MEMORIA PRÁCTICA 1. SISTEMAS DE INFORMACIÓN

MARCOS BONILLA CUBERO 70068909F

PABLO GRACIA CORREA 50257261F

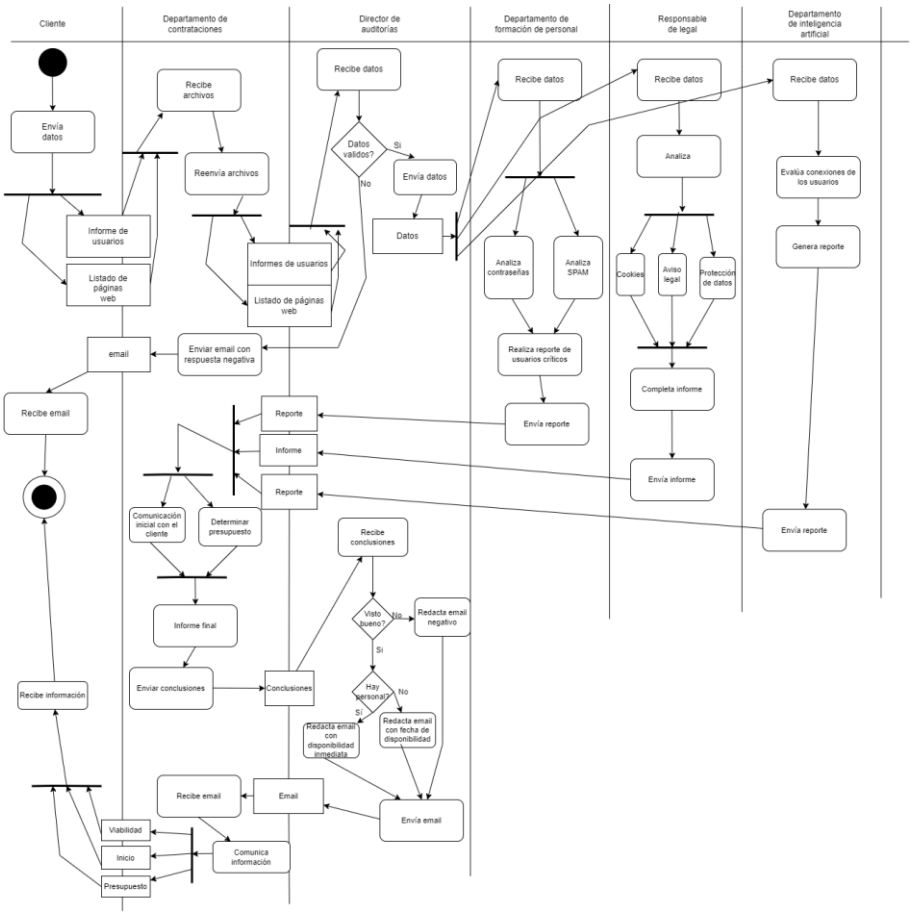
EJERCICIO 1

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.

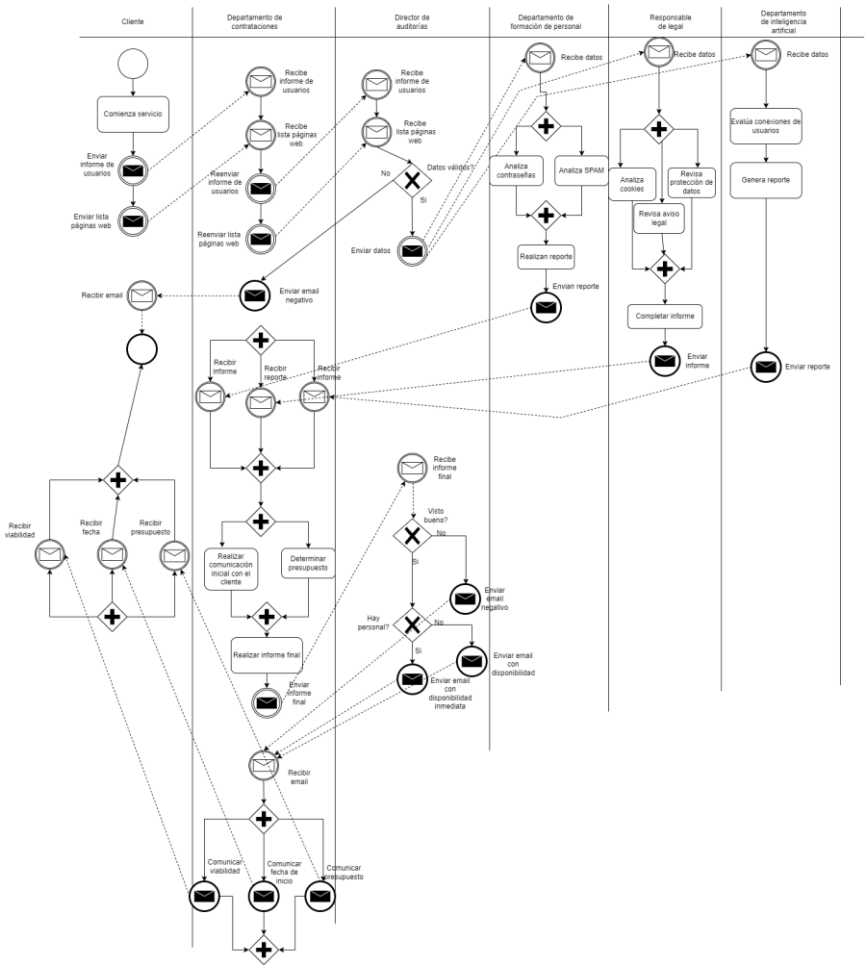
La construcción de los diagramas se ha realizado teniendo en cuenta la separación de los diferentes departamentos, sobre todo los 3 departamentos que trabajan en paralelo con los datos del director de auditorías (formación al personal, legal e IA).

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

BPMN



UML



EJERCICIO 2

Para importar los datos de los archivos json a la base de datos usaremos la librería de python “json”

```
def usersToDB():
    with open("Logs/users.json") as usersjson:
        users = json.load(usersjson)
    for user in users['usuarios']:
        insertarUsers(user)
```

```
def legalToDB():
    with open("Logs/legal.json") as legaljson:
        legal = json.load(legaljson)
    for web in legal['legal']:
        insertarLegal(web)
```

Así los leeriamos y ahora creamos las tablas e insertamos los valores uno a uno:

Users:

```python
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, phisingEmails 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']}', '{provincia}',"
                f"{(user[list(user.keys())[0]])['permisos']}, {((user[list(user.keys())[0]])['emails'])['total']},"
                f"{((user[list(user.keys())[0]])['emails'])['phishing']}, {((user[list(user.keys())[0]])['emails'])['cliclados']},"
                f"{len((user[list(user.keys())[0]])['fechas'])}, {len((user[list(user.keys())[0]])['ips'])})")
```

```python
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:

```python
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']}, {(web[list(web.keys())[0]])['aviso']},"
        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:

```python
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:

```python
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:

```python
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:

```python
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 phising de usuarios con permiso 0: " + str(dfPermisos0['clicadosEmails'].sum()))
    print("Observaciones totales de phising de usuarios con permiso 1: " + str(dfPermisos1['clicadosEmails'].sum()))
    print("Observaciones totales de phising de usuarios con 200 emails o más: " + str(dfEmailsMas['clicadosEmails'].sum()))
    print("Observaciones totales de phising de usuarios con menos de 200 emails: " + str(dfEmailsMenos['clicadosEmails'].sum()))
    print("\nTotal Valores Ausentes\n")
    print("Número de valores ausentes en phising de usuarios con permiso 0: " + str(dfPermisos0['phisingEmails'].isnull().sum()))
    print("Número de valores ausentes en phising de usuarios con permiso 1: " + str(dfPermisos0['phisingEmails'].isnull().sum()))
    print("Número de valores ausentes en phising de usuarios con 200 emails o más: " + str(dfEmailsMas['phisingEmails'].isnull().sum()))
    print("Número de valores ausentes en phising de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phisingEmails'].isnull().sum()))
    print("\nMedias:\n")
    print("Media de  emails de phising de usuarios con permiso 0: " + str(dfPermisos0['phisingEmails'].mean()))
    print("Media de emails de phising de usuarios con permiso 1: " + str(dfPermisos1['phisingEmails'].mean()))
    print("Media de emails de phising de usuarios con 200 emails o más: " + str(dfEmailsMas['phisingEmails'].mean()))
    print(
        "Media de emails de phising de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phisingEmails'].mean()))
    print("\nMedianas:\n")
    print("Mediana de emails de phising de usuarios con permiso 0: " + str(dfPermisos0['phisingEmails'].median()))
    print("Mediana de emails de phising de usuarios con permiso 1: " + str(dfPermisos1['phisingEmails'].median()))
    print(
        "Mediana de emails de phising de usuarios con 200 emails o más: " + str(dfEmailsMas['phisingEmails'].median()))
    print("Mediana de emails de phising de usuarios con menos de 200 emails: " + str(
        dfEmailsMenos['phisingEmails'].median()))
    print("\nVarianzas:\n")
    print("Varianza de emails de phising de usuarios con permiso 0: " + str(dfPermisos0['phisingEmails'].var()))
    print("Varianza de emails de phising de usuarios con permiso 1: " + str(dfPermisos1['phisingEmails'].var()))
    print("Varianza de emails de phising de usuarios con 200 emails o más: " + str(dfEmailsMas['phisingEmails'].var()))
    print("Varianza de emails de phising de usuarios con menos de 200 emails: " + str(
        dfEmailsMenos['phisingEmails'].var()))
```

```python
print("\nMaximos:\n")
print("Máximo de emails de phising de usuarios con permiso 0: " + str(dfPermisos0['phisingEmails'].max()))
print("Máximo de emails de phising de usuarios con permiso 1: " + str(dfPermisos1['phisingEmails'].max()))
print("Máximo de emails de phising de usuarios con 200 emails o más: " + str(dfEmailsMas['phisingEmails'].max()))
print(
    "Máximo de emails de phising de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phisingEmails'].max()))
print("\nMinimos:\n")
print("Mínimo de emails de phising de usuarios con permiso 0: " + str(dfPermisos0['phisingEmails'].min()))
print("Mínimo de emails de phising de usuarios con permiso 1: " + str(dfPermisos1['phisingEmails'].min()))
print("Mínimo de emails de phising de usuarios con 200 emails o más: " + str(dfEmailsMas['phisingEmails'].min()))
print(
    "Mínimo de emails de phising de usuarios con menos de 200 emails: " + str(dfEmailsMenos['phisingEmails'].min()))
```

RESULTADOS:

Total de observaciones:

Observaciones totales de phising de usuarios con permiso 0: 537

Observaciones totales de phising de usuarios con permiso 1: 1012

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

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

Total Valores Ausentes

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

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

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

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

Emails Totales:

Emails totales de phising de usuarios con permiso 0: 1277

Emails totales de phising de usuarios con permiso 1: 2003

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

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

Medias:

Media de emails dephising de usuarios con permiso 0: 79.8125

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

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

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

Medianas:

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

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

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

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

Varianzas:

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

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

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

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

Máximos:

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

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

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

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

Mínimos:

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

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

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

Mínimo de emails de phising 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 contrasenasVulnerables(dfUsers):
    contrasenas = list(dfUsers['contrasena'])
    with open("diccionarioHash.txt", "r") as dic:
        diccionario = dic.read()
    diccionario = list(diccionario.split("\n"))
    vulnerables = []
    for contrasena in contrasenas:
        if contrasena in diccionario:
            vulnerables.append(contrasena)
    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.pylot.

Para la primera gráfica que muestra los 10 usuarios más críticos primero hemos calculado la criticidad como el número de emails de spam/phishing clicados dividido entre el total de emails de phishing/spam y hemos creado un dataframe de los usuarios más la columna de criticidad.

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.

```
def showCritcUsers(dfUsers,vulnerables = ['5f4dcc3b5aa765d61d8327deb882cf99', '3bf1114a986ba87ed28fc1b5884fc2f8',
                '276f8db0b86edaa7fc805516c852c889', '84d961568a65073a3bcf0eb216b2a576',
                '0acf4539a14b3aa27deeb4cbdf6e989f', '1660fe5c81c4ce64a2611494c439e1ba',
                'd16d377af76c99d27093abc22244b342', 'eb0a191797624dd3a48fa681d3061212','714ab9fbdad5c5da1b5d34fe1a09
                '4297f44b13955235245b2497399d7a93', '37b4e2d82900d5e94b8da524fbeb33c0','d0763edaa9d9bd2a9516280e9044
    usersVulnerable = []
    usersVulnerableCriticidad = []
    dfUsersCriticos = dfUsers.copy()
    dfUsersCriticos['criticidad'] = dfUsersCriticos['clicadosEmails'] / dfUsersCriticos['phisingEmails']
    dfUsersCriticos = dfUsersCriticos.sort_values("criticidad", ascending=False)
    for i in dfUsersCriticos.index:
        if dfUsersCriticos['contrasena'][i] in vulnerables and len(usersVulnerable) <= 10:
            usersVulnerable.append(dfUsersCriticos['name'][i])
            usersVulnerableCriticidad.append(dfUsersCriticos['criticidad'][i])
    parameters = {'axes.labelsize': 10,
                  'xtick.labelsize': 6}
    plt.rcParams.update(parameters)
    plt.xticks(rotation=32)
    plt.bar(usersVulnerable, usersVulnerableCriticidad, color='red')
    plt.ylabel("Criticidad")
    plt.xlabel("Usuarios")
    plt.title('Los 10 usuarios más críticos')
    plt.show()
```

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:

```
def showOutdatedWebs(dfLegal):
    webDesactualizadas = []
    dfDesactualizados = dfLegal.copy()
    dfDesactualizados['desact'] = dfLegal.loc[:, ['cookies', 'aviso', 'proteccionDeDatos']].sum(axis=1)
    ejex = []
    ejey = []
    while len(webDesactualizadas) < 5:
        min = dfDesactualizados['desact'].min()
        dfAux = dfDesactualizados[dfDesactualizados['desact'] == min]
        dfAux = dfAux.sort_values('creacion')
        name = list(dfAux['name'].head(1))[0]
        ejex.append(name)
        ejey.append(min)
        webDesactualizadas.append(name)
        dfDesactualizados = dfDesactualizados.drop(dfDesactualizados[dfDesactualizados['name'] == name].index)
    plt.bar(ejex, ejey, color='blue')
    plt.ylabel("Cantidad de políticas desactualizadas")
    plt.xlabel("Nombre de la web")
    plt.title('Las 5 webs más desactualizadas')
    plt.show()
```

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',
                '0acf4539a14b3aa27deeb4cbdf6e989f', '1660fe5c81c4ce64a2611494c439e1ba',
                'd16d377af76c99d27093abc22244b342', 'eb0a191797624dd3a48fa681d3061212',
                '714ab9fbdad5c5da1b5d34fe1a093b79', 'd8578edf8458ce06fbc5bb76a58c5ca4',
                '4297f44b13955235245b2497399d7a93', '37b4e2d82900d5e94b8da524fbeb33c0',
                'd0763edaa9d9bd2a9516280e9044d885', 'a152e841783914146e4bcd4f39100686']):
    mediaVulnera = 0
    mediaNoVulnera = 0
    for i in dfUsers.index:
        if dfUsers['contrasena'][i] in vulnerables:
            mediaVulnera += dfUsers['fechas'][i]
        else:
            mediaNoVulnera += dfUsers['fechas'][i]
    mediaVulnera /= len(vulnerables)
    mediaNoVulnera /= (dfUsers.shape[0] - len(vulnerables))
    plt.bar(["Usuarios Vulnerables", "Usuarios No Vulnerables"], [mediaVulnera, mediaNoVulnera],
            color=['blue', 'orange'])
    plt.title("Comparativa")
    plt.text(0, mediaVulnera, mediaVulnera, ha='center')
    plt.text(1, mediaNoVulnera, mediaNoVulnera, ha='center')
    plt.show()
```

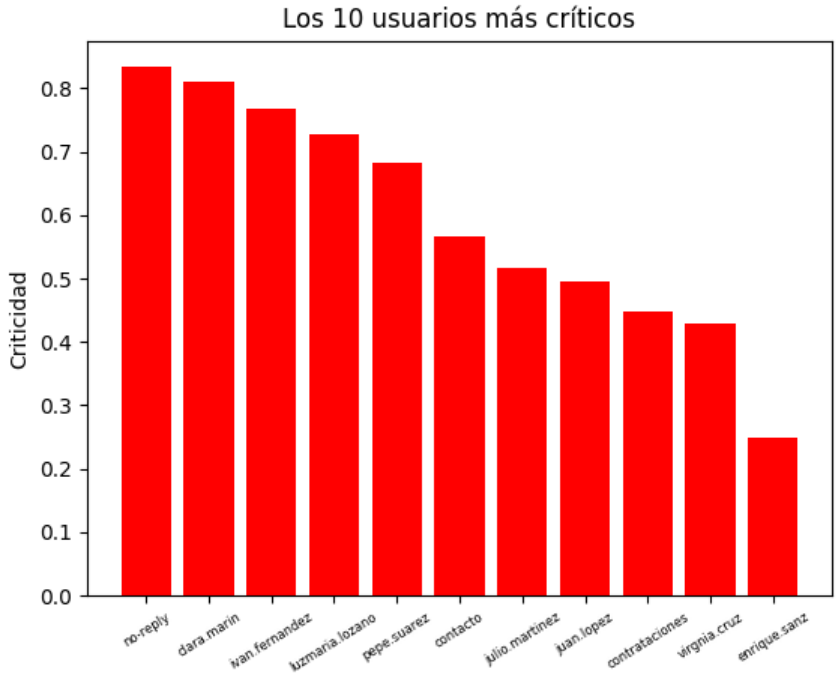
Para la tabla de las políticas de privacidad según el año, sacamos todos los años en los que se ha creado alguna web y contamos cuantas hay en cada año con política y cuantas no:

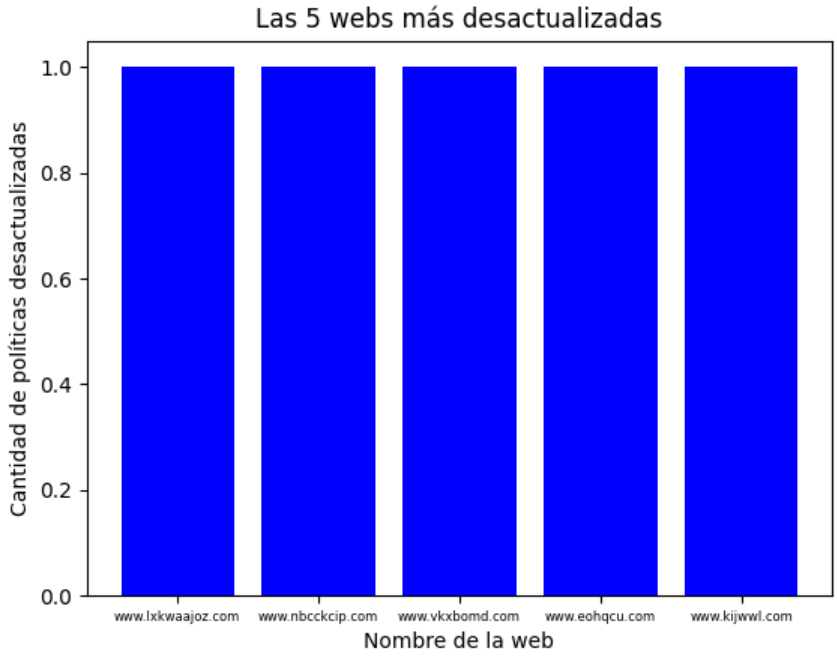
```
def showPrivacyPoliticy(dfLegal):
    anos = sorted(dfLegal['creacion'].unique().tolist())
    siPoli = [0] * len(anos)
    noPoli = [0] * len(anos)
    i = 0
    for ano in anos:
        dfAux = dfLegal[dfLegal['creacion'] == ano]['proteccionDeDatos']
        for aux in list(dfAux):
            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()
```

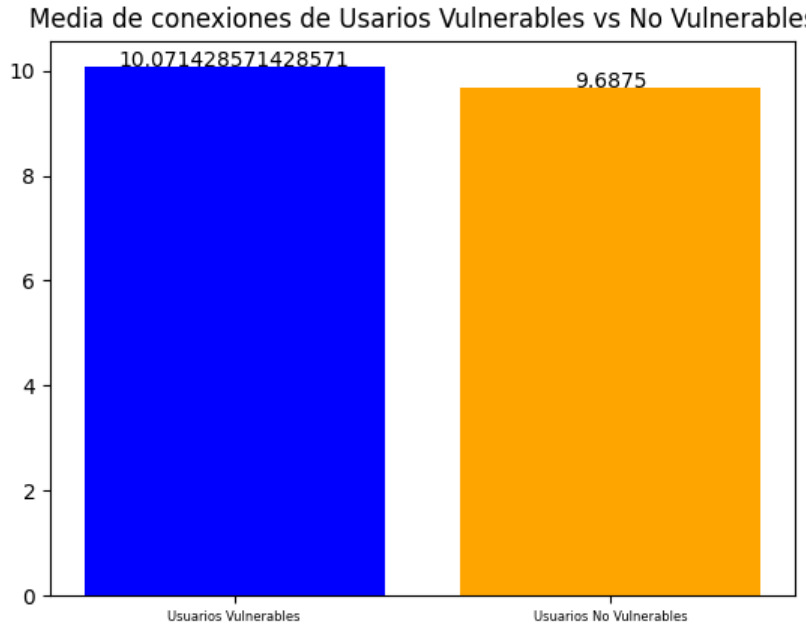
Y para la última simplemente vemos cuantas contraseñas vulnerables hay y cuantas no:

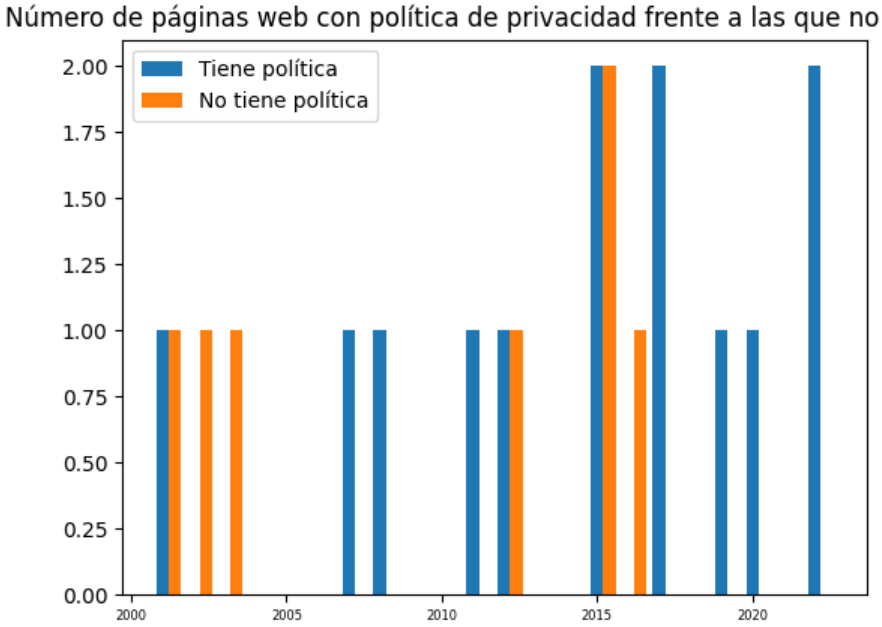
```
def showComparativePassword(dfUsers,vulnerables = ['5f4dcc3b5aa765d61d8327deb882cf99', '3bf1114a986ba87ed28fc1b5884fc2f8',
                '276f8db0b86edaa7fc805516c852c889', '84d961568a65073a3bcf0eb216b2a576',
                '0acf4539a14b3aa27deeb4cbdf6e989f', '1660fe5c81c4ce64a2611494c439e1ba',
                '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()
```

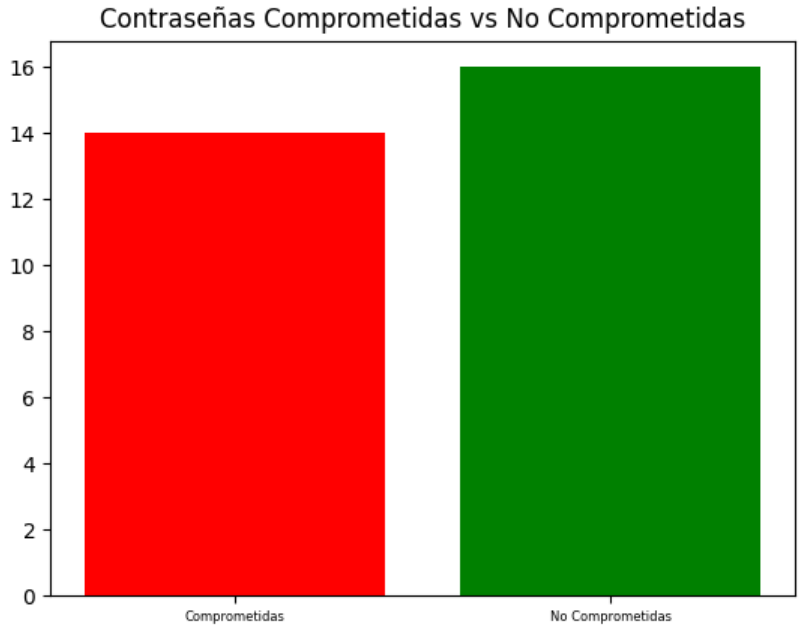
RESULTADOS:


Los 10 usuarios más críticos
Criticidad
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
no-reply
dara.marin
ivan.fernandez
luzmaria.lozano
pepe.suarez
contacto
julio.martinez
juan.lopez
contrataciones
virginia.cruz
enrique.sanz


Las 5 webs más desactualizadas
Cantidad de políticas desactualizadas
0.0
0.2
0.4
0.6
0.8
1.0
www.lxkwaajoz.com
www.nbcckcip.com
www.vkxbomd.com
www.eohqcu.com
www.kijwwl.com
Nombre de la web


Media de conexiones de Usarios Vulnerables vs No Vulnerables
10.071428571428571
9.6875
10
8
6
4
2
0
Usuarios Vulnerables
Usuarios No Vulnerables


Número de páginas web con política de privacidad frente a las que no por añ
Tiene política
No tiene política
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2000
2005
2010
2015
2020


Contraseñas Comprometidas vs No Comprometidas
16
14
12
10
8
6
4
2
0
Comprometidas
No Comprometidas