Universidad ORT Uruguay
Facultad de Ingeniería

Diseño de Aplicaciones 1

Primer Obligatorio

Cristian Palma 208443

Federico Alonso 182999

Gestor de Contraseñas

Docentes: Leonardo Cecilia, Bruno Balduccio.

Grupo N5A

Repositorio: https://github.com/ORT-DA1/208443_182999

Índice

Descripción general del trabajo y del sistema.

Objetivo de la aplicación

El objetivo de esta aplicación es permitir al usuario registrar sus contraseñas e información de tarjetas de crédito de forma segura mediante un gestor de contraseñas, para que el usuario sólo tenga que memorizar una clave para acceder a todas las demás.

El sistema cuenta de distintas secciones dentro de las cuales se pueden realizar acciones tales como altas, bajas, modificaciones, mostrar resumen y vulnerabilidades de los datos guardados.

Las secciones son:

- Categorías
- Contraseñas
- Tarjetas de crédito
- Análisis de vulnerabilidades

Las categorías son ingresadas por el usuario, solamente requiere un nombre y luego vamos a poder almacenar contraseñas y/o tarjetas de crédito asociando las mismas a una determinada categoría previamente registrada.

El sistema permite almacenar contraseñas, para agregar una se pide la categoría, un sitio, un nombre de usuario, la contraseña y una nota opcional. La contraseña podrá ser ingresada manualmente por el usuario o generada por la aplicación. Además, se brinda un reporte de la fortaleza de las mismas, las cuales son clasificadas por colores en base a su complejidad.

Otra funcionalidad es almacenar tarjetas de crédito, para agregar una nueva se pide categoría, un nombre, tipo, los 16 números de la tarjeta, el código de seguridad, mes/año de vencimiento y una nota opcional.

Dentro de la sección de análisis de vulnerabilidades tenemos el chequeo de las tarjetas de crédito y/o contraseñas que se encontraron expuestas en data breaches. En esta primera entrega se brinda solamente la opción de cargar esos datos expuestos de forma manual.

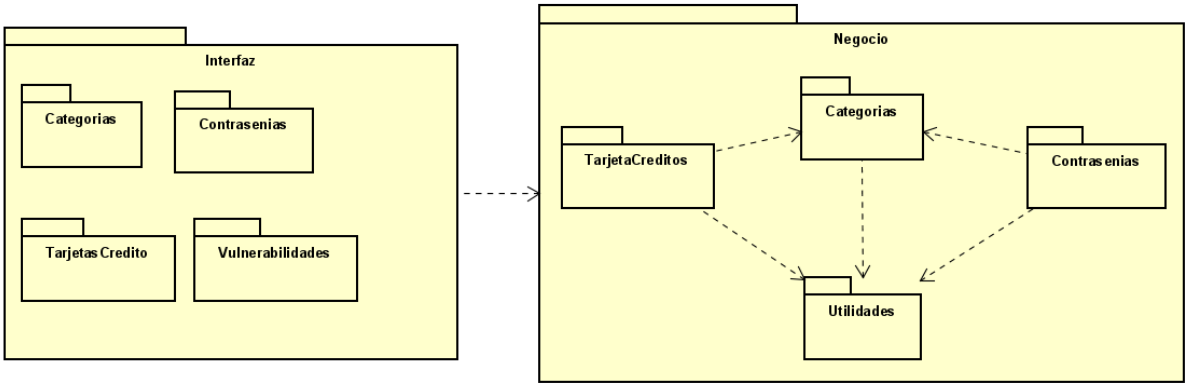
Errores conocidos

- Existen diferencia en la lógica de creación de categoría con respecto a las demás clases. Para crear categoría enviamos por parámetro los atributos de la misma, en cambio para crear clases más complejas, enviamos el objeto. La diferencia fue evidente al momento de confeccionar los diagramas y no contamos con tiempo de modificarlo.

Descripción y justificación de diseño

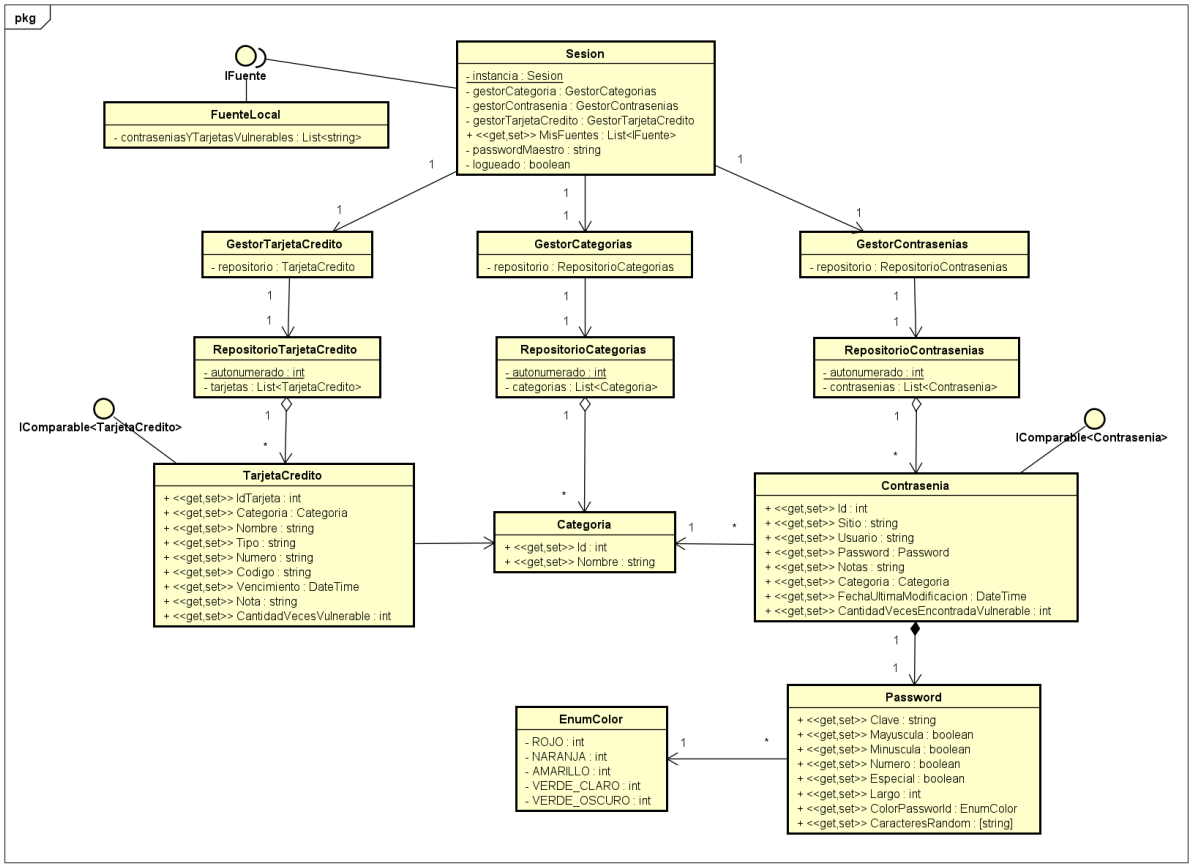
Diagrama de paquetes

El diagrama de paquetes corresponde a la forma en la que están distribuidas las clases del sistema. En primera instancia se separan en dos grandes paquetes (Interfaz y Negocio), y dentro de ellos se vuelve a subdividir como se muestra a en la siguiente imagen:



Diagramas de clases

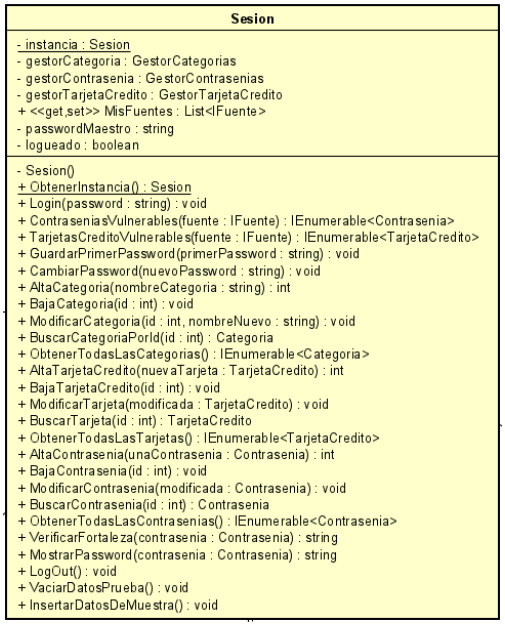
A continuación, se muestra un diagrama de clases simplificado, donde se observan los atributos de las clases principales del dominio y la interacción entre las mismas. Se simplifica el diagrama quitándole las excepciones y los métodos a efectos de que sea legible, se adjunta como **Anexo_1** el diagrama de clases completo.



Interacción de la Interfaz con el Dominio

En el sistema realizamos una separación entre la interfaz y la capa de negocio, su único vínculo es la clase Sesión que actúa de **"fachada"** ya que la misma no contiene lógica, si no que recibe los requerimientos de la interfaz y los deriva a la clase que corresponda. Esta decisión fue tomada al momento de hacer clean code y aplicar la ley de Demeter.

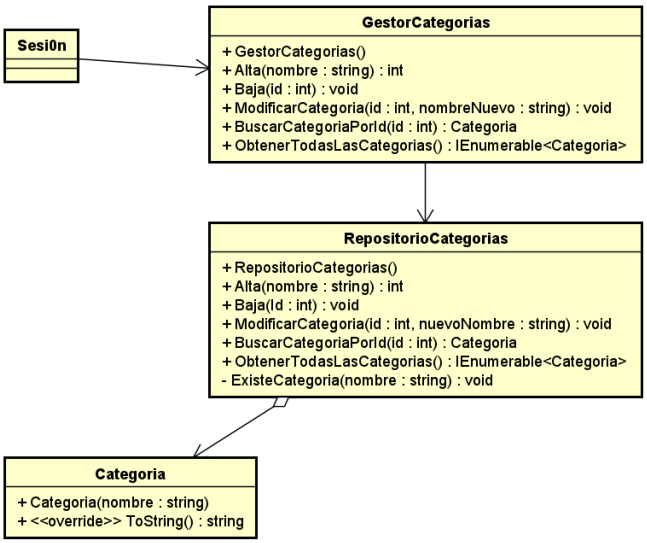
A continuación, se muestra la clase Sesión con sus atributos y métodos.



Almacenamiento de datos

El almacenamiento de datos en esta etapa se hace mediante listas en memoria, cuando se termina la ejecución de la aplicación la información no persiste. A pesar de esto, se diseñó la aplicación pensando que esta situación iba a cambiar a futuro, entonces la lógica específica para la manipulación de listas se encapsuló en las clases repositorios de cada entidad (Categorías, Contraseñas y Tarjetas).

De esta forma los gestores desconocen la lógica de guardar la información, facilitando el cambio de la forma de persistencia en una segunda etapa del proyecto, a continuación, se muestra un ejemplo de la clase categoría.



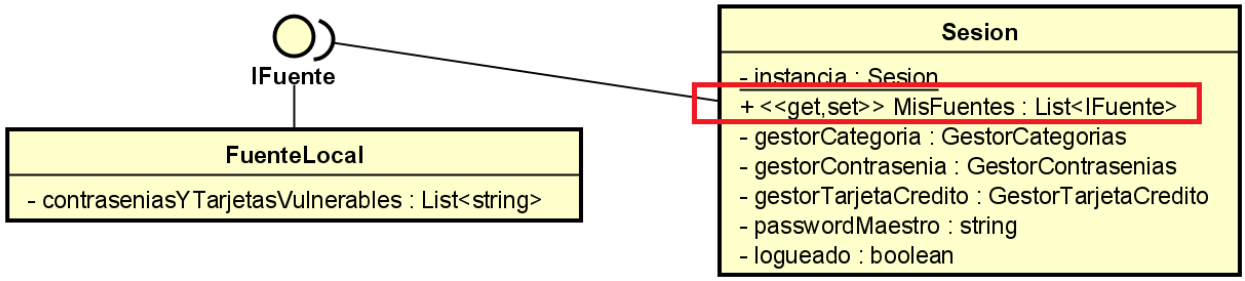
Diseño para chequear Data Breaches

Siendo que por el momento se cuenta con una fuente local, y que en el futuro se puede contar con otro tipo de fuente, se decide implementarla a través de una interfaz **IFuente.**

Esta interfaz posee la firma de los métodos necesarios para encontrar contraseñas y/o tarjetas de créditos vulnerables y retornarnos cuantas veces fueron halladas vulnerables.

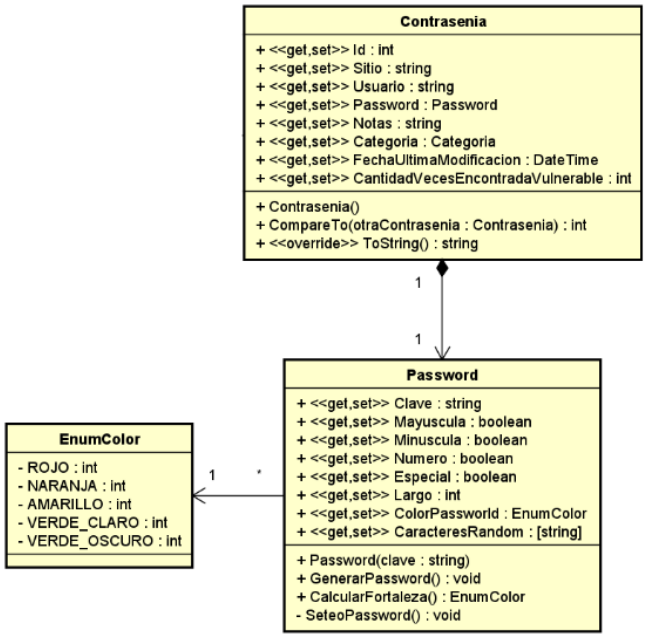
Actualmente solo se cuenta con una fuente sola, la cual llamamos "**FuenteLocal"**, que consiste en una lista de strings. Esta clase **Implementa** la interfaz IFuente y pone lógica a los métodos, luego la clase **Sesión Usa** la interfaz ya que cuenta con una lista de elementos IFuentes.

De esta forma a futuro si nos agregan otra fuente, no tenemos que modificar el código existente, solamente debemos agregamos una clase nueva que implemente la interfaz IFuente y gracias al polimorfismo podemos mandar el mismo mensaje (Ej buscar contraseña y/o tarjeta en fuente) a cualquier objeto de la List<IFuente>.



Diseño de la Clase Password

Al finalizar las funcionalidades básicas, nos dimos cuenta de que la clase correspondiente al guardado de contraseñas contenía demasiado lógica, por lo que se decidió crear una clase Password, la cual es la encargada de la lógica correspondiente a la clave de la contraseña, auto creación del mismo, reporte de fortalezas, etc.



Manejo de errores y excepciones

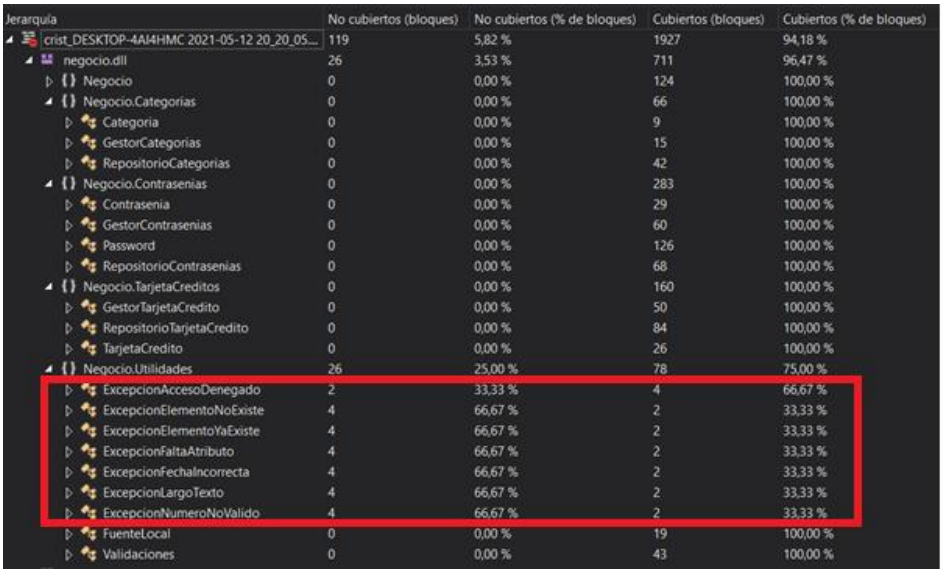
Las excepciones en la aplicación fueron creadas de forma genérica, por ejemplo, "**ExcepcionLargoTexto"**, "**ExcepcionElementoFaltante"**, etc.

Las mismas son utilizadas para todas las clases en las que se realizan validaciones, diferenciando en ellas un mensaje personalizado de acuerdo con el error especifico que la haya disparado. De esta forma evitamos que exista un paquete repleto de excepciones ocurridas por cada atributo controlado. Las excepciones son controladas por la interfaz, quien tiene control de estas, mostrándole al usuario de forma oportuna cuando alguna de ellas ocurre.

Se decide "trimar" los campos de texto, a efecto de que los caracteres espacio en blanco no ocupen mayor lugar del necesario, esto no se toma en cuenta para los "*password*", debido a que se decidió en la letra del foro de que el espacio en blanco fuera tomado como carácter especial.

Cobertura de pruebas unitarias

La cobertura de las pruebas unitarias en el proyecto negocio es del 96,47%, incluyendo la reducción de cobertura de las clases excepción que se explicará a continuación.



Jerarquía	No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
crist_DESKTOP-4AI4HMC 2021-05-12 20_20_05...	119	5,82 %	1927	94,18 %
negocio.dll	26	3,53 %	711	96,47 %
Negocio	0	0,00 %	124	100,00 %
Negocio.Categorias	0	0,00 %	66	100,00 %
Categoria	0	0,00 %	9	100,00 %
GestorCategorias	0	0,00 %	15	100,00 %
RepositorioCategorias	0	0,00 %	42	100,00 %
Negocio.Contrasenias	0	0,00 %	283	100,00 %
Contrasenia	0	0,00 %	29	100,00 %
GestorContrasenias	0	0,00 %	60	100,00 %
Password	0	0,00 %	126	100,00 %
RepositorioContrasenias	0	0,00 %	68	100,00 %
Negocio.TarjetaCreditos	0	0,00 %	160	100,00 %
GestorTarjetaCredito	0	0,00 %	50	100,00 %
RepositorioTarjetaCredito	0	0,00 %	84	100,00 %
TarjetaCredito	0	0,00 %	26	100,00 %
Negocio.Utilidades	26	25,00 %	78	75,00 %
ExcepcionAccesoDenegado	2	33,33 %	4	66,67 %
ExcepcionElementoNoExiste	4	66,67 %	2	33,33 %
ExcepcionElementoYaExiste	4	66,67 %	2	33,33 %
ExcepcionFaltaAtributo	4	66,67 %	2	33,33 %
ExcepcionFechaIncorrecta	4	66,67 %	2	33,33 %
ExcepcionLargoTexto	4	66,67 %	2	33,33 %
ExcepcionNumeroNoValido	4	66,67 %	2	33,33 %
FuenteLocal	0	0,00 %	19	100,00 %
Validaciones	0	0,00 %	43	100,00 %

Los casos que no llegan al 90% de cobertura corresponden a las clases creadas para manejar nuestras propias Excepciones como se muestra en la siguiente foto:

```
1   using System;
2
3   namespace Negocio.Utilidades
4   {
        41 referencias | federicoalonso, Hace 1 día | 1 autor, 1 cambio
5       public class ExcepcionAccesoDenegado : Exception
6       {
            1 referencia | federicoalonso, Hace 1 día | 1 autor, 1 cambio
7           public ExcepcionAccesoDenegado() : base() { }
8
            17 referencias | federicoalonso, Hace 1 día | 1 autor, 1 cambio
9           public ExcepcionAccesoDenegado(string message) : base(message) { }
10
            0 referencias | federicoalonso, Hace 1 día | 1 autor, 1 cambio
11          public ExcepcionAccesoDenegado(string message, Exception innerException) : base(message, innerException) { }
12      }
13  }
14
```

Todas las excepciones creadas, cuentan con esos tres constructores, pero en esta etapa del obligatorio no fue necesario utilizar los mismos y esto explica la reducción de la cobertura de las pruebas.

Documentación de caos de prueba elaborados

A continuación, se deja enlaces de Youtube a videos donde se muestra el uso de la aplicación en lo referente a alta, baja y modificación de contraseñas, abarcando los siguiente casos:

1. Video 1: https://youtu.be/ssi-2tK_A_Y

- Intento de guardar una contraseña sin categoría.
- Error al intentar guardar una contraseña sin sitio.
- Error al intentar guardar una contraseña con un sitio con 2 caracteres.
- Error al intentar guardar una contraseña con un sitio con más de 25 caracteres.
- Error al intentar guardar una contraseña sin usuario.
- Error al intentar guardar una contraseña con un usuario con 4 caracteres.
- Error al intentar guardar una contraseña con un usuario con más de 25 caracteres.
- Error al intentar guardar una contraseña sin usuario.
- Error al intentar guardar una contraseña con un password con 4 caracteres.
- Error al intentar guardar una contraseña con un password con más de 25 caracteres.
- Error al intentar guardar una contraseña con un notas con más de 250 caracteres.
- Éxito al guardar una contraseña.
- Error al intentar guardar una contraseña repetida.
- Éxito al autogenerar los passwords, se verifican los caracteres solicitados y que se muestran alternados.

2. Video 2: https://youtu.be/6W3ClEO5Sag

- Listado de contraseñas correcto.
- Se puede ver la contraseña por 30 segundos.
- Se puede modificar la contraseña desde el listado.
- Se puede eliminar la contraseña desde el listado.
- Se muestran las contraseñas ordenadas por categoría.

3. Video 3: https://youtu.be/d-se0gdgjJE

- Se puede modificar una contraseña correctamente.
- No se modifica la fecha al modificar un atributo menos el password.
- Se modifica correctamente la fecha al modificar el password.
- Error al intentar guardar una contraseña con un sitio con 2 caracteres.
- Error al intentar guardar una contraseña con un sitio con más de 25 caracteres.
- Error al intentar guardar una contraseña sin usuario.
- Error al intentar guardar una contraseña con un usuario con 4 caracteres.
- Error al intentar guardar una contraseña con un usuario con más de 25 caracteres.
- Error al intentar guardar una contraseña sin usuario.
- Error al intentar guardar una contraseña con un password con 4 caracteres.
- Error al intentar guardar una contraseña con un password con más de 25 caracteres.
- Error al intentar guardar una contraseña con un notas con más de 250 caracteres.

4. Video 4: https://youtu.be/pZuAKVFFm4g

- Se puede eliminar una contraseña.
- Se muestra correctamente el reporte de fortaleza.
- Se pueden ver todas las contraseñas de una misma fortaleza.
- Se puede modificar la contraseña y esta cambia de fortaleza.