tabla developer:**

Crear origen (importa la tabla)

 Armar la navegación



Promover encabezados



Cambiar tipos:

Id a int64,

developer a text.



**Tabla score:**

Crear origen (importa la tabla)

 Armar la navegación



Promover encabezados



Cambiar tipos:

Id a int64,

appId a int64,

positive a int64,

negative a int64.



**Tabla Player\_base:**

Crear origen (importa la tabla)



Cambiar tipos:

Column1 a text,

Column2 a text,

Column3 a text,

Column4 a text,

Column5 a text,

Column6 a text,



Eliminar filas en blanco



Renombrar columnas

Column1 a month,

Column2 a avg players,

Column3 a gain,

Column4 a %gain,

Column5 a peak players,

Column6 a gameid,



Reemplazamos valor last 30 days, february 2022 a month



Quitar filas superiores



Ordenar las filas



Cambiar tipo Month a type date



Reordenar las filas



Renombramos month a date.



Creamos la tabla calendar.

Tabla

Descripción generada automáticamente con confianza media

*Medidas Calculadas*

Los cambios de las medidas calculadas fueron creadas el 4/3/2022

Las medidas calculadas que armarmos son:

En Price history

Agregamos Price 2 ya que Price se generaba erróneamente.

Price2 = DIVIDE((price\_history[price]),100)

En score:

Agregamos approved que busca juegos populares, con un score ratio mayor a 0.75 y que valgan menos de 10 usd.

approved = IF(AND(score[ratio] > 0.75, AND(score[positive]+score[negative] > 10000, CALCULATE(MAX(price\_history[Price2]), LASTDATE(price\_history[Date])) < 10)) ,TRUE ,FALSE)

calculamos el percentil.

Percentile = (score[positive] - score[negative])/(score[positive] + score[negative])

calculamos el ratio.

ratio = (score[positive] - score[negative])/(score[positive] + score[negative])

calculamos la cantidad de reviews.

Reviews = score[positive]+score[negative]

En Measure\_Table

Calculamos la cantidad de juegos

Amount\_Of\_Games = COUNTROWS(app\_id)

Sumamos todos las reviews negativas

Total\_Negative = SUM(score[negative])

Sumamos todas las reviews positivas

Total\_Positive = SUM(score[positive])

Sumamos todas las reviews

Total\_Reviews = SUM(score[Reviews])

*Tema elegido*

Imagen que contiene Tabla

Descripción generada automáticamente

Esta paleta de colores fue elegida por que se asemeja a la paletas de colores de Steam.

*Portada*

Interfaz de usuario gráfica

Descripción generada automáticamente

Esta es la portada del tablero en la cual se puede presenciar cual fue la ultima vez que se actualizo la base de datos.

*General* Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

En esta primer solapa podemos investigar cual fue el pico de jugadores de los distintos juegos así como a que genero pertenecen para poder comenzar a diferenciar distintos juegos que podrían ser de interés.

Interfaz de usuario gráfica

Descripción generada automáticamente

En esta segunda solapa podemos ver como es la comparación directa entre la base de jugadores activa de un juego y su precio en la plataforma de ventas. En el caso de no tener seleccionado ningún juego podemos ver como vario la cantidad de jugadores en simultaneo de todos ellos a través del tiempo. Algo interesante para remarcar es que podemos ver un pico de jugadores a mitad del 2020 en plena pandemia. ´podemos filtrar por juego y acotar los gráficos por fechas

Interfaz de usuario gráfica

Descripción generada automáticamente

En esta solapa podemos observar cómo es el “rating” de cada juego que tanto le ha gustado a los jugadores. En esta solapa podemos filtrar por rating, votos positivos , cantidad de reviews , votos negativos, nombre y por approved siendo este ultimo un filtro de tipo bool que indicaría que juegos tienen un rating > 0.75 mas de reviews y que salga menos de 10 usd

*Recoleccion de informacion*

Para recolectar la información utilizada en este proyecto nos dirigimos a 2 paginas web y comenzamos a scrappearlas con el siguiente código. Básicamente lo que hace este código cuando se corre es entrar en cada pagina y guardar cada código html de forma individual en una carpeta + una carpeta de error, las paginas ingresadas son : <https://steampricehistory.com/> y <https://steamcharts.com/> todo el código esta hecho con python3

Los ids fueron obtenidos desde este github: <https://github.com/nik-davis/steam-data-science-project> que arma en jupyternotebooks un archivo .py que obtiene desde la api de steam toda la informacionj con respecto a los ids,

import requests

import csv

import json

import time

INICIO = 0

FIN = 29237

#

STEAM\_PRICE = "https://steampricehistory.com/app/"

STEAM\_CHARTS = "https://steamcharts.com/app/"

PATH\_SP = "SCRAPPER/sprice"

PATH\_SC = "SCRAPPER/scharts"

PATH\_ERROR = "SCRAPPER/ERROR"

def scrapper\_(id\_):

steam\_price\_url = STEAM\_PRICE + str(id\_)

steam\_charts\_url = STEAM\_CHARTS + str(id\_)

steam\_price = requests.get(steam\_price\_url)

steam\_charts = requests.get(steam\_charts\_url)

try:

with open (PATH\_SP+ "/sp"+ str(id\_) + ".txt", "w") as sp\_txt:

with open (PATH\_SC + "/sc"+ str(id\_) + ".txt", "w") as sc\_txt:

sp\_txt.write(steam\_price.text)

sc\_txt.write(steam\_charts.text)

except:

with open (PATH\_ERROR + str(id\_) +"---ERROR---.txt", "w") as ERROR:

ERROR.write("ERROR")

def scrapper (idlist,start,end):

x = 0

for id\_pos in range(start, end+1):

id\_ = idlist[id\_pos]

print(id\_)

scrapper\_(id\_)

time.sleep(60)

print(x)

x += 1

def get\_id(PATH):

list\_ = []

with open (PATH,encoding="utf8") as file:

fcsv = csv.reader(file,delimiter=",")

for fila in fcsv:

list\_.append(fila[0])

return list\_

def main():

idlist = get\_id("app\_list.csv")

idlist.pop(0)

scrapper(idlist,INICIO,FIN)

main()

En el siguiente codigo agarramos todos los html que obtuvimos con el scrapper y nos dedicamos a parsear la información para guardarla de forma csv. Utilizando una librería llamada Beautifullsoup

from bs4 import BeautifulSoup

import csv

import json

import time

START = 0

END = 29237

"END\_END = 29237"

PARSED\_CSV\_PATH = "PARSER/"

SCRAPPER\_PATH = "SCRAPPER/"

HEADER\_PRICE\_HISTORY = ['Date', 'Price', 'Gain', 'Discount','game\_id']

HEADER\_SALES = ['Sale', 'Date Start', 'Price', 'Discount','game\_id']

HEADER\_PLAYER\_BASE = ['Month', 'Avg. Players', 'Gain', '% Gain', 'Peak Players','game\_id']

def parse(soup):

line\_list = []

line =soup.find\_all(['th', 'td'])

for col in line:

aux = col.get\_text().strip()

line\_list.append(aux)

#print(aux)

return line\_list

def parse\_table(soup,game\_id):

game\_id = game\_id.strip()

list\_csv = []

tr = soup.find\_all('tr')

for line in tr:

list\_csv.append(parse(line)+[game\_id])

return list\_csv[1:]

def parser(game\_id):

try:

with open (SCRAPPER\_PATH + "/sprice/sp"+ str(game\_id) + ".txt", "r") as sp\_txt, open (SCRAPPER\_PATH + "/scharts/sc"+ str(game\_id) + ".txt", "r") as sc\_txt:

soup\_sp = BeautifulSoup(sp\_txt, 'html.parser')

table\_separator\_sp = soup\_sp.find\_all('table')

if len(table\_separator\_sp) < 4: return [],[],[]

a , player\_price\_history, sales\_table, v = table\_separator\_sp

#print(player\_price\_history)

lph = parse\_table(player\_price\_history,game\_id)

ls = parse\_table(sales\_table,game\_id)

soup\_sc = BeautifulSoup(sc\_txt, 'html.parser')

aux=soup\_sc.find\_all('table')

#print(lph)

if len(aux) == 0: return [],[],[]

table\_player\_base = aux[0]

lpb = parse\_table(table\_player\_base,game\_id)

except FileNotFoundError:

return [],[],[]

return lpb, ls, lph

def write\_csv (idlist,start,end):

with open (PARSED\_CSV\_PATH + "player\_base.csv", "w") as pb\_csv, open (PARSED\_CSV\_PATH + "sales.csv", "w") as s\_csv, open (PARSED\_CSV\_PATH + "price\_history.csv", "w") as ph\_csv:

pb = csv.writer(pb\_csv, delimiter=';')

s = csv.writer(s\_csv, delimiter=';')

ph = csv.writer(ph\_csv, delimiter=';')

pb.writerow(HEADER\_PLAYER\_BASE)

s.writerow(HEADER\_SALES)

ph.writerow(HEADER\_PRICE\_HISTORY)

for game\_id\_pos in idlist[start:end+1]:

list\_pb, list\_s, list\_ph = parser(game\_id\_pos)

pb.writerows(list\_pb)

s.writerows(list\_s)

ph.writerows(list\_ph)

def get\_id(PATH):

list\_ = []

with open (PATH,encoding="utf8") as file:

fcsv = csv.reader(file,delimiter=",")

for fila in fcsv:

list\_.append(fila[0])

return list\_

def main():

idlist = get\_id("app\_list.csv")

idlist.pop(0)

write\_csv(idlist,START,END)

#add\_column\_number(csv)

main()

*Conclusion*

En conclusion pudimos recolectar y parsear los datos de forma tal en la que pudimos generar un tablero interactivo con un alcance estratégico en el cual distintas personas podrán interactuar con el y obtener la información que deseen, esta con un enfoque a la relación de como varia la tasa de jugadores activa en un juego a través del tiempo

*Futuras Lineas*

Para el futuro habría que mejorar y profundizar las tablas de player\_base y Price\_history , junto con esto también habría que meter mas énfasis en la variación del precio de los juegos, con especial atención en las fechas de rebajas.