Forma

Descripción generada automáticamente con confianza media

Práctica 1: Recopilación, estructuración y análisis de datos

Autores: Alfonso Zarco Mármol

  Sergio Díaz Arias

Contenido

[Github 3](#_Toc99185753)

[Base de datos 3](#_Toc99185754)

[Ejercicio 1 4](#_Toc99185755)

[UML: 4](#_Toc99185756)

[BPMN: 4](#_Toc99185757)

[Ejercicio 2 4](#_Toc99185758)

[Ejercicio 3 5](#_Toc99185759)

[Ejercicio 4 7](#_Toc99185760)

# Github

<https://github.com/Alfonsozm/SSII>

# Base de datos

En la creación de la base de datos hemos usado en total 4 tablas, 3 para el json usuarios.json, y otra para el json legal.json.

En usuarios.json usamos tres tablas porque las fechas y las ips las almacenamos de manera individual, no como un string o como una cantidad. Los emails se almacenan como 3 columnas de la tabla principal, en vez de como una tabla externa.

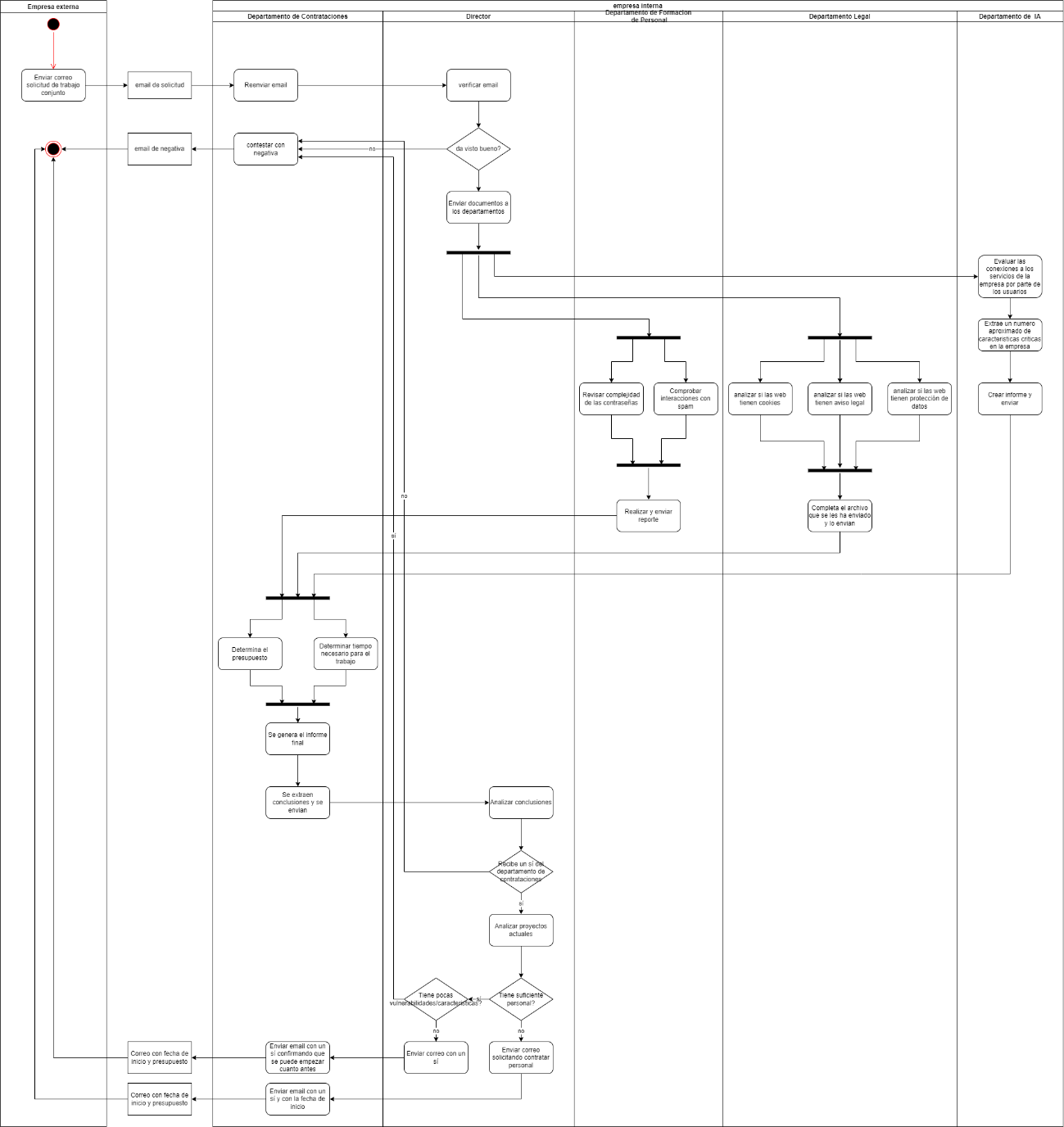
Además, antes de los CREATE TABLE contamos con sentencias DROP TABLE IF EXISTS, que permite regenerar completamente la base de datos en cada instanciación.

En la tabla de legal.json se insertan las filas con todas las columnas a la vez, en cambio en usuarios.json, en un primer lugar insertamos la PK, y posteriormente hacemos un update con cada dato por separado. Esto permite que podamos diferenciar entre datos validos y los que no lo son, haciendo más fácil la creación de la query en un primer lugar.

Los valores missing, directamente no se incluirán en la base de datos, dejando ese atributo como null.

# Ejercicio 1

## UML:



## BPMN:

# Ejercicio 2

Para generar el dataframe cargamos la tabla de usuarios completa, utilizando read\_sql\_query de pandas, y en el caso de las tablas de fechas e ips solo cargamos la cantidad de datos por usuarios usando una query de tipo SELECT COUNT utilizando además un GROUP BY. A través de ese dataframe obtenemos todos los datos necesarios utilizando muy pocas operaciones.

Los máximos, mínimos, medias y desviaciones típicas se calculan usando las funciones max(), min(), sum() y len(), y la función std() de numpy.

En cambio, para calcular los valores presentes, tomamos las dimensiones del dataframe y las multiplicamos, filas por columnas, y le restamos los valores missing, que obtenemos comprobando las columnas que podían contenerlos en el json. Los valores missing son None o NaN en el dataframe (dependiendo de si son números o no). En el caso de la columna ip debido a que es la cantidad de ips, se toma como missing si el valor es 0.

La salida al ejecutar:

Texto

Descripción generada automáticamente

# Ejercicio 3

En un primer lugar generamos 2 funciones que permitan extraer los valores de la base de datos filtrando por el tipo de permisos del usuario (0 o 1) y en función de la cantidad de correos totales recibidos.

A la función de los permisos se le introduce el permiso por el que se quiera filtrar y devuelve el dataframe con los valores filtrados.

A la función de los correos, si se le pasa un valor positivo, se devuelve el dataframe filtrado con la información de los usuarios que tengan esa cantidad o más correos totales; y si se le pasa un valor negativo, se devuelve el dataframe filtrado con la información de los usuarios que tengan menos de esa cantidad en valor absoluto de correos totales.

La media, la varianza y los valores máximos y mínimos se calculan de igual forma que en el ejercicio 2.

La mediana se calcula utilizando la función median de numpy.

Se entiende como número de observaciones como el total de correos de phishing que se han recibido.

Se entiende como valores missing como valores None o NaN.

La salida al ejecutar:

Texto

Descripción generada automáticamente

# Ejercicio 4

En primer lugar, debemos obtener qué usuarios son críticos, para esto es necesario comprobar si sus contraseñas son débiles y si tienen mayor probabilidad de pulsar en correo de spam.

Para comprobar si una contraseña es débil, hemos utilizado carckstation.net, de donde hemos descargado una wordlist de contraseñas débiles, las hemos hasheado y hemos comprobado cuales de los hashes de las contraseñas de los usuarios estaban en ese archivo (el archivo weak\_pass.txt contiene los hashes que coinciden con alguno de los de los usuarios, el otro archivo es demasiado grande para github o el aula virtual).

Para calcular la probabilidad de pulsar en correo de spam, se divide el número de veces que un usuario ha pulsado en un correo de spam entre la cantidad de correos de spam que ha recibido.

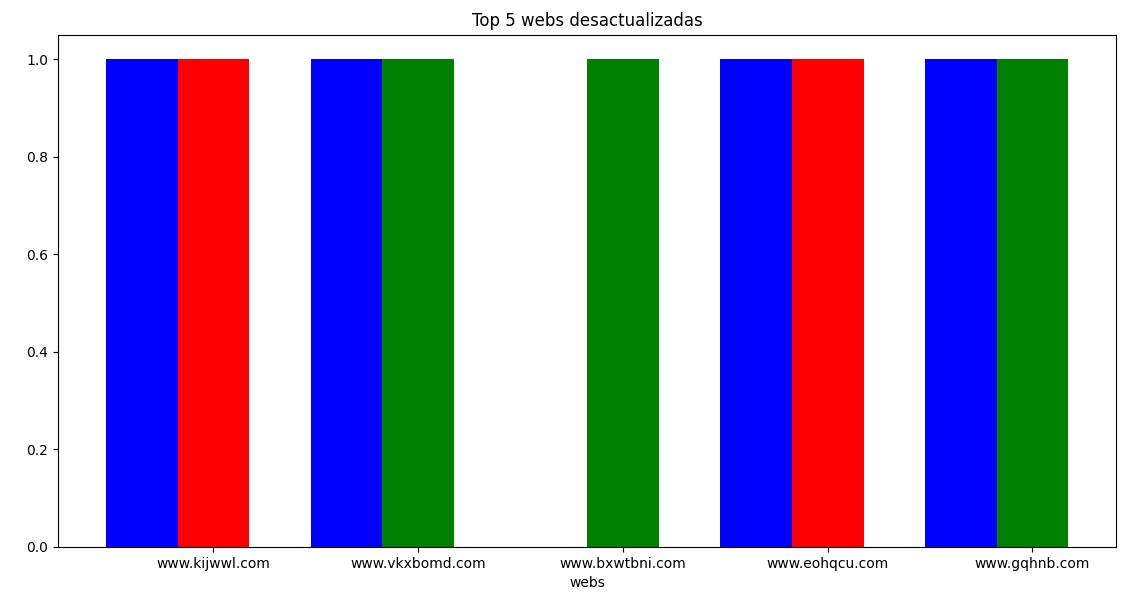
Gráfico de barras representando a los usuarios y su probabilidad de clicar un email de phishing:

Gráfico, Gráfico de barras

Descripción generada automáticamente

Para obtener las webs con más políticas desactualizadas, extraemos el número total de políticas desactualizadas, y ordenamos en base a ese número.

El gráfico de barras según las políticas (verde-protección de datos, rojo-aviso legal, azul-cookies):



Para extraer la media de conexiones de usuarios con contraseña vulnerable, se filtra el dataframe y se calcula la media como anteriormente se ha explicado en el ejercicio 2.

Para mostrar el año de creación de las webs que no cumplen las políticas se filtra el datafrase, y se extraen los datos necesarios.

Para mostrar el número de contraseñas comprometidas y no comprometidas (se entiende por comprometida, una contraseña débil, es decir, que ya forme parte de un wordlist) se filtra el datafrase según esos dos parámetros.

La salida después de ejecutarse (los gráficos anteriormente mostrados también forman parte de la salida):

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media