

MEMORIA PRÁCTICA 1. SISTEMAS DE INFORMACIÓN



MARCOS BONILLA CUBERO 70068909F

PABLO GRACIA CORREA 50257261F

<https://github.com/pgraciac/PracticaSI>

Para el ejercicio de los diagramas hemos decidido realizarlo con la herramienta diagrams.net, Creando un esquema principal a papel y luego convirtiéndolo correctamente en la aplicación.

Los diagramas han quedado así (adjuntamos las fotos en el .rar para mayor claridad):

El diagrama de flujo describe un proceso de auditoría y análisis de datos, organizado en seis departamentos:

- Cliente:**
 - Envía datos.
 - Recibe email.
 - Recibe información.
- Departamento de contrataciones:**
 - Recibe archivos.
 - Reenvía archivos.
 - Envía email con respuesta negativa.
 - Informe de usuarios.
 - Listado de páginas web.
 - Informe final.
 - Envía conclusiones.
 - Comunica información.
 - Viabilidad.
 - Inicio.
 - Presupuesto.
- Director de auditorías:**
 - Recibe datos.
 - Decide: Datos válidos? (Sí/No).
 - Envía datos.
 - Datos.
 - Reporte.
 - Informe.
 - Reporte.
 - Conclusiones.
 - Decide: Visto bueno? (Sí/No).
 - Redacta email negativo.
 - Decide: Hay personal? (Sí/No).
 - Redacta email con fecha de disponibilidad.
 - Redacta email con disponibilidad inmediata.
 - Email.
- Departamento de formación de personal:**
 - Recibe datos.
 - Analiza contraseñas.
 - Analiza SPAM.
 - Realiza reporte de usuarios críticos.
 - Envía reporte.
- Responsable de legal:**
 - Recibe datos.
 - Analiza.
 - Cookies.
 - Aviso legal.
 - Protección de datos.
 - Completa informe.
 - Envía informe.
- Departamento de inteligencia artificial:**
 - Recibe datos.
 - Evalúa conexiones de los usuarios.
 - Genera reporte.
 - Envía reporte.

El flujo comienza con el Cliente enviando datos al Departamento de contrataciones. Este departamento envía archivos al Director de auditorías, quien evalúa si los datos son válidos. Si no lo son, se reenvían al Departamento de contrataciones. Si son válidos, se envían al Departamento de formación de personal y al Responsable de legal. El Responsable de legal analiza los datos y genera un informe, el cual es enviado al Departamento de inteligencia artificial. Este departamento evalúa las conexiones de los usuarios y genera un reporte, el cual es enviado al Director de auditorías. El Director de auditorías también genera un informe y un reporte, los cuales son enviados al Departamento de contrataciones. Este departamento envía conclusiones al Director de auditorías, quien decide si está visto bueno. Si no lo está, redacta un email negativo. Si está visto bueno, decide si hay personal. Si no hay personal, redacta un email con fecha de disponibilidad. Si hay personal, redacta un email con disponibilidad inmediata. Ambos emails son enviados al Departamento de contrataciones, quien comunica la información y genera un informe final. El informe final es enviado al Cliente, quien recibe la información y envía un email al Departamento de contrataciones.

UML

Users:

```
def insertarUsers(user):
    con = sqlite3.connect('PracticaSistemas.db')
    cur = con.cursor()
    cur.execute(
        "CREATE TABLE IF NOT EXISTS users(name text primary key, telefono int, contrasena text, provincia text,"
        " permisos int, totalEmails int, phishingEmails int, clicadosEmails int, fechas int, ips int)"

    if (user[list(user.keys())[0]]['telefono'] != 'None':
        telefono = (user[list(user.keys())[0]]['telefono'])
    else:
        telefono = 'NULL'

    if (user[list(user.keys())[0]]['provincia'] != 'None':
        provincia = (user[list(user.keys())[0]]['provincia'])
    else:
        provincia = 'NULL'

    cur.execute(f"INSERT INTO users VALUES('{list(user.keys())[0]}',{telefono},{
        f'\"{user[list(user.keys())[0]]['contrasena']}\"},{
        f'\"{user[list(user.keys())[0]]['permisos']}\"},{
        f'\"{user[list(user.keys())[0]]['emails']}[total]}\"},{
        f'\"{user[list(user.keys())[0]]['emails']}[phishing]}\"},{
        f'\"{len((user[list(user.keys())[0]]['fechas']))},{len((user[list(user.keys())[0]]['ips']))})")

    cur.execute("CREATE TABLE IF NOT EXISTS fecha(fecha text,name text, foreign key(name) references users(name))")
    for fecha in (user[list(user.keys())[0]]['fechas']):
        cur.execute(f"INSERT INTO fecha VALUES('{fecha}','{list(user.keys())[0]}')")

    cur.execute("CREATE TABLE IF NOT EXISTS ip(ip text,name text, foreign key(name) references users(name))")
    for ip in (user[list(user.keys())[0]]['ips']):
        cur.execute(f"INSERT INTO ip VALUES('{ip}','{list(user.keys())[0]}')")

    con.commit()
    con.close()
```

Como en el json hay algunos valores que están como “None” y eso no lo acepta la base de datos los cambiamos por “NULL”. En el caso de los valores de las fechas y las ips hemos decidido insertar en la tabla de usuarios el total de estos campos y crear una tabla aparte con un campo name que se relaciona con el usuario de esta tabla y otro campo para la fecha.

Legal:

```
def insertarLegal(web):
    con = sqlite3.connect('PracticaSistemas.db')
    cur = con.cursor()
    cur.execute(
        "CREATE TABLE IF NOT EXISTS legal(name text primary key, cookies int, aviso int, proteccionDeDatos int, creacion int)"

    cur.execute(
        f"INSERT INTO legal VALUES('{list(web.keys())[0]}',{web[list(web.keys())[0]]['cookies']},{
        f'\"{web[list(web.keys())[0]]['proteccion-de-datos']}\"},{web[list(web.keys())[0]]['creacion']})")

    con.commit()
    con.close()
```

Y para importar los datos de la base de datos a un dataframe usamos pandas:

```
def getFromDB(table):
    con = sqlite3.connect('PracticaSistemas.db')
    df = pd.read_sql_query(f"SELECT * FROM {table}", con)
    con.commit()
    con.close()
    return df
```

Para las estadísticas que nos piden en este ejercicio usamos funciones que ya están implementadas:

```
def basicStats(dfUsers, dfLegal):
    print("Muestras de usuarios: " + str(dfUsers.shape[0]))
    print("Muestras de legal: " + str(dfLegal.shape[0]))
    print("Media del total de fechas en las que se ha iniciado sesión: " + str(dfUsers["fechas"].mean()))
    print("Desviación estándar del total de fechas en las que se ha iniciado sesión: " + str(
        dfUsers["fechas"].std(axis=0)))
    print("Media del total de ips detectadas: " + str(dfUsers["ips"].mean()))
    print("Desviación estándar del total de ips detectadas: " + str(dfUsers["ips"].std(axis=0)))
    print("Media del total de emails recibidos: " + str(dfUsers["totalEmails"].mean()))
    print("Desviación estándar del total de emails recibidos: " + str(dfUsers["totalEmails"].std(axis=0)))
    print("Máximo del total de fechas que se ha iniciado sesión: " + str(dfUsers["fechas"].max()))
    print("Mínimo del total de fechas que se ha iniciado sesión: " + str(dfUsers["fechas"].min()))
    print("Máximo del total de emails recibidos: " + str(dfUsers["totalEmails"].max()))
    print("Mínimo del total de emails recibidos: " + str(dfUsers["totalEmails"].min()))
```

RESULTADOS

Muestras de usuarios: 30

Muestras de legal: 20

Media del total de fechas en las que se ha iniciado sesión: 9.866666666666667

Desviación estándar del total de fechas en las que se ha iniciado sesión: 6.055680338972335

Media del total de ips detectadas: 9.733333333333333

Desviación estándar del total de ips detectadas: 6.141623582562274

Media del total de emails recibidos: 247.86666666666667

Desviación estándar del total de emails recibidos: 141.44274663167653

Máximo del total de fechas que se ha iniciado sesión: 20

Mínimo del total de fechas que se ha iniciado sesión: 1

Máximo del total de emails recibidos: 493

Mínimo del total de emails recibidos: 20

EJERCICIO 3

Para este ejercicio lo primero que hacemos es hacer los cuatro grupos que nos dicen, por permisos y por número de emails:

```
def statsByGroups(dfUsers):
    dfPermisos0 = dfUsers[dfUsers["permisos"] == 0]
    dfPermisos1 = dfUsers[dfUsers["permisos"] == 1]
    dfEmailsMas = dfUsers[dfUsers["totalEmails"] >= 200]
    dfEmailsMenos = dfUsers[dfUsers["totalEmails"] < 200]
```

Y ahora sacamos las estadísticas con funciones ya implementadas:

```
def statsByGroups(dfUsers):
    dfPermisos0 = dfUsers[dfUsers["permisos"] == 0]
    dfPermisos1 = dfUsers[dfUsers["permisos"] == 1]
    dfEmailsMas = dfUsers[dfUsers["totalEmails"] >= 200]
    dfEmailsMenos = dfUsers[dfUsers["totalEmails"] < 200]
    print("\nTotal de observaciones:\n")
    print("Observaciones totales de phishing de usuarios con permiso 0: " + str(dfPermisos0['clicadosEmails'].sum()))
    print("Observaciones totales de phishing de usuarios con permiso 1: " + str(dfPermisos1['clicadosEmails'].sum()))
    print("Observaciones totales de phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['clicadosEmails'].sum()))
    print("Observaciones totales de phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['clicadosEmails'].sum()))
    print("\nTotal Valores Ausentes\n")
    print("Número de valores ausentes en phishing de usuarios con permiso 0: " + str(dfPermisos0['phishingEmails'].isnull().sum()))
    print("Número de valores ausentes en phishing de usuarios con permiso 1: " + str(dfPermisos1['phishingEmails'].isnull().sum()))
    print("Número de valores ausentes en phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['phishingEmails'].isnull().sum()))
    print("Número de valores ausentes en phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phishingEmails'].isnull().sum()))
    print("\nMedias:\n")
    print("Media de emails de phishing de usuarios con permiso 0: " + str(dfPermisos0['phishingEmails'].mean()))
    print("Media de emails de phishing de usuarios con permiso 1: " + str(dfPermisos1['phishingEmails'].mean()))
    print("Media de emails de phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['phishingEmails'].mean()))
    print("Media de emails de phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phishingEmails'].mean()))
    print("\nMedianas:\n")
    print("Mediana de emails de phishing de usuarios con permiso 0: " + str(dfPermisos0['phishingEmails'].median()))
    print("Mediana de emails de phishing de usuarios con permiso 1: " + str(dfPermisos1['phishingEmails'].median()))
    print("Mediana de emails de phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['phishingEmails'].median()))
    print("Mediana de emails de phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phishingEmails'].median()))
    print("\nVarianzas:\n")
    print("Varianza de emails de phishing de usuarios con permiso 0: " + str(dfPermisos0['phishingEmails'].var()))
    print("Varianza de emails de phishing de usuarios con permiso 1: " + str(dfPermisos1['phishingEmails'].var()))
    print("Varianza de emails de phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['phishingEmails'].var()))
    print("Varianza de emails de phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phishingEmails'].var()))
```

```
print("\nMáximos:\n")
print("Máximo de emails de phishing de usuarios con permiso 0: " + str(dfPermisos0['phishingEmails'].max()))
print("Máximo de emails de phishing de usuarios con permiso 1: " + str(dfPermisos1['phishingEmails'].max()))
print("Máximo de emails de phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['phishingEmails'].max()))
print("Máximo de emails de phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phishingEmails'].max()))
print("\nMínimos:\n")
print("Mínimo de emails de phishing de usuarios con permiso 0: " + str(dfPermisos0['phishingEmails'].min()))
print("Mínimo de emails de phishing de usuarios con permiso 1: " + str(dfPermisos1['phishingEmails'].min()))
print("Mínimo de emails de phishing de usuarios con 200 emails o más: " + str(dfEmailsMas['phishingEmails'].min()))
print("Mínimo de emails de phishing de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phishingEmails'].min()))
```

RESULTADOS:

Total de observaciones:

Observaciones totales de phishing de usuarios con permiso 0: 537

Observaciones totales de phishing de usuarios con permiso 1: 1012

Observaciones totales de phishing de usuarios con 200 emails o más: 1172

Observaciones totales de phishing de usuarios con menos de 200 emails: 377

Total Valores Ausentes

Número de valores ausentes en phishing de usuarios con permiso 0: 0

Número de valores ausentes en phishing de usuarios con permiso 1: 0

Número de valores ausentes en phishing de usuarios con 200 emails o más: 0

Número de valores ausentes en phishing de usuarios con menos de 200 emails: 0

Emails Totales:

Emails totales de phishing de usuarios con permiso 0: 1277

Emails totales de phishing de usuarios con permiso 1: 2003

Emails totales de phishing de usuarios con 200 emails o más: 2581

Emails totales de phishing de usuarios con menos de 200 emails: 699

Medias:

Media de emails de phishing de usuarios con permiso 0: 79.8125

Media de emails de phishing de usuarios con permiso 1: 143.07142857142858

Media de emails de phishing de usuarios con 200 emails o más: 143.38888888888889

Media de emails de phishing de usuarios con menos de 200 emails: 58.25

Medianas:

Mediana de emails de phishing de usuarios con permiso 0: 41.0

Mediana de emails de phishing de usuarios con permiso 1: 138.0

Mediana de emails de phishing de usuarios con 200 emails o más: 134.5

Mediana de emails de phishing de usuarios con menos de 200 emails: 41.0

Varianzas:

Varianza de emails de phishing de usuarios con permiso 0: 9881.229166666666

Varianza de emails de phishing de usuarios con permiso 1: 12490.840659340658

Varianza de emails de phishing de usuarios con 200 emails o más: 15699.545751633985

Varianza de emails de phishing de usuarios con menos de 200 emails: 1945.1136363636363

Máximos:

Máximo de emails de phishing de usuarios con permiso 0: 382

Máximo de emails de phishing de usuarios con permiso 1: 372

Máximo de emails de phishing de usuarios con 200 emails o más: 382

Máximo de emails de phishing de usuarios con menos de 200 emails: 133

Mínimos:

Mínimo de emails de phishing de usuarios con permiso 0: 0

Mínimo de emails de phishing de usuarios con permiso 1: 1

Mínimo de emails de phishing de usuarios con 200 emails o más: 0

Mínimo de emails de phishing de usuarios con menos de 200 emails: 1

EJERCICIO 4

Para este ejercicio hemos descargado la lista de contraseñas (la pequeña) de la web <https://crackstation.net/> y hemos creado un diccionario de esas contraseñas hasheadas de esta forma:

```
def diccionarioHasheado():
    dicti = open("realhuman_phill.txt", "r", encoding='ISO-8859-1')
    diccionario = dicti.read()
    diccionario = list(diccionario.split("\n"))
    diccionarioHash = open("diccionarioHash.txt", "w")
    for dic in diccionario:
        diccionarioHash.write(hashlib.md5(dic.encode('ISO-8859-1')).hexdigest() + "\n")
    dicti.close()
    diccionarioHash.close()
```

Y luego hemos sacado una lista con las contraseñas vulnerables de esta forma:

```
def contraseñasVulnerables(dfUsers):
    contraseñas = list(dfUsers['contraseña'])
    with open("diccionarioHash.txt", "r") as dic:
        diccionario = dic.read()
        diccionario = list(diccionario.split("\n"))
        vulnerables = []
    for contraseña in contraseñas:
        if contraseña in diccionario:
            vulnerables.append(contraseña)
    return vulnerables
```

Por el motivo de que los diccionarios pesan mucho y no se pueden subir a GitHub y además la operación donde compara las contraseñas no es instantánea hemos sacado la lista de los hashes de contraseñas vulnerables y hemos trabajado con ella.

```
vulnerables = ['5f4dcc3b5aa765d61d8327deb882cf99', '3bf1114a986ba87ed28fc1b5884fc2f8',
               '276f8db0b86edaa7fc805516c852c889', '84d961568a65073a3bcf0eb216b2a576',
               '0acf4539a14b3aa27deeb4cbdf6e989f', '1660fe5c81c4ce64a2611494c439e1ba',
               'd16d377af76c99d27093abc22244b342', 'eb0a191797624dd3a48fa681d3061212',
               '714ab9fbdad5c5da1b5d34fe1a093b79', 'd8578edf8458ce06fbc5bb76a58c5ca4',
               '4297f44b13955235245b2497399d7a93', '37b4e2d82900d5e94b8da524fbeb33c0',
               'd0763edaa9d9bd2a9516280e9044d885', 'a152e841783914146e4bcd4f39100686']
```

Para la creación de las gráficas hemos usado la librería matplotlib.pyplot.

Una vez tenemos esta columna ordenamos el dataframe en orden descendente según la criticidad e iteramos el dataframe hasta tener 10 usuarios que tienen la contraseña vulnerable.

Para la gráfica de las 5 páginas web con más políticas desactualizadas hemos añadido una columna más con la suma de las tres columnas de las políticas. Y además hemos ordenado el dataframe en orden ascendente según el año de creación porque como hay que sacar sólo 5 páginas web y puede que haya muchas con las mismas políticas desactualizadas hemos considerado que estarán más desactualizadas las que se crearon antes:

Para la tercera gráfica simplemente dividimos el total de conexiones (fechas) de los usuarios vulnerables y lo dividimos entre la cantidad de usuarios vulnerables y con los usuarios no vulnerables lo mismo:

```
def showComparativeUsers(dfUsers, vulnerables = ['5f4dcc3b5aa765d61d8327deb882cf99', '3bf1114a986ba87ed28fc1b5884fc2f8',
'276f8db0b86edaa7fc805516c852c889', '84d961568a65073a3bcf0eb216b2a576',
'0ac4539a14b3aa27deeb4cbdf6e989f', '1660fe5c81c4ce64a2611494c439e1ba',
'd16d377af76c99d27093abc22244b342', 'eb0a191797624dd3a48fa681d3061212',
'714ab9fbdad5c5da1b5d34fe1a093b79', 'd8578edf8458ce06fbc5bb76a58c5ca4',
'4297f44b13955235245b2497399d7a93', '37b4e2d82900d5e94b8da524fbeb33c0',
'd0763edaa9d9bd2a9516280e9044d885', 'a152e841783914146e4bcd4f39100686']):
    mediaVulnerna = 0
    mediaNoVulnerna = 0
    for i in dfUsers.index:
        if dfUsers['contrasena'][i] in vulnerables:
            mediaVulnerna += dfUsers['fechas'][i]
        else:
            mediaNoVulnerna += dfUsers['fechas'][i]
    mediaVulnerna /= len(vulnerables)
    mediaNoVulnerna /= (dfUsers.shape[0] - len(vulnerables))
    plt.bar(["Usuarios Vulnerables", "Usuarios No Vulnerables"], [mediaVulnerna, mediaNoVulnerna],
            color=['blue', 'orange'])
    plt.title("Comparativa")
    plt.text(0, mediaVulnerna, mediaVulnerna, ha='center')
    plt.text(1, mediaNoVulnerna, mediaNoVulnerna, ha='center')
    plt.show()
```

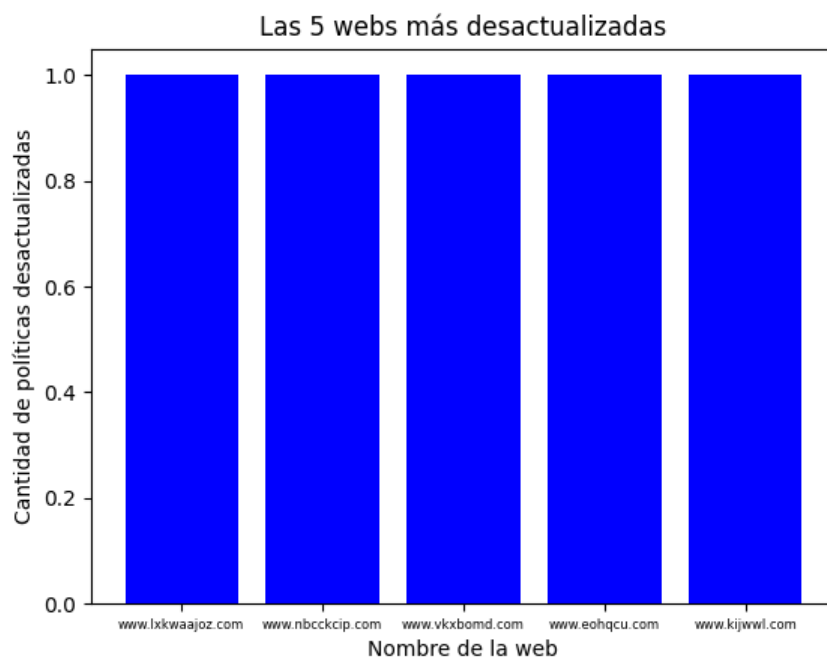
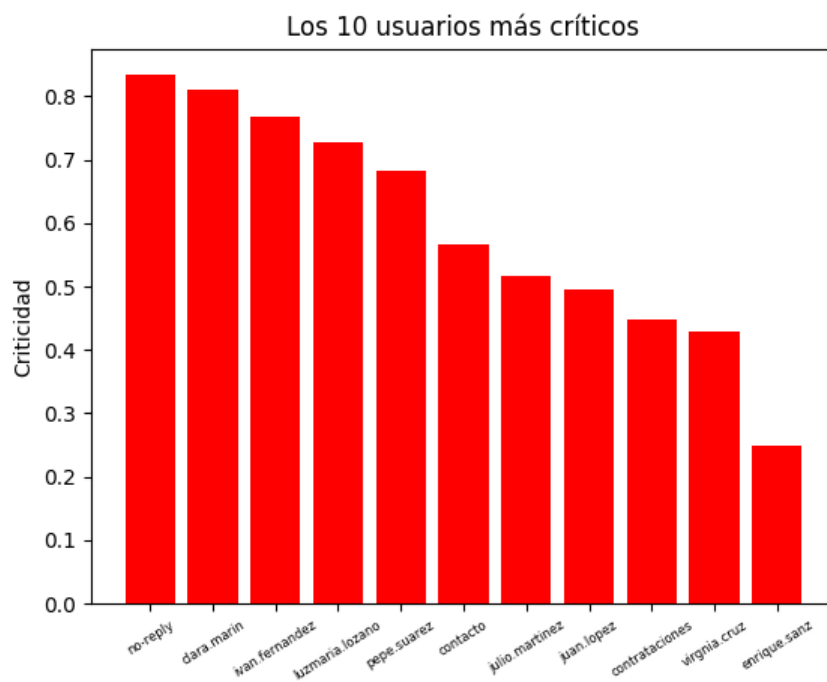
Para la tabla de las políticas de privacidad según el año, sacamos todos los años en los que se ha creado alguna web y contamos cuantas hay en cada año con política y cuantas no:

```
def showPrivacyPolicy(dfLegal):
    anos = sorted(dfLegal['creacion'].unique().tolist())
    siPoli = [0] * len(anos)
    noPoli = [0] * len(anos)
    i = 0
    for ano in anos:
        dfAux = dfLegal[dfLegal['creacion'] == ano]['proteccionDeDatos']
        for aux in list(dfAux):
            if aux == 0:
                noPoli[i] += 1
            else:
                siPoli[i] += 1
        i += 1
    ancho = 0.4
    plt.bar(anos, siPoli, ancho, label='Tiene política')
    anosb = anos.copy()
    for i in range(len(anosb)):
        anosb[i] += ancho
    plt.bar(anosb, noPoli, ancho, label='No tiene política')
    plt.legend(loc='best')
    plt.show()
```

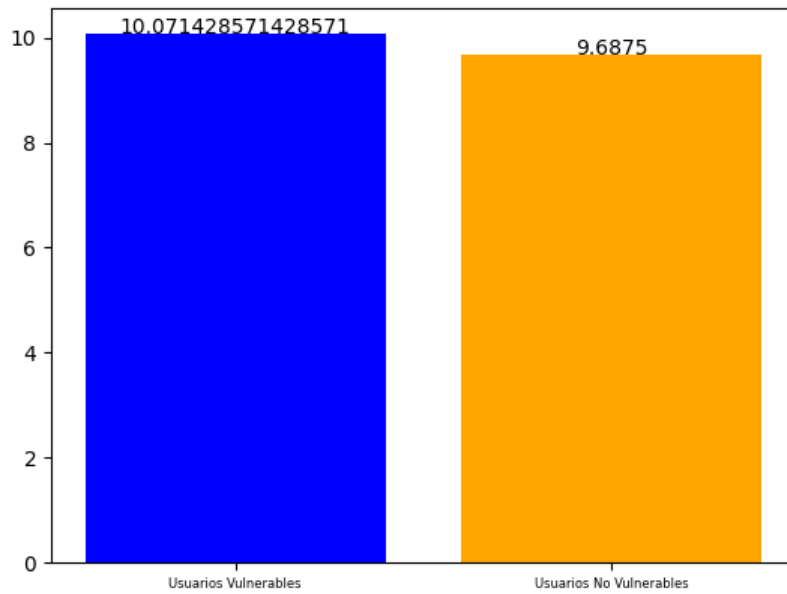
Y para la última simplemente vemos cuantas contraseñas vulnerables hay y cuantas no:

```
def showComparativePassword(dfUsers, vulnerables = ['5f4dcc3b5aa765d61d8327deb882cf99', '3bf1114a986ba87ed28fc1b5884fc2f8',
'276f8db0b86edaa7fc805516c852c889', '84d961568a65073a3bcf0eb216b2a576',
'0ac4539a14b3aa27deeb4cbdf6e989f', '1660fe5c81c4ce64a2611494c439e1ba',
'd16d377af76c99d27093abc22244b342', 'eb0a191797624dd3a48fa681d3061212',
'714ab9fbdad5c5da1b5d34fe1a093b79', 'd8578edf8458ce06fbc5bb76a58c5ca4',
'4297f44b13955235245b2497399d7a93', '37b4e2d82900d5e94b8da524fbeb33c0',
'd0763edaa9d9bd2a9516280e9044d885', 'a152e841783914146e4bcd4f39100686']):
    plt.bar(["Comprometidas", "No Comprometidas"], [len(vulnerables), (dfUsers.shape[0] - len(vulnerables))],
            color=['red', 'green'])
    plt.title("Contraseñas Comprometidas vs No Comprometidas")
    plt.show()
```

RESULTADOS:



Media de conexiones de Usuarios Vulnerables vs No Vulnerables



Número de páginas web con política de privacidad frente a las que no por año

