|  |  |  |
| --- | --- | --- |
| Monserrat Castillo Quesada Universidad Nacional de Costa Rica, Heredia, Costa Rica  montserrat.castillo.quesada@est.una.ac.cr | Adrián Morales Pincay Universidad Nacional de Costa Rica,  Heredia, Costa Rica adrian.morales.pincay@est.una.ac.cr | Derek Espinach Murillo Universidad Nacional de Costa Rica,  Heredia, Costa Rica derek.espinach.murillo@est.una.ac.cr |

Venta e información de juegos digitales usando los datos de la tienda virtual Steam desde 2003 hasta 2019

|  |  |
| --- | --- |
| Ángel Quesada Jiménez Universidad Nacional de Costa Rica,  Heredia, Costa Rica angel.quesada.jimenez@est.una.ac.cr |  |

***Abstract*—La siguiente investigación permite aplicar los conceptos de algunas de las funciones que tiene el Lenguaje Python en el análisis de los datos. Utilizando las librerías Pandas y Matplotlib para la obtención de los datos y creación de las gráficas. Se obtuvo resultados de los juegos con más desarrollo de Steam.**

# I. INTRODUCCIÓN

El mercado digital en la era actual tiene un gran porcentaje de ventas en comparación con el mercado de videojuegos físicos, al punto de que gigantes de la industria como Microsoft y Sony en esta última generación de consolas de videojuegos han decidido remover el disco óptico de lectura de discos físicos. Dentro de esta industria, [[1]](#footnote-1)Steam, la plataforma de distribución de videojuegos más famosa y usada para computadoras personales se establece como una fuerza dominante. Con millones de usuarios y miles de títulos disponibles, Steam ha revolucionado la forma en que se consumen y distribuyen los videojuegos. Este documento presenta un análisis de datos sobre Steam y los videojuegos disponibles en esta plataforma hasta el año 2019, abordando algunas de las tendencias y estadísticas que definen esta plataforma. Buscaremos indagar sobre aspectos como la popularidad de géneros de videojuegos, el crecimiento de número anual de videojuegos, entre otros. Como nota adicional, se eligió Steam por su gran presencia en el mercado digital y porque este mantiene muchos de sus datos de venta abierto al público. Aunque este análisis apenas rasca la superficie de un tema tan amplio y dinámico, proporcionará una visión inicial de las tendencias más notables que rodean al mercado digital con este enfoque. Existe una diferencia entre juegos digitales y videojuegos, los juegos digitales pueden ser DLC, skins de juegos, entre otros tipos de atributos y en videojuegos es un juego electrónico que puede ser jugado por más de una persona por medio de un controlador.

# II. MARCO TEÓRICO

A continuación, se presenta diferentes aplicaciones de Python que se han utilizado para trabajos como:

1. *La aplicación del formato de datos base estándar de Weather Radar*

Resuelve el fenómeno de que una variedad de formatos de archivos de datos básicos coexista en CINRAD Weather Radar durante mucho tiempo. En terminales de aplicaciones empresariales, solo se necesita el método de formato de datos base estándar para analizar todos los datos base de los datos base del radar meteorológico CINRAD. Este artículo presenta un análisis de los datos base estándar del radar meteorológico e introduce cómo utilizar el lenguaje de programación Python para realizar lecturas de datos y otras operaciones gráficas en el formato de datos base estándar del radar meteorológico. [1]

*B. Análisis de datos dentro del área visual en Python.*

El lenguaje de programación Python proporciona un entorno de desarrollo adecuado tanto para tareas computacionales como de visualización. Una de las ventajas clave de Python es que permite a los desarrolladores utilizar paquetes que amplían el lenguaje para proporcionar capacidades avanzadas, como manipulación de matrices y matrices, procesamiento de imágenes, procesamiento de señales digitales y visualización. Se han creado varias herramientas populares de exploración y visualización de datos en Python.:[2]

*C. Análisis de circuitos con el lenguaje Python.*

En el análisis de circuitos a menudo aparecen cálculos y dibujos manuales complicados, lo que dificulta el análisis de circuitos. Los métodos manuales de análisis del circuito requieren mucho tiempo, son laboriosos y propensos a errores. Python puede resolver mejor el problema en el análisis de circuitos que de forma manual. Python es un lenguaje de programación interpretado de alto nivel y de propósito general que se usa ampliamente en informática e ingeniería científicas. [3]

*D. Traducción de pseudocódigo a lenguaje natural*

Traducir pseudocódigo a un lenguaje de programación puede resultar difícil y llevar mucho tiempo, especialmente para personas que no están familiarizadas con los lenguajes involucrados. Sin embargo, los avances recientes en el procesamiento del lenguaje natural (NLP) han hecho posible crear un modelo que puede convertir automáticamente pseudocódigo escrito en lenguaje natural en código en un lenguaje de programación específico. Este estudio de investigación propone una solución a este problema mediante el uso de un modelo transformador de PNL para traducir pseudocódigo a código Python automáticamente. [4]

1. *Desarrollo de gráficos de forma cualitativa y cuantitativa con Python.*

En los últimos años, la amplia aplicación del lenguaje Python ha hecho que su trabajo de análisis sea cada vez más valioso. Muchos algoritmos de análisis estático deben depender de la construcción de gráficos de llamadas. En este artículo, realizamos un análisis empírico comparativo de varias herramientas de gráficos de llamadas estáticas de Python ampliamente utilizadas, tanto cuantitativa como cualitativamente. [5]

*F. Optimización en la búsqueda de errores con Python*

Como lenguaje de programación dinámico, Python se utiliza ampliamente en muchos campos. Para los desarrolladores, varias características del lenguaje afectan la experiencia de programación. Para los investigadores, afectan la dificultad de desarrollar tareas como la búsqueda de errores y la optimización de la compilación. Investigaciones anteriores han demostrado que los programas con características dinámicas de Python son más propensos a cambios. Sin embargo, sabemos poco sobre el uso y el impacto de las características del lenguaje Python en proyectos Python del mundo real. [6]

*G. Desarrollo de aprendizaje experimental con lenguaje Python*

En este artículo se intenta esbozar la contribución del aprendizaje experiencial, la gamificación y las tecnologías emergentes a la enseñanza del lenguaje de programación Python. Se diseñaron cuatro escenarios con el fin de constituir una nueva propuesta didáctica para la consolidación de módulos específicos, incorporando elementos de juegos dinámicos, tecnología emergente y abordando el tema que se trata, de manera experiencial en un marco de aprendizaje auténtico mediante la resolución de problemas de la vida cotidiana. [7]

1. *Uso de juegos 3D para el aprendizaje de lenguajes de programación*

Los métodos de enseñanza y los enfoques de aprendizaje tradicionales ya no son apropiados para muchos niños. Según los estudios, estos enfoques perjudican la motivación y el interés de los estudiantes por estudiar programación. Según la investigación, los juegos pueden ayudar a aprender a programar en Python. El objetivo de este artículo es presentar nuestro juego educativo que desarrollamos en el motor de juego Unity 3D. El juego se centra en los conceptos básicos puros de programación en lenguaje Python.[8]

1. *Árbol de sintaxis aplicado a Python y otros lenguajes*

Este artículo propone un método que se basa en la semántica compartida para leer, escribir o manipular diseños a nivel de transistor. El objetivo final del marco es: leer un diseño de entrada escrito en una sintaxis específica y luego permitir escribir el mismo diseño en otra sintaxis. Primero, la descripción de entrada es analizada por una interfaz específica del lenguaje que la convierte en un árbol de sintaxis abstracta en memoria que sigue la semántica común. Esta actividad requiere agregar, quitar o reemplazar nodos, componentes o incluso subcircuitos completos. Por lo tanto, el marco está completamente escrito en C++ y sus API también están conectadas con Python. Todo el marco es de código abierto y está disponible en GitHub. [9]

1. *Traducción automatizada neuronal de lenguaje inglés a Python*

Escribir código utilizando lenguaje natural es una aplicación muy interesante de la traducción automática neuronal. Para lograr una pequeña parte de dicha aplicación, en este artículo intentamos generar fragmentos de código fuente de Python a partir de descripciones en inglés natural utilizando el conjunto de datos de Django. Entrenamos la arquitectura del transformador basada en la autoatención en los fragmentos del conjunto de datos. [10]

# III. DESCRIPCIÓN DE PÁGINA

Para la obtención de los datos para analizar y crear las gráficas se utilizó la librería Pandas para leer los datos del archivo CSVInvestigacion.csv y Matplotlib para el diseño de los gráficos. A continuación, se presenta la sintaxis de código Python para dar solución a los casos escogidos.

La información de los datos se obtuvo de la página www.kaggle.com, la cual es un distribuidos de datos públicos de diferentes aplicaciones a nivel global, las estadísticas y datos proporcionados en esta web pueden ser en su mayoría de páginas para descarga de juegos hasta de juegos mismos.

De la gran variedad de temas se decide seleccionar el tema de estadísticas generales de los diferentes juegos encontrados en Steam. En específico los juegos con encontrados actualizados desde el año 2019 hasta el año 2023.

*A. ¿Cuál es la cantidad de juegos con una cantidad de logros mayor o menos al requerido?*

Para la solución de este caso se implementó un algoritmo de sumatoria donde se utiliza len () que retorna el número de datos obtenidos

import pandas as pd

df=pd.read\_csv('/content/drive/MyDrive/steam\_games.csv')

df=pd.DataFrame(df)

def sumatoria\_juegos\_logros(df,cantidad):

mayor\_diez = df[df['achievements'] >cantidad]

menor\_o\_igual\_diez=df[df['achievements'] <=cantidad]

cantidad\_logros = len(mayor\_diez)

cantidad\_logros\_menor = len(menor\_o\_igual\_diez)

return cantidad\_logros, cantidad\_logros\_menor

cantidad=10

cantidad\_juegos\_mayor\_diez, cantidad\_juegos\_menor\_diez = sumatoria\_juegos\_logros(df,cantidad)

print("Cantidad de juegos con logros mayores a :"+cantidad+ cantidad\_juegos\_mayor\_diez)

print("Cantidad de juegos con logros menores a :"+cantidad+,cantidad\_juegos\_menor\_diez)

Resultado:

|  |  |
| --- | --- |
| Cantidad de juegos con logros mayores a 10 | 9632 |
| Cantidad de juegos con logros menores a 10 | 2562 |

*Tabla 1 sumatoria de juegos con una cantidad de logros mayores y menores a 10.*

Se concluye que existen 9632 juegos con logros mayores a 10 y un total de 2562 menos a 10.

*B. Obtener los desarrolladores (“developer” por su traducción en inglés) con un desarrollo de juegos mayores a 10 hasta el año 2019*

Para la solución se filtró la columna de developer en todas las que era el mismo tipo de developer y se agrupo todas las que su sumatoria fuera mayor a 10.

import pandas as pd

df=pd.read\_csv('/content/drive/MyDrive/steam\_games.csv')

df=pd.DataFrame(df)

def contardor\_juegos\_por\_columna(df): try: agrupado = df.groupby('developer').size().reset\_index(name='cantidad') resultados\_filtrados = agrupado[agrupado['cantidad'] > 10] return resultados\_filtrados except FileNotFoundError: print(f"El archivo {df} no se encuentra.") except Exception as e: print(f"Ocurrió un error: {str(e)}")

print("Los developer con una cantidad de juegos mayores a 10 acutalmente en Steam") print(contardor\_juegos\_por\_columna(df))

Resultado:

|  |  |
| --- | --- |
| Developer | Cantidad |
| 07th Expansion | 11 |
| 10tons Ltd | 16 |
| 11 bit studios | 13 |
| 34BigThings srl | 11 |
| AAAGAMES | 18 |
| Stillalive studios | 14 |
| Vadimalexepta | 11 |
| virtualRailroads | 17 |
| KnKo | 20 |
| 搞快点工作室 | 14 |

*Tabla 2 Desarrollador con una cantidad de juegos mayores a 10*

Se puede asumir que la mayoría de los juegos de Steam su desarrollador o desarrollo es de origen americano o asiático.

*C. Contador de juegos gratis y de paga*

Esta función cuenta los juegos gratis y de paga disponibles en la plataforma Steam. La funcion crea un nuevo DataFrame llamado gratis que contiene todas las filas donde la columna llamada 'original\_price' tiene el valor 'Free' (que generalmente se usa para indicar que un juego es gratuito). Ocurre los mismo para el nuevo DataFrame llamado pago, pero con los juegos que no son ‘Free’.

Después se calcula la longitud (el número de filas) del DataFrame gratis con el comando len() y se guarda en la variable gratis. Ocurre lo mismo con el DataFrame pago.

Al final de la función se devuelven (“return”) las variables que contienen la cantidad de juegos gratis y de paga.

import pandas as pd

df=pd.read\_csv('/content/drive/MyDrive/steam\_games.csv')

df=pd.DataFrame(df)

def contador\_juegos(df):

gratis = df[df['original\_price'] == 'Free']

pago = df[df['original\_price'] != 'Free']

cantidad\_gratis = len(gratis)

cantidad\_pago = len(pago)

return cantidad\_gratis, cantidad\_pago

# Llama a la función y almacena los resultados en variables respectivas.

cantidad\_gratis, cantidad\_pago = contador\_juegos(df)

# Print los resultados.

print("Cantidad de juegos gratis:", cantidad\_gratis)

print("Cantidad de juegos de pago:",cantidad\_pago)

Resultado:

|  |  |
| --- | --- |
| Costo | Cantidad de Juegos |
| Gratis | 1326 |
| De pago | 39507 |

*Tabla 3 Cantidad de juegos gratis y de paga.*

La gran mayoría de juegos y elementos (como contenido adicional para dichos juegos) son de paga.

*D. Ordenamiento de juegos por fecha disponibles en Steam.*

Convierte la columna 'release\_date' del DataFrame df en un formato de fecha (Año/Mes/Día). Posteriormente se ordena un nuevo df de manera ascendente y al final de la función se devuelve el df ordenado con las columnas 'release\_date' y 'name'.

def ordenar\_por\_fecha(df):

df['release\_date'] = pd.to\_datetime(df['release\_date'], errors='coerce')

ordenado\_fecha = df.sort\_values(by='release\_date')

return ordenado\_fecha[['release\_date', 'name']]

resultado\_ordenado = ordenar\_por\_fecha(df)

print(resultado\_ordenado)

print("Cantidad de juegos disponibles en Steam:",len(resultado\_ordenado))

Resultado:

“Cantidad de juegos disponibles en Steam: 40833”

|  |  |
| --- | --- |
| Fecha de salida | Nombre |
| 1981-04-22 | The Mystery of the Uurnog |
| 1983-06-19 | Dragon's Lair |
| … | … |
| NaT | Gamers Club |
| NaT | Gravia |

*Tabla 4 Juegos en orden de salida.*

Steam tiene una selección muy amplia de videojuegos que se extiende desde 1981 hasta el presente y además incluye juegos por anunciarse o que se encuentran en desarrollo. Una librería así de grande es de gran interés para posibles jugadores.

*E. Juegos en Steam ordenados por fecha después de la fundación de Steam y antes del 31/12/2019*

Al igual que el punto D, convierte la columna 'release\_date' del DataFrame df al formato de fecha (Año/Mes/Día). Se definen dos variables nuevas: fundación-steam que contiene la fecha de fundación de Steam (12 de septiembre de 2003) como un objeto de fecha y hasta-2019 que contiene la fecha límite del 31 de diciembre de 2019 como un objeto de fecha. Después se filtra el DataFrame df para seleccionar solo los juegos cuya fecha de lanzamiento está en el rango desde la fecha de fundación de Steam hasta el final del año 2019. Luego, se ordenan los juegos filtrados por su fecha de lanzamiento ascendentemente. Para finalizar se devuelve un nuevo DataFrame llamado juegos-post-steam que contiene solo las columnas 'release\_date' y 'name'.

import pandas as pd

def juegos\_desde\_fundacion\_hasta\_2019(df):

df['release\_date'] = pd.to\_datetime(df['release\_date'], errors='coerce')

fundacion\_steam = pd.to\_datetime('2003-09-12')

hasta\_2019 = pd.to\_datetime('2019-12-31')

juegos\_post\_steam = df[(df['release\_date'] >= fundacion\_steam) & (df['release\_date'] <= hasta\_2019)]

juegos\_post\_steam = juegos\_post\_steam.sort\_values(by='release\_date')

return juegos\_post\_steam[['release\_date', 'name']]

resultado\_juegos = juegos\_desde\_fundacion\_hasta\_2019(df)

print(resultado\_juegos)

print("Cantidad de juegos estrenados durante la existencia de Steam:",len(resultado\_juegos))

Resultado:

“Cantidad de juegos estrenados durante la existencia de Steam: 36149”

|  |  |
| --- | --- |
| Fecha de salida | Nombre |
| 2003-09-15 | Alien Shooter |
| 2003-09-16 | STAR WARS™ Jedi Knight II - Jedi Outcast™ |
| … | … |
| 2019-12-31 | Reality Incognita |
| NaT | RB: Axolotl |

*Tabla 5 Juegos en orden de salida desde la creación de Steam hasta el 31 de diciembre de 2019.*

La cantidad de juegos que han sido creados/añadidos a la librería de Steam desde su creación en comparación con los disponibles antes de su existencia (Únicamente 4,684 registrados en la librería antes de la creación de Steam) reflejan el desarrollo de la industria a través del siglo XXI donde la industria creció exponencialmente.

*F. Conocer el valor de los juegos según el precio digitado.*

Para la siguiente se utiliza tolits() para agregar todos los datos que encontremos con el precio deseado, donde nos dice juegos, expansiones, dlc, entre otros datos que pueden tener ese valor en dólares.

import pandas as pd

def contador\_valor\_juegos(dataframe,columna,valor):

if columna not in dataframe.columns:

return None

resultado = dataframe[dataframe[columna] == valor] ['name']. tolist()

return resultado

print ("Ingrese el valor en dólares de los juegos que desea conocer en ese precio:")

valor\_juego= input ("Ingresar en formato: $19.00 o Free: “) #en este caso será un valor de $1.00

result=contador\_valor\_juegos(df,"original\_price”, valor\_juego)

print(result)

Resultado:

|  |
| --- |
| Juegos precio $1.00 |
| Sketch tales |
| ArsonVille |
| Gnome Light |
| Super Zombie Arcade |
| 元素战争 Elemental war |
| Project Pastorate OST |

*Tabla 6 del valor de los juegos en Steam con un precio de $1.00.*

Se observa una cantidad de 6 juegos con un valor de $1.00, en este caso se pueden observar solo juegos, con otros precios se pueden dar la búsqueda de dlc, skins, paquetes, entre otras cosas.

*G. Obtiene la cantidad de veces que aparece cierto nombre en una categria.*

def contar\_nombres\_en\_csv(columna\_nombre, nombres\_a\_contar):

try:

conteos = {}

for nombre in nombres\_a\_contar:

conteo = df[df[columna\_nombre] == nombre].shape[0]

conteos[nombre] = conteo

return conteos

except FileNotFoundError:

return f"El archivo {archivo\_csv} no se encontró."

except Exception as e:

return f"Ocurrió un error: {str(e)}"

columna\_nombre = 'game\_details'

nombres\_a\_contar = ['Single-player', 'Multi-player', 'Downloadable Content', 'MMO', 'Online Multi-Player', 'Stats', 'Partial Controller Support']

conteos = contar\_nombres\_en\_csv(columna\_nombre, nombres\_a\_contar)

print(f"Cantidad de juegos según su etiqueta")

for nombre, conteo in conteos.items():

print(f"'{nombre}': {conteo}")

Resultado:

|  |  |
| --- | --- |
| Conteos de nombres en la columna 'game\_details' |  |
| 'Single-player' | 3547 |
| 'Multi-player' | 15 |
| 'Downloadable Content' | 159 |
| 'MMO' | 13 |
| 'Online Multi-Player' | 28 |
| 'Stats' | 1 |
| 'Partial Controller Support' | 17 |

*Tabla 7 Cantidad de apariciones por palabra*

Esta función le ayuda a analizar rápida y cuantitativamente la frecuencia de aparición de un nombre o categoría en particular en una columna particular de un archivo CSV. Esta característica es particularmente útil para tareas de procesamiento de datos y análisis exploratorio, ya que le permite ver rápidamente con qué frecuencia aparece una etiqueta o valor en particular en un conjunto de datos.

*H. Recolecta y ordena la cantidad de logros que tienen los juegos*

def ordenar\_lista(columna\_ordenar):

try:

lista = df[columna\_ordenar].tolist()

lista\_aux = [x for x in lista if not math.isnan(x)]

lista\_ordenada = sorted(lista\_aux)

return lista\_ordenada

except FileNotFoundError:

print(f"El archivo no fue encontrado.")

return []

except KeyError:

print(f"La columna '{columna\_ordenar}' no fue encontrada en el archivo CSV.")

return []

columna\_a\_ordenar = "Logros"

lista\_ordenada = ordenar\_lista( columna\_a\_ordenar)

print(lista\_ordenada)

Resultado:

|  |
| --- |
| Cantidad de logros por juego |
| 26.0 |
| 27.0 |
| 28.0 |
| 29.0 |
| 30.0 |
| 31.0 |

*Tabla 8 Cantidad de logros por juego*

Esta función toma una columna de un archivo CSV, elimina los valores NaN de esa columna, ordena los valores numéricamente y devuelve una lista de resultados. En general, esta función está diseñada para ayudarle a editar datos almacenados en archivos CSV, permitiéndole extraer, filtrar y ordenar columnas específicas.

*I. Filtro de desarrolladores según lo introducido por el usuario.*

Este código le permite buscar registros de juegos desarrollados por un desarrollador específico. El usuario ingresa el nombre del desarrollador que está buscando. La función buscar\_por\_desarrollador busca juegos que coincidan con este desarrollador y muestra los resultados. Esto le ayuda a encontrar juegos de un desarrollador específico dentro de su conjunto de datos*.*

import pandas as pd

def buscar\_por\_desarrollador(df, desarrollador\_buscado):

df['developer'] = df['developer'].fillna('')

juegos\_del\_desarrollador = df[df['developer'].str.contains(desarrollador\_buscado, case=False)]

return juegos\_del\_desarrollador

desarrollador\_buscado = input("Ingrese el nombre del desarrollador a buscar: ")

resultados = buscar\_por\_desarrollador(df, desarrollador\_buscado)

print(resultados)

Resultado:

El código cuenta con la capacidad de buscar registros de juegos desarrollados por un desarrollador específico. El usuario ingresa el nombre de aquel desarrollador que esté buscando. La función buscar\_por\_desarrollador se encarga de buscar los juegos que coincidan con este desarrollador y muestra los resultados. Esto le ayuda a encontrar juegos de un desarrollador específico dentro de su conjunto de datos.

Se ingresa el nombre del desarrollador de interés y seguidamente se muestra una lista de todo lo relacionado con este desarrollador. Esto facilita la búsqueda en nuestra gran base de datos y ayuda a aprender más sobre el trabajo de un desarrollador en particular.

*J. Contador de idiomas de los videojuegos de la plataforma steam.*

Esta función toma un DataFrame que contiene información sobre los juegos y usa la columna Idiomas para calcular la cantidad de juegos que admiten cada idioma. contador\_idiomas= pd.Series(lista\_idiomas).str.strip().value\_counts(): En esta línea, la lista de idiomas se convierte en una serie de pandas. A continuación, utilice str.strip() para eliminar espacios en blanco adicionales alrededor de cada idioma. A continuación, aplique Value\_counts() para contar cuántas veces aparece cada idioma en la serie. Esto creará una fila que contiene la cantidad de juegos por idioma.

import pandas as pd

def contar\_idiomas(df):

idiomas\_concatenados = ','.join(df['languages'].dropna())

lista\_idiomas = idiomas\_concatenados.split(',')

contador\_idiomas = pd.Series(lista\_idiomas).str.strip().value\_counts()

return contador\_idiomas

conteo\_idiomas = contar\_idiomas(df)

print("Conteo de juegos por idioma:")

print(conteo\_idiomas)

Resultado:

|  |  |
| --- | --- |
| Idioma | Cantidad por idioma |
| English | 40,779 |
| German | 14,492 |
| French | 13,878 |
| Spanish-Spain | 12,818 |
| Russian | 11,019 |
| Italian | 10,682 |
| Japanese | 9,712 |
| Simplified Chinese | 8,130 |
| Portuguese-Brazil | 6,098 |
| Polish | 5,693 |
| Korean | 5,405 |
| Traditional Chinese | 5,014 |
| Portuguese | 2,941 |
| Dutch | 2,849 |
| Czech | 2,446 |
| Turkish | 2,366 |
| Swedish | 1,741 |
| Hungarian | 1,647 |
| Arabic | 1,488 |
| Ukrainian | 1,483 |
| Danish | 1,479 |
| Finnish | 1,338 |
| Norwegian | 1,316 |
| Thai | 1,162 |
| Romanian | 1,093 |
| Greek | 1,082 |
| Bulgarian | 972 |
| Spanish-Latin America | 855 |
| Vietnamese | 423 |

*Tabla 9 Cantidad de idiomas de videojuegos en la plataforma Steam*

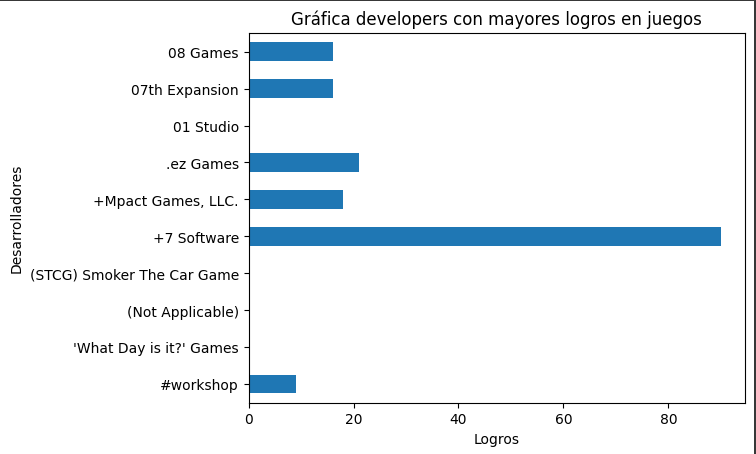
El código, junto con la función contador idiomas facilita el comprender qué idiomas se utilizan en el conjunto de datos del juego. Esta función cuenta la cantidad de juegos disponibles en cada idioma. Esto ayuda a ver cómo se distribuyen los idiomas en los juegos y qué tan accesibles son en diferentes regiones. Esto puede ser importante para que los desarrolladores y editores de juegos tomen decisiones de localización y lleguen a audiencias de todo el mundo. Para resumir, esta función proporciona de manera sencilla una forma de obtener información sobre el idioma de los juegos en su conjunto de datos.

# V. ANÁLISIS DE LOS DATOS

Para un obtener un análisis de los datos óptimo, se importa la librería Matplotlib y de esta manera generar gráficos que nos ayuden a visualizar aún mejor la información obtenida mediante los métodos realizados en Python.

Inicialmente, según la Descripción del procesamiento de la Información, tomando en cuenta el método del punto E, se procede a utilizar la función plot.barh(), de esta manera generar un gráfico horizontal donde se visualiza la información del resultado de los developers con diferentes variedad de logros por todos sus juegos desarrollados.

Además, se implementa la función plot.barh() para generar un Grafico Pie, que nos ayuda a visualizar cantidad de logros por developer.



*Gráfico 1 Developers y su cantidad de logros por juegos desarrollados.*

Seguidamente se tiene una comparativa visual de un gráfico de líneas para los datos de juegos gratis y juegos de paga. Se está utilizando la librería “seaborn” para añadir etiquetas debajo de las barras.

Gráfico

Descripción generada automáticamente

*Gráfico 2 Comparativa de juegos gratis y de paga.*

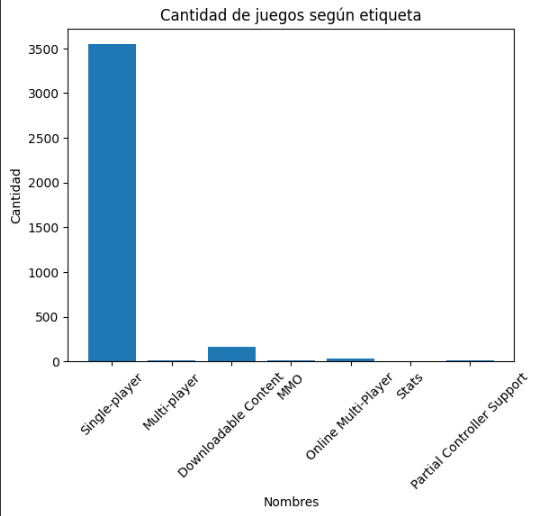
En el siguiente gráfico se visualiza la distribución de las fechas de lanzamiento de videojuegos, mostrando que en años más recientes se tiene un crecimiento exponencial de la producción y lanzamiento de bienes digitales en la plataforma.

Gráfico, Histograma

Descripción generada automáticamente

*Gráfico 3 Histograma de lanzamiento de videojuegos*

Esta función utiliza la biblioteca Matplotlib para crear un gráfico de barras altamente personalizado que muestra la cantidad de nombres para una columna de datos específica. No solo muestra efectivamente la cantidad de nombres en una columna, sino que también cuida la belleza y legibilidad del gráfico. Esto facilita la comunicación de resultados y mejora la comprensión de la información que se muestra en la visualización.

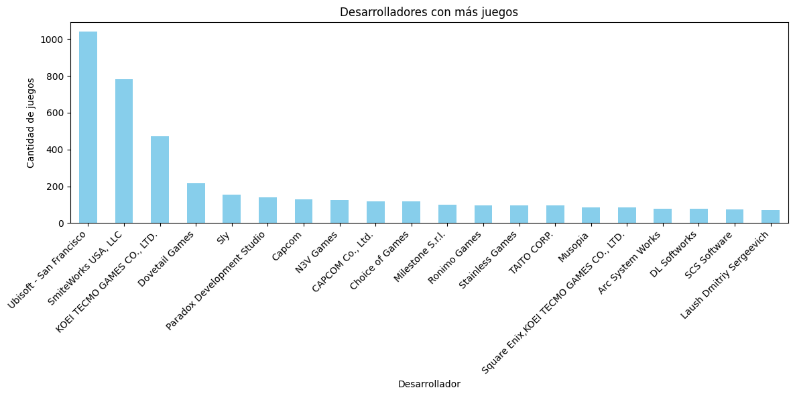


*Gráfico 4 Comparación de cantidad de tipos de juegos*

Este código realiza una gráfica la cual muestra los 20 desarrolladores de videojuegos y su respectiva cantidad de videojuegos por medio de columnas verticales, apareciendo debajo los nombres del desarrollador respectivo

Se importa la biblioteca Matplotlib, la cual genera gráficos que ayudan a visualizar mejor la información obtenida a través de métodos ejecutados en Python.

Utilice df['developer'].value\_counts() para calcular la cantidad de juegos por desarrollador. Esto crea una fila que muestra la cantidad de juegos desarrollados por cada desarrollador en el conjunto de datos.Utilice .head(20) para seleccionar los 20 principales desarrolladores con más juegos. dejando ver de manera grafica los desarrolladores más influyentes,creando un gráfico de barras usando Developer\_counts.plot(kind='bar', color='skyblue'). Esto representa la cantidad de juegos por desarrollador como una barra en el gráfico.



*Gráfico 5 los 20 desarrolladores con más juegos en el mercado.*

El código realiza una gráfica la cual muestra los 20 desarrolladores de videojuegos y su respectiva cantidad de videojuegos por medio de columnas verticales, apareciendo debajo los nombres del desarrollador respectivo

Se importa la biblioteca Matplotlib, la cual genera gráficos que ayudan a visualizar de una mejor manera la información obtenida a través de métodos ejecutados en Python.

Se da uso de df['developer'].value\_counts() para calcular la cantidad de juegos por desarrollador. Esto crea una fila que muestra la cantidad de juegos desarrollados por cada desarrollador en el conjunto de datos Seguidamente se utiliza. head (20) para seleccionar los 20 principales desarrolladores con más juegos, dejando ver de manera gráfica los desarrolladores más influyentes, creando un gráfico de barras usando Developer\_counts.plot(kind='bar', color='skyblue'). Esto representa la cantidad de juegos por desarrollador como una barra en el gráfico.

VI. CONCLUSIONES Y RECOMENDACIONES.

Al completar el proyecto se reconoce la importancia de cómo usar la programación para analizar datos y extraer información relevante. La implementación de funciones y la creación de gráficos han permitido extraer y presentar visualmente información valiosa de manera efectiva. Esto nos dio una mejor comprensión de las tendencias y patrones que rodean a los videojuegos disponibles en Steam.

El enfoque práctico y visual utilizado en el proyecto facilita la comprensión de los datos, lo cual es esencial para tomar decisiones informadas. Este proyecto proporciona herramientas útiles para explorar y comprender el uso de CSVs. Al analizar estos datos, podemos identificar tendencias emergentes, preferencias u otro tipo de dato a resaltar.

Este proyecto demuestra las capacidades de Python como una poderosa herramienta para el análisis de datos en situaciones del mundo real. La combinación de funcionalidad y gráficos crea una experiencia que ayuda a los usuarios a comprender y utilizar la información de manera más efectiva. Este enfoque práctico y visual abre nuevas oportunidades para comprender y mejorar el rendimiento de los productos y servicios en el mercado en constante evolución.

# VII. REFERENCIAS

[1] Y. Wen, Z. Jing, and W. Huan, “Application of Python in Weather Radar Standard Base Data,” in 2019 International Conference on Meteorology Observations (ICMO), Chengdu, China: IEEE, Dec. 2019, pp. 1–3. doi: 10.1109/ICMO49322.2019.9076594.

[2] E. W. Anderson, G. A. Preston, and C. T. Silva, “Using Python for Signal Processing and Visualization,” Comput. Sci. Eng., vol. 12, no. 4, pp. 90–95, 2010, doi: 10.1109/MCSE.2010.91.

[3] H. Zhang, P. Yang, and Y. Niu, “Application of Python Scientific computing library and Simulation in Circuit Analysis,” in 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India: IEEE, Apr. 2023, pp. 892–898. doi: 10.1109/CSNT57126.2023.10134600.

[4] S. P. Tiwari, S. Prasad, and M. G. Thushara, “Machine Learning for Translating Pseudocode to Python: A Comprehensive Review,” in 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India: IEEE, May 2023, pp. 274–280. doi: 10.1109/ICICCS56967.2023.10142254.

[5] L. Yu, “Empirical Study of Python Call Graph,” in 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA: IEEE, 2019, pp. 1274–1276. doi: 10.1109/ASE.2019.00160.

[6] Y. Peng, Y. Zhang, and M. Hu, “An Empirical Study for Common Language Features Used in Python Projects,” in 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA: IEEE, 2021, pp. 24–35. doi: 10.1109/SANER50967.2021.00012.

[7] C. Velaora and A. Kakarountas, “Logic Design as an Enabler to Python Programming Language Teaching,” in 2019 Panhellenic Conference on Electronics & Telecommunications (PACET), Volos, Greece: IEEE, 2019, pp. 1–6. doi: 10.1109/PACET48583.2019.8956286.

[8] M. Hlavaty, A. Kozakova, and O. Haffner, “Application for Python Programming Language Education Developed by Unity Engine,” in 2022 Cybernetics & Informatics (K&I), Visegrád, Hungary: IEEE, Sep. 2022, pp. 1–6. doi: 10.1109/KI55792.2022.9925955.

[9] N. Dall’Ora, S. Azam, E. Fraccaroli, A. Alberts, and F. Fummi, “A Common Manipulation Framework for Transistor-Level Languages,” in 2021 Forum on specification & Design Languages (FDL), Antibes, France: IEEE, Sep. 2021, pp. 01–07. doi: 10.1109/FDL53530.2021.9568379.

[10] M. Shah, R. Shenoy, and R. Shankarmani, “Natural Language to Python Source Code using Transformers,” in 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India: IEEE, Jun. 2021, pp. 1–4. doi: 10.1109/CONIT51480.2021.9498268.

10.1109/SIITME53254.2021.9663677.

1. <https://store.steampowered.com> [↑](#footnote-ref-1)