

# Ingeniería de software

Alumnos: Gaston Rousseau y Renata Vidal

Materia: Ingeniería en Software

Docente: Julián Martin Rodríguez Escobedo

Universidad: UAI

Localización: Castelar

Comisión: 3A

Turno: Noche

# Índice

<b>T01 Arquitectura del Sistema</b>	<b>2</b>
<b>T02 Gestión de Log in/Log out del Sistema</b>	<b>6</b>
<b>T03 Gestión de encriptado</b>	<b>10</b>
<b>T04 Gestión de perfiles de usuario</b>	<b>11</b>
<b>T05 Gestión de múltiples idiomas</b>	<b>17</b>
<b>T06 Gestión de bitácora y control de cambios</b>	<b>19</b>
<b>T07 Gestión de dígito verificador</b>	<b>22</b>

# **T01 Arquitectura del Sistema**

¿Qué es?

La arquitectura del sistema indica la estructura, funcionamiento e interacción entre las partes del software.

La arquitectura en capas es un enfoque de diseño de software que separa la aplicación en capas lógicas y funcionales. Cada capa tiene un conjunto de responsabilidades y tareas específicas por cumplir, lo que permite poder realizar una mejor organización del código y una mayor facilidad de mantenimiento.

¿Cómo está conformado nuestro sistema?

Nuestro sistema se divide en capas. En otras palabras, está compuesto por subsistemas. Este sistema está compuesto por 7 capas, las cuales son UI, BE, BLL, MPP, DAL, Servicios y Abstracción.

Además de que nuestro sistema está conectado a una base de datos que nos permite almacenar, recuperar y gestionar información. Trayéndonos ventajas como son la eficiencia, seguridad, accesibilidad y escalabilidad.

Esta se conecta al sistema a través de la DAL ya que esta cumple un rol de intermediaria. Proporcionándonos un sistema consistente y unificado capaz de acceder a los datos almacenados en la base de datos.

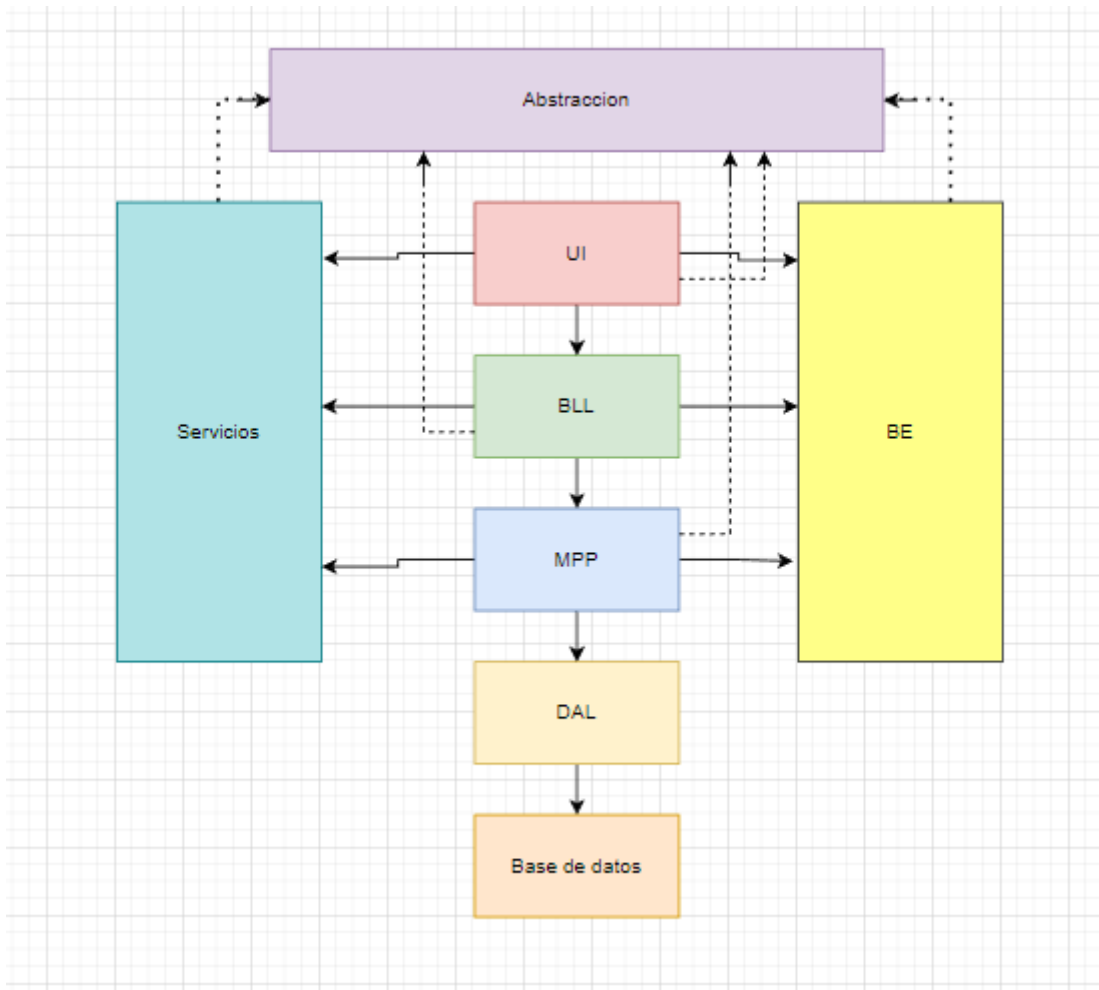


Diagrama de la arquitectura del sistema

Estas capas se conectan a través de interfaces y servicios, lo que permite a cada capa comunicarse con las capas cercanas de manera segura y eficiente.

La capa de usuario(UI): Esta capa se encarga de presentar la interfaz gráfica de usuario y procesar los datos de entrada de usuario. Esta capa se comunica con la BE, la BLL y la capa de servicios.

La capa de entidades de negocio(BE): esta capa se encarga de representar las entidades y objetos de negocio del sistema. Esta define cómo se modelan estas entidades y cómo se relacionan entre sí. Está conectada con la UI, la BLL, el MPP y la capa de Abstracción

La capa de lógica de negocios(BLL): esta capa contiene la lógica del negocio y las reglas con las que definen cómo se deben procesar los datos. Esta capa se comunica con la BE, el MPP, la capa de abstracción, la de servicios y la UI.

La capa de procesamiento de Datos(MPP): esta capa se utiliza para procesar los datos antes de que se envíen a la DAL o después de recibirlos de la misma. El MPP se comunica con la DAL, la BLL, la BE, la capa de abstracción y la capa de servicios

La capa de acceso a datos (DAL): esta capa se utiliza para acceder a la base de datos y realizar las operaciones de escritura y de lectura. La DAL se comunica con el MPP y la base de datos.

La capa de servicios se encarga de proveer todo tipo de servicios al sistema, un ejemplo de estos servicios es la seguridad que es utilizada por ejemplo para el encriptado de la contraseña.

La capa de Abstracción es una interfaz en donde declaramos una propiedad que es "id" que se utiliza para definir las característica básica de una entidad genérica. Esta capa está conectada con la BE.

La Base de datos se compone de tablas, que a su vez está compuesto por columnas y filas. Los usuarios pueden realizar consultas en la base de datos para buscar y recuperar información específica. Es fundamental ya que nos permite almacenar, recuperar y gestionar información del sistema.

#### Justificación de elección de arquitectura:

Elegimos realizar el sistema con este tipo de arquitectura porque separa claramente las responsabilidades del sistema en capas separadas, lo que en un futuro hace fácil el mantenimiento y la actualización del sistema, logrando un sistema mucho más organizado y más sencillo a la comprensión, logrando ser muy atractivo a la vista a comparación de otras arquitecturas con menos cantidad de capas. Además de que esta arquitectura es escalable a largo plazo.

También esta arquitectura es muy útil para que en el futuro cuando planteemos la propuesta de proyecto nos permita cumplir con los requisitos del sistema y del negocio.

A la arquitectura del sistema le agregamos las capas de Mapper y de abstracción debido a que permite integrar diferentes sistemas, sin afectar la lógica del negocio. Nos resultan bastante útiles ya que nos ayuda a separar la lógica de negocio de los detalles de implementación, oculta la complejidad, mejora la reutilización de componentes y facilita la integración.

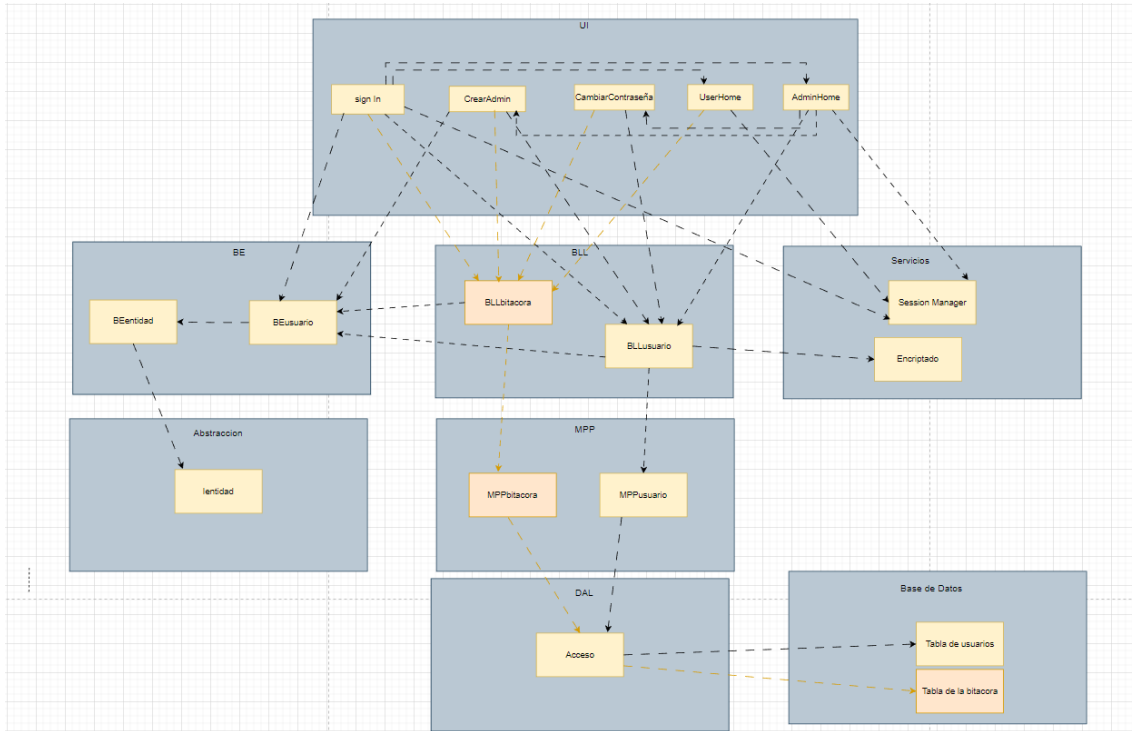


Diagrama de componentes de sign in

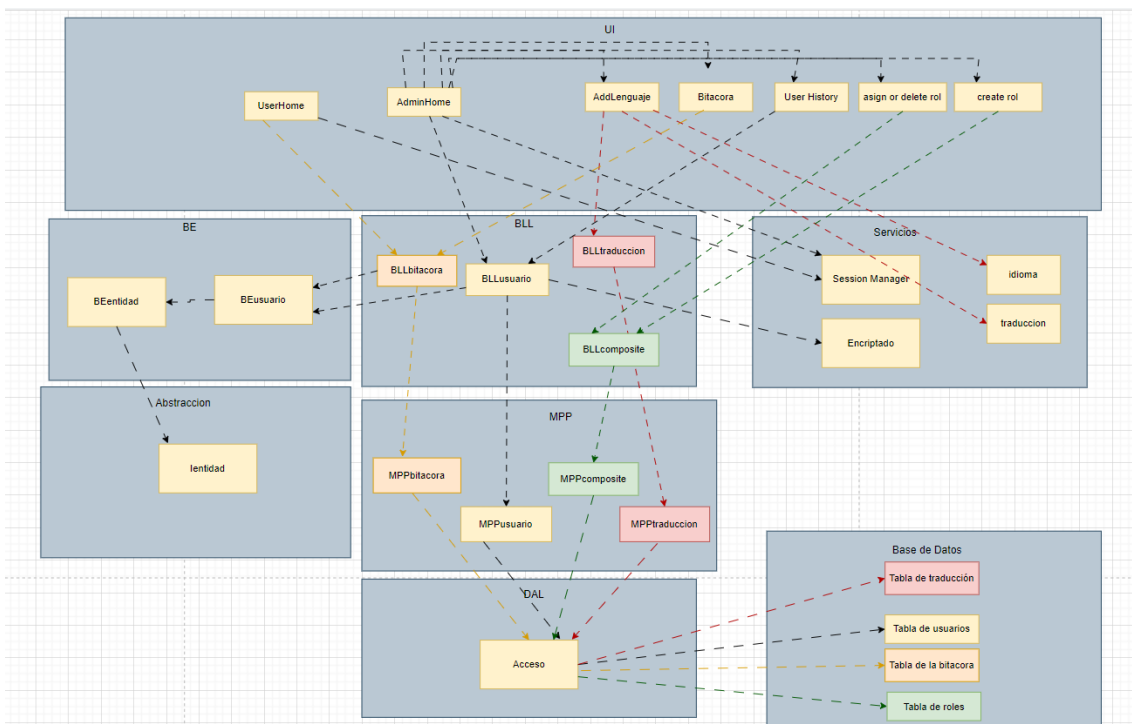
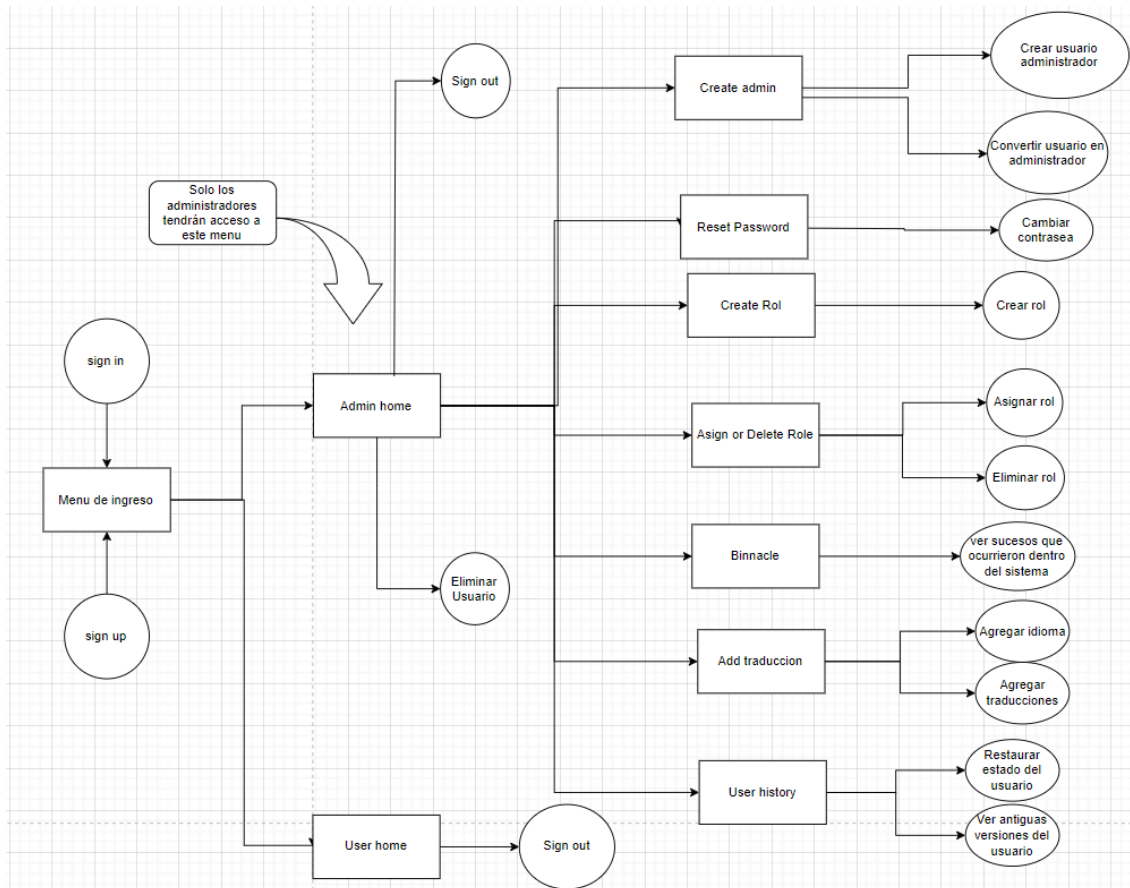


Diagrama de componentes dentro de Admin home



Mapa tentativo de navegación de los menús del sistema

## T02 Gestión de Log in/Log out del Sistema

### ¿Qué es?

El log in es un proceso que se utiliza en el sistema para autenticar al usuario y permitirles acceder tanto a los recursos como a los servicios que se encuentran protegidos por contraseña. Es una medida de seguridad que protege la privacidad de los datos y evita el acceso no autorizado al sistema.

El log out por otra parte es un proceso que permite a un usuario finalizar su sesión actual del sistema. Cuando un usuario inicia sesión en el sistema, se le asigna un conjunto de credenciales que le permiten acceder a ciertas funciones y recursos. Al cerrar sesión o hacer Log out, el usuario finaliza su sesión activa y se desconecta del sistema.

### ¿Cómo funciona?

Cuando el usuario quiera acceder al sistema se le pedirá que ingrese su credencial de usuario y contraseña a través de un formulario. Luego el sistema autentica la identidad del usuario comparando sus credenciales con las que están almacenadas en la base de datos. Además, que

en el proceso de autenticación ocurrirá un proceso de encriptación de la contraseña para aumentar la seguridad del sistema. Esto ocurre gracias a los servicios del sistema, que es una de las capas de la arquitectura o subsistema del sistema.

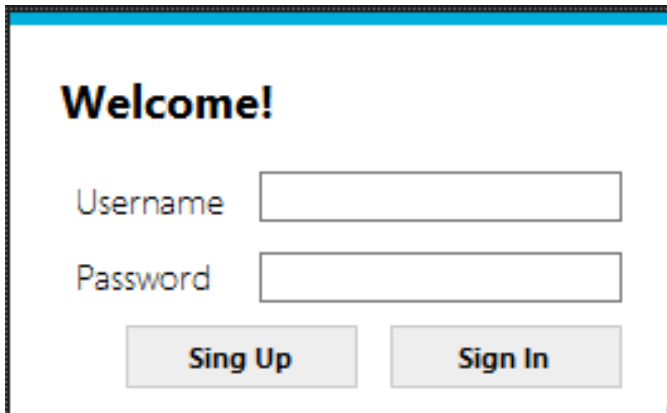
The image shows a login form within a window. At the top, it says "Welcome!". Below that, there are two input fields: "Username" and "Password". Under the "Password" field, there are two buttons: "Sing Up" and "Sign In". The form is enclosed in a simple rectangular border.

Imagen del formulario de log in

#### Explicación del código:

Para realizar la gestión de log in y log out , utilizamos el patrón de diseño singleton, tiene como objetivo asegurar de que solo haya una instancia de una clase en todo el programa.

Este patrón se encuentra dentro de la capa de servicios. Esta clase tiene dos métodos estáticos, "log in" y "log out".

El método "log in" se utiliza para iniciar la sesión. Si "session" es nulo, se creará una nueva instancia de la clase y se almacenará en "session". Además, se establecerá el usuario y la hora de inicio de sesión. Si "sesión" no es nulo, se lanzará una excepción indicando que la sesión ya se ha iniciado.

El método "log out" se utiliza para cerrar la sesión. Si "session" no es nulo, se establecerá en nulo. Si "session" es nulo, se lanzará una excepción indicando que la sesión no se ha iniciado.

Para validar al usuario creamos una clase dentro del mapper, esta tiene dos métodos.

El primer método se llama "validar" y recibe un objeto de tipo BEUsuario, perteneciente a la capa BE. Este método se encarga de verificar si el usuario y contraseña que proporciona el objeto coinciden con algún usuario almacenado en una tabla de la base de datos. Para lograr eso, el método utiliza una instancia de la clase "Acceso" que representa una conexión a la base de datos. Este "Acceso" se encuentra dentro de la capa DAL.

El método crea una instancia de la clase DataTable para poder almacenar los resultados de la consulta que se le hizo a la base de datos. Después utiliza la instancia de "Acceso" para leer los



datos de la tabla de usuarios llamando al método “leer” con el nombre de la consulta “s\_usuario\_listar”. Lo que devuelve esta consulta se guarda en el DataTable que se creó anteriormente.

El método utiliza un Foreach para recorrer todas las filas del DataTable. Se compara los valores del usuario y contraseña con los valores de la fila en la tabla. En caso de que haya una coincidencia el método retorna un True, en caso contrario retorna un False.

Dentro del UI, creamos un formulario con los controles necesarios para que el usuario sea capaz de ingresar el usuario y la contraseña.

En el código del formulario se crea una instancia de la clase “BLLusuario” y una instancia de la clase “BEUsuario” para interactuar con los datos del usuario y llegar hasta el MPP, debido a que la BLL está conectada con el mismo.

El evento “metroButton1\_Click” se desencadena cuando se presiona el botón iniciar sesión dentro del formulario. Aquí es donde se verifican si se han ingresado los datos válidos en los campos de usuario y contraseña.

El evento comprueba si los campos de usuario y contraseña están vacíos, si esto ocurre se lanza una excepción y se muestra un mensaje de error.

En el caso de que los campos no estén vacíos, el evento crea un objeto de la clase BE usuario.

Aquí es donde el objeto toma los valores de los Textbox y a su vez son encriptados gracias al método “Encriptar” de la clase de Servicios.

Posteriormente se llama al método validar de la instancia “oLog” de la clase BLLUsuario para verificar si las credenciales son válidas. En caso de que las credenciales no sean válidas, se muestra un mensaje de error.

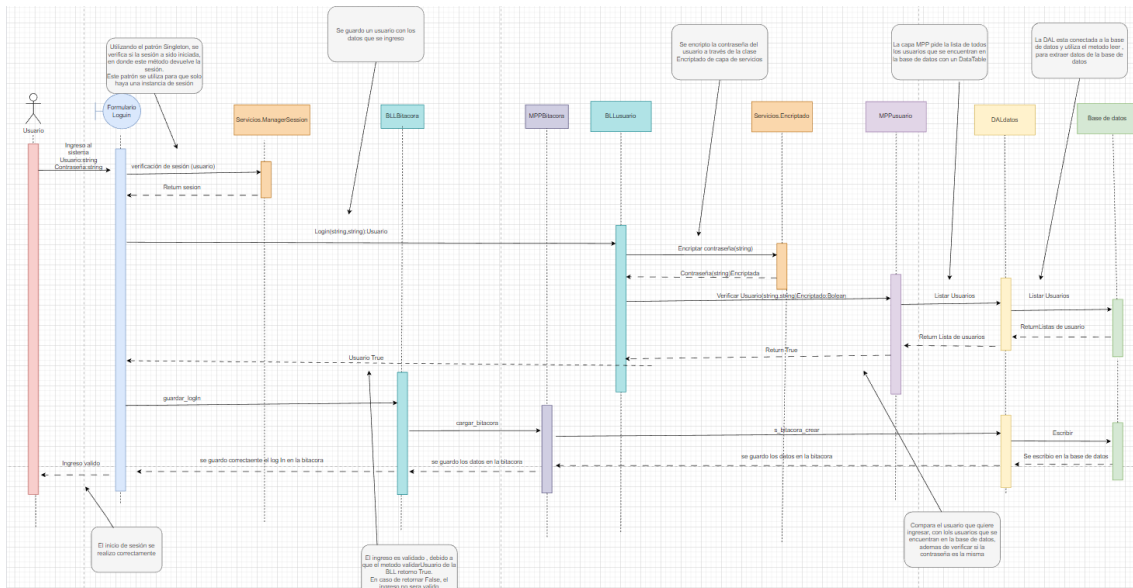


Diagrama de secuencia del Log in

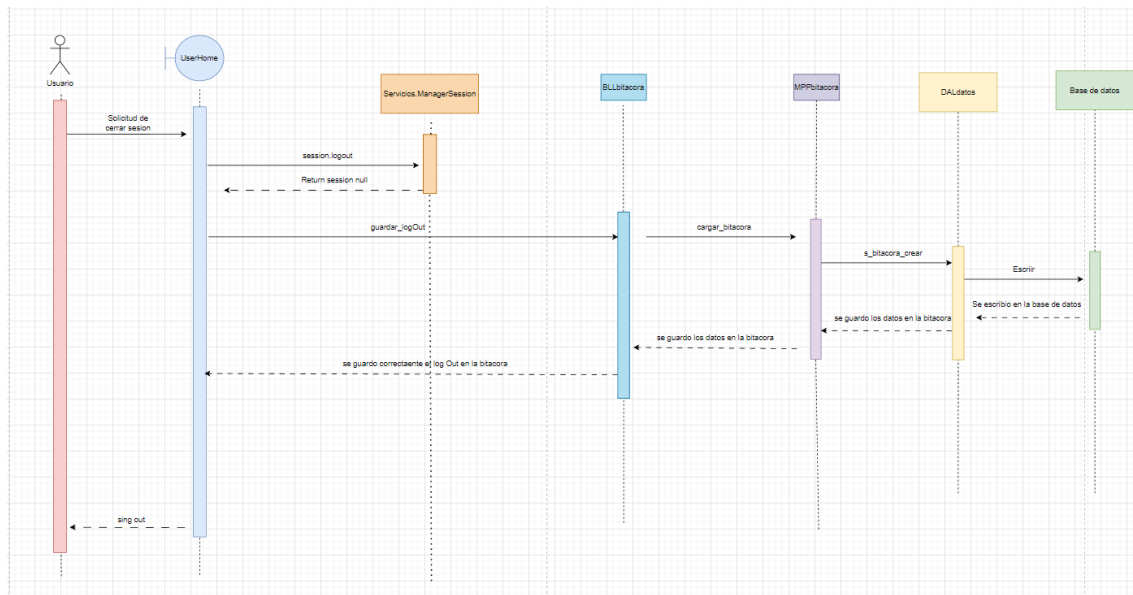


Diagrama de secuencia de Log out

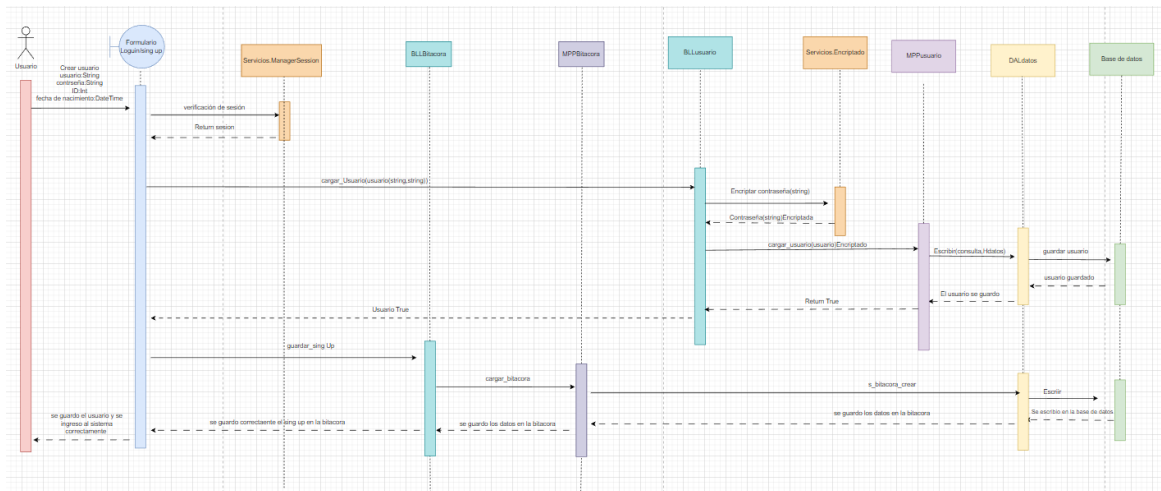


Diagrama de secuencia de Sign Up

## T03 Gestión de encriptado

### ¿Qué es?

El encriptado es un proceso de seguridad que utilizamos para proteger la información confidencial. El objetivo de este encriptado es transformar el texto plano en un formato ilegible y protegido.

La utilización de este genera ventajas al sistema como son la seguridad, la privacidad y confidencialidad.

Ya que su utilización garantiza la protección de información, además de que la vuelve confidencial y privada.

### ¿Cómo funciona?

Este tipo de encriptado es TripleDES este utiliza una clave secreta compartida y un HASH de mensaje.

TripleDES es un algoritmo de cifrado simétrico que utiliza una clave secreta y compartida, para descifrar y cifrar los datos. Esta clave compartida tiene una cierta longitud de bits y se utiliza para cifrar los datos utilizando tres rondas de cifrado DES de forma consecutiva.

### Explicación de código:

Nuestra encriptación tiene dos funciones, ambas retornan un string , nuestra primera función va a ser “static string encriptar” que recibe como parámetro un string que es lo que queremos encriptar.

Este código se encuentra en la capa de Servicios. Hay dos métodos de encriptación de datos.

El primer método se llama “GenerarSHA”, este se utiliza para generar un hash SHA-1 a partir de una cadena de texto dada. Este utiliza una instancia de la clase “SHA1CryptoServiceProvider” para calcular un hash SHA-1 de la cadena de texto dada. En donde el resultado retorna como una cadena en un formato base de 64 bytes.

El segundo método se llama “encriptar” y utiliza una combinación de MD5 y tripleDES para encriptar una cadena de texto dada. Este utiliza una clave llamada “qualityinfosolutions” que es transformada en una matriz de bytes utilizando la clase “MD5CryptoServiceProvider”.

Posteriormente utiliza la clase “TripleDESCryptoServiceProvider” para crear un objeto de cifrado y encriptar la cadena de texto dada.

La cadena de texto encriptada se devuelve como un objeto Hashtable que contiene los valores encriptados de usuario y contraseña, que son agregados a través del método “Add” de la clase Hashtable.

## **T04 Gestión de perfiles de usuario**

¿Qué es?

La gestión de roles es la capacidad del sistema para asignar y administrar diferentes roles y patentes. De esta manera se administran los permisos que los usuarios poseen a las diferentes funcionalidades del sistema.

¿Qué ventajas trae aplicarlo?

La gestión de roles por medio del composite permite que los usuarios puedan tratar a un rol o patente de la misma manera, es decir, es transparente para ellos. De este modo pueden crear, eliminar, asignar o desasignar roles o patentes si tienen el permiso necesario para ello.

¿cómo lo realizamos?

Para gestionar roles utilizamos el patrón composite, en el cual existen roles (familias) y patentes (permisos). Esto nos permite hacer esa diferencia transparente al usuario, y agrupar roles y permisos de una manera flexible y escalable.

Para agregar un nuevo rol un usuario deberá seleccionar un rol padre y tantos roles o permisos hijos como desee, siempre y cuando no genere un loop infinito, ni resulte repetido.

Se introdujo también la funcionalidad de asignar o desasignar roles a un usuario. Esto solo puede llevarse a cabo por otro usuario con permisos de administrador. Para esto se utiliza una interfaz intuitiva que permite seleccionar roles y asignarlos o quitarlos.

Por último, es posible eliminar roles (no así permisos) mediante una interfaz sencilla, se prevé siempre que esos roles no estén en uso por ningún usuario primero.

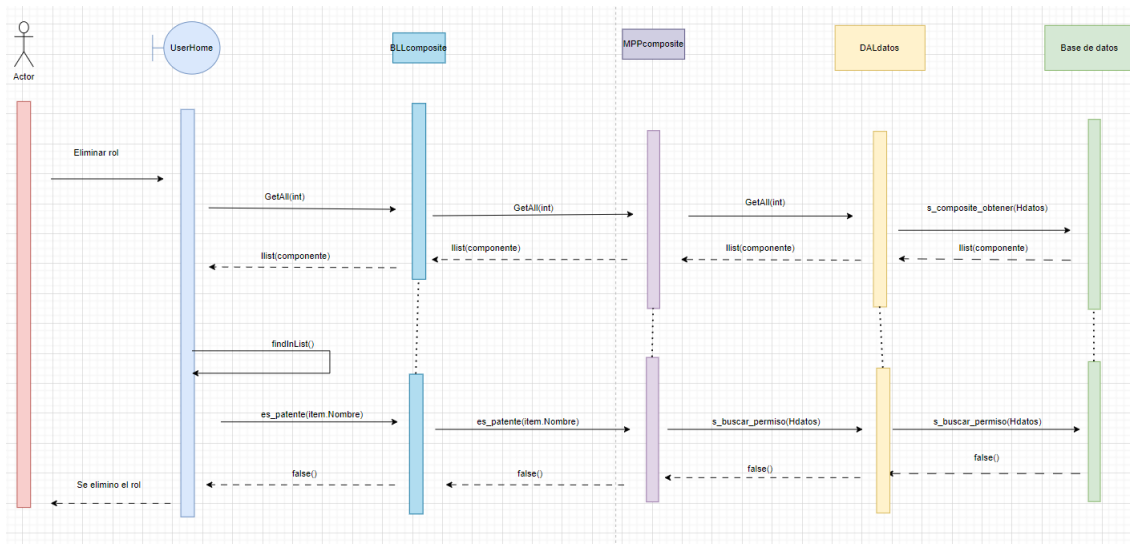
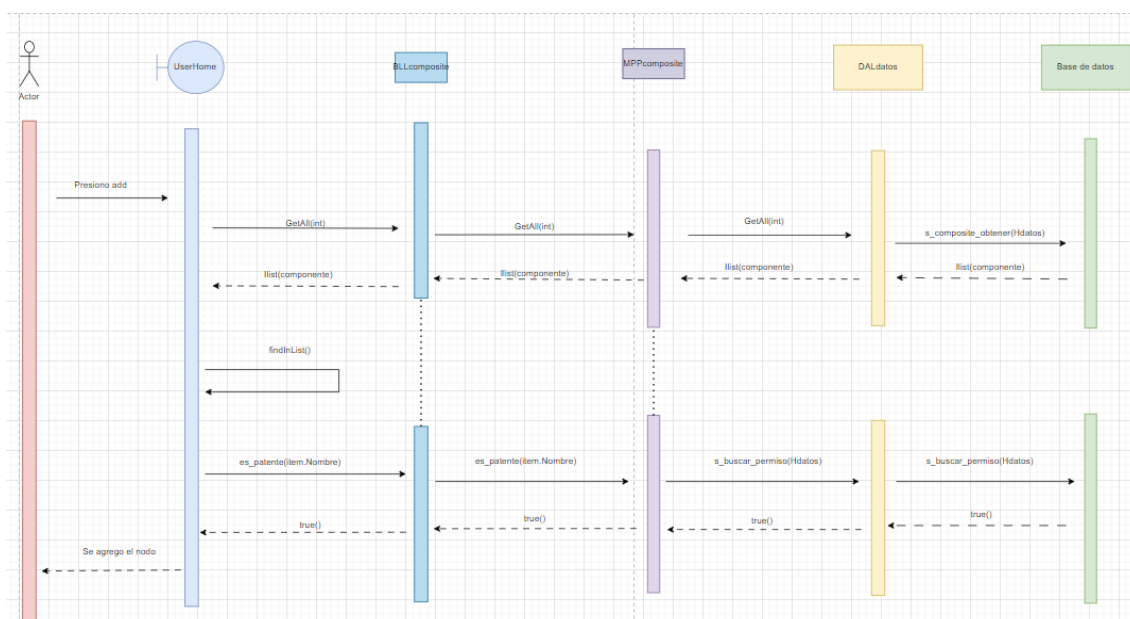
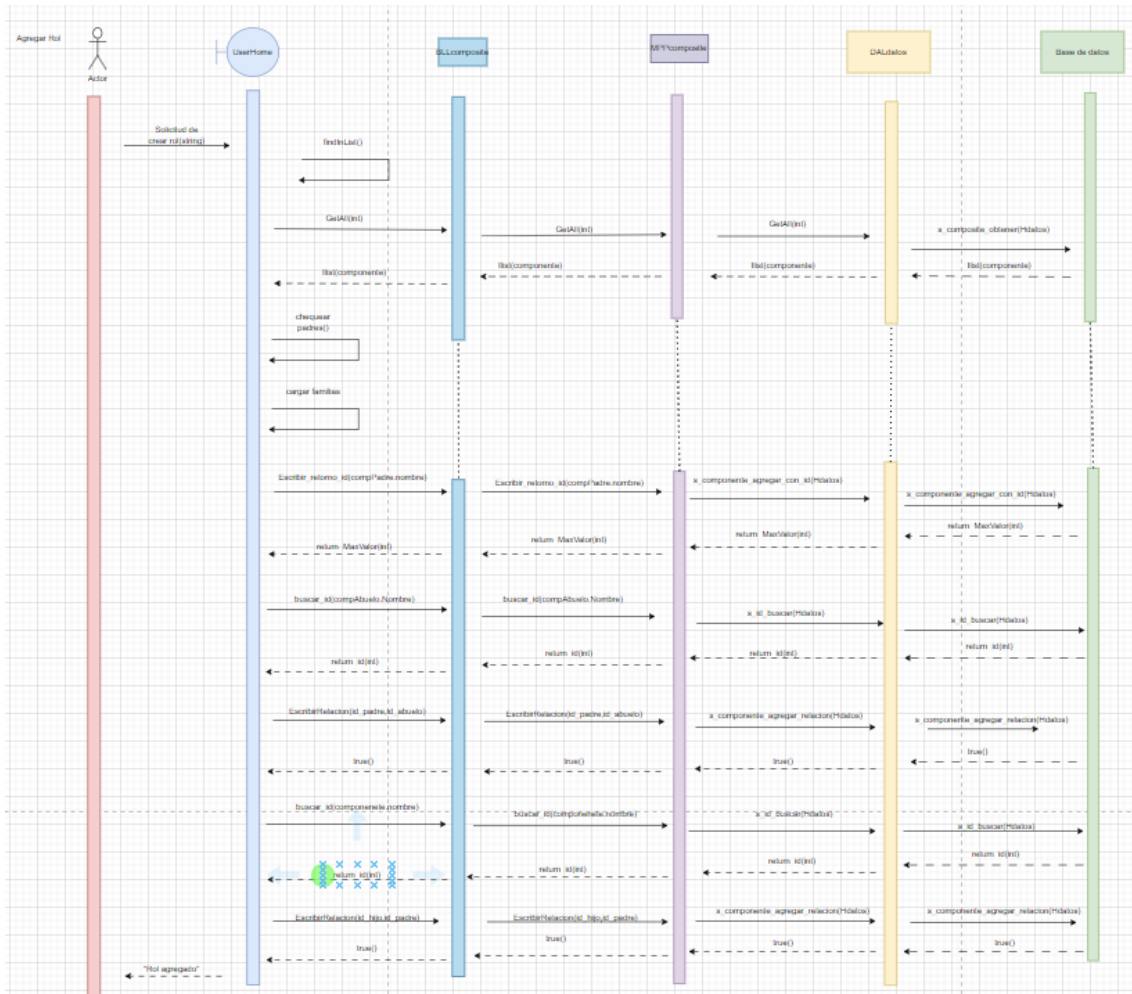


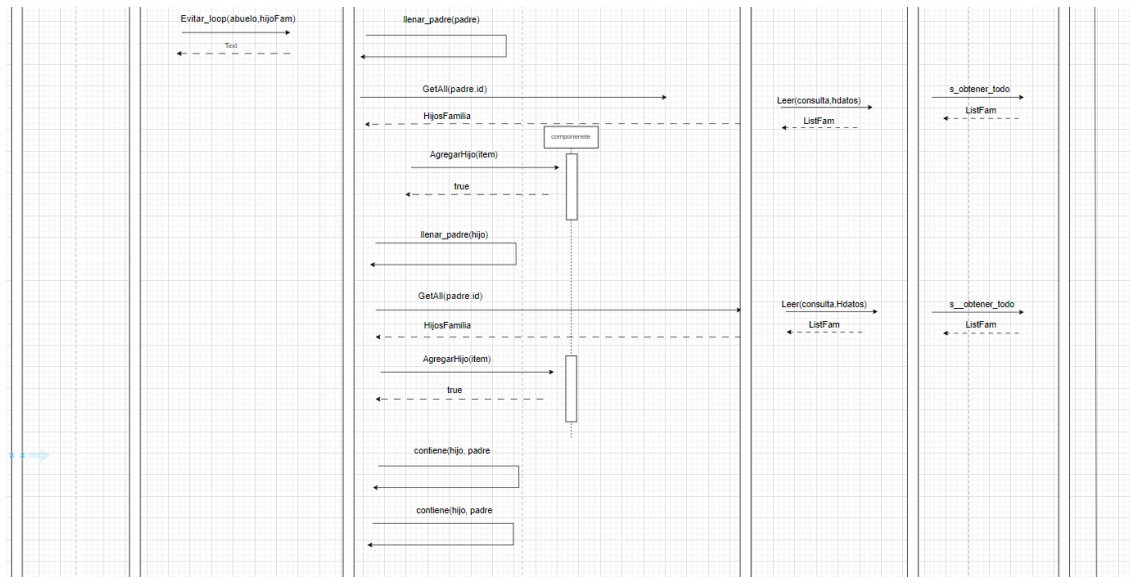
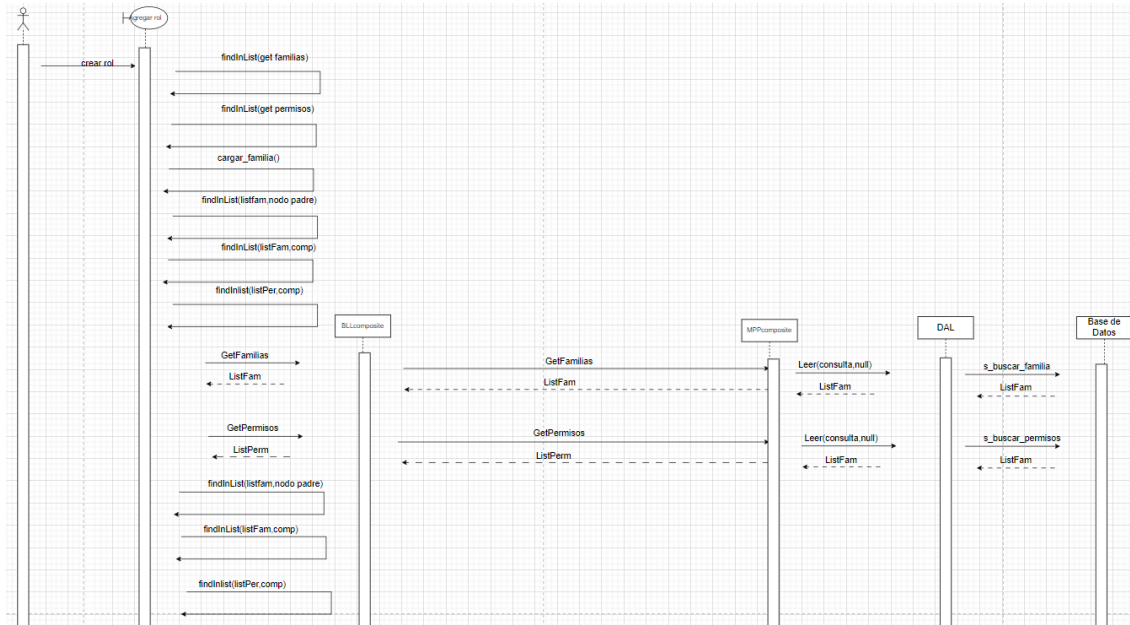
Diagrama de secuencia de eliminación de rol



## Diagrama de secuencia de creación de rol



## Diagrama de secuencia de agregar rol



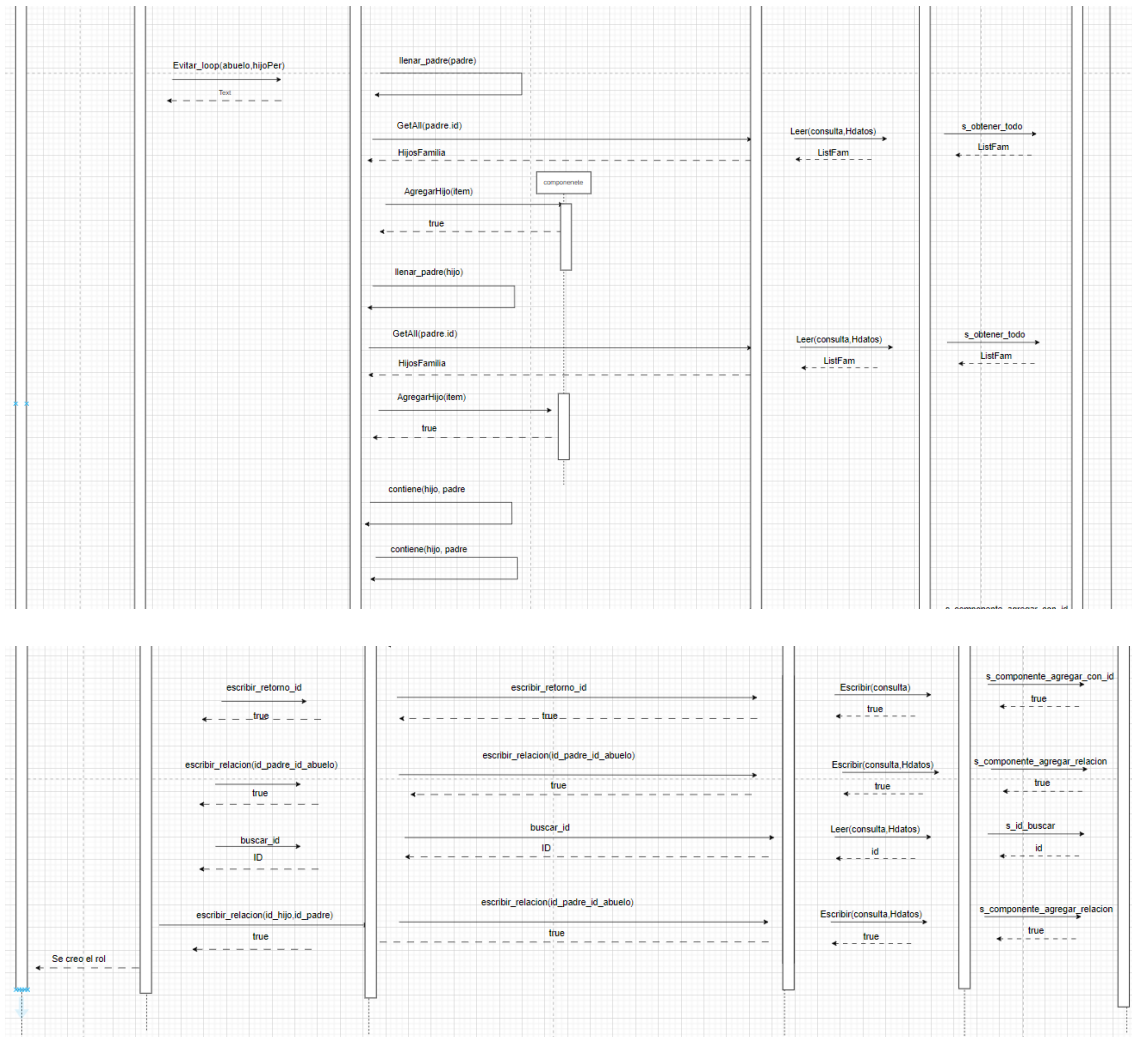


Diagrama de secuencia de crear rol

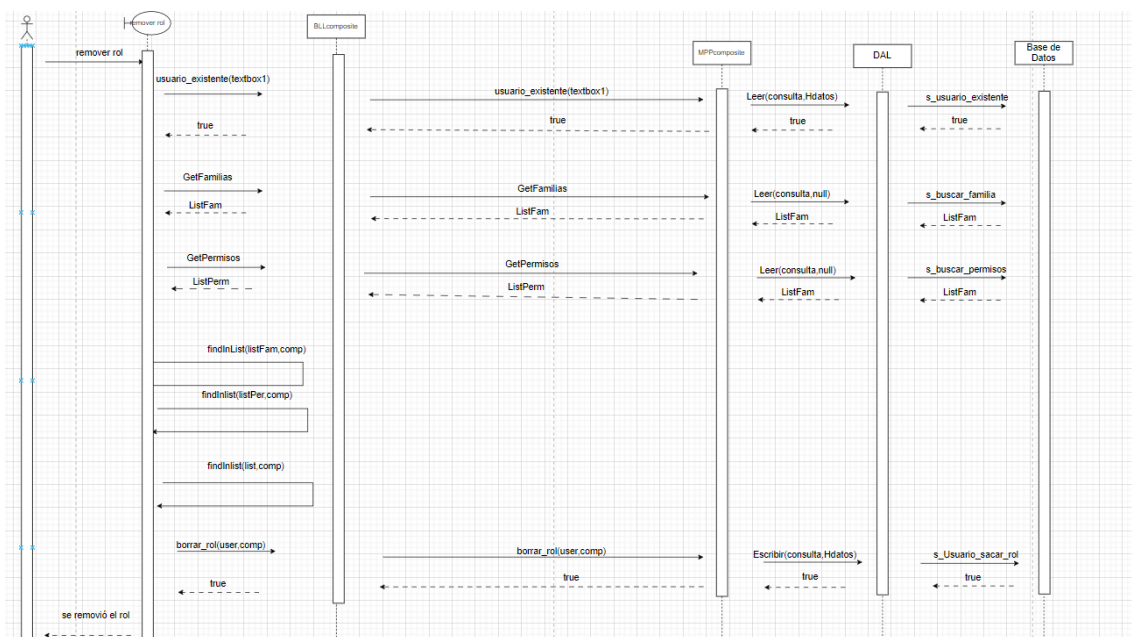


Diagrama de secuencia de remover rol a un usuario



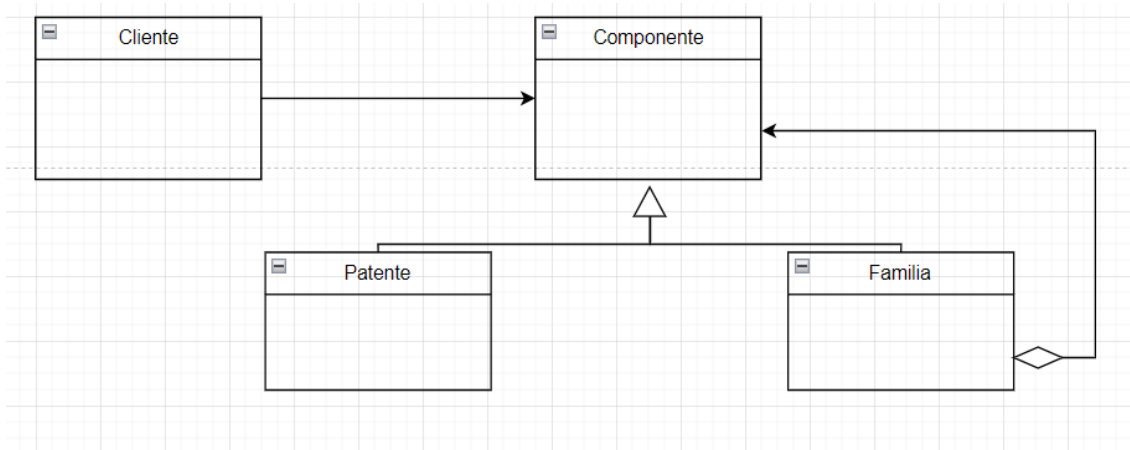


Diagrama de clases de la gestión de roles

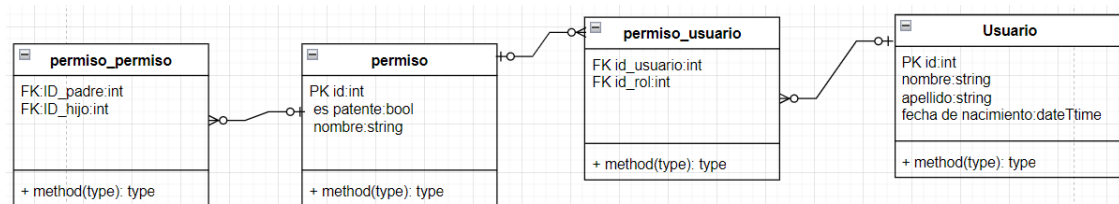


Diagrama de modelado de datos de la gestión de roles

### Diagramas de persistencias:

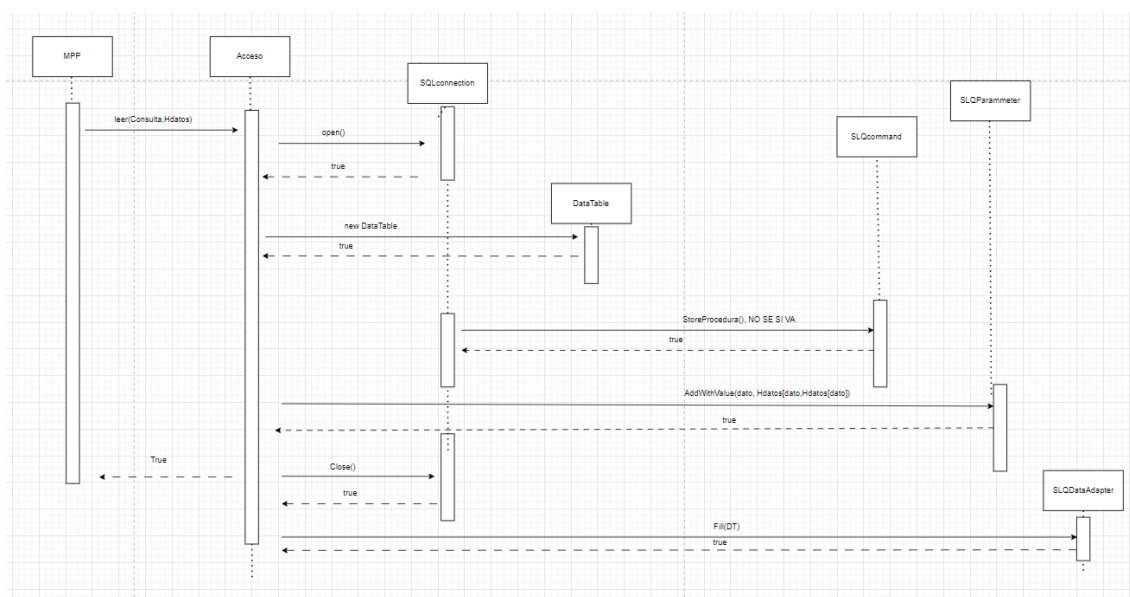


Diagrama de secuencia de lectura de datos

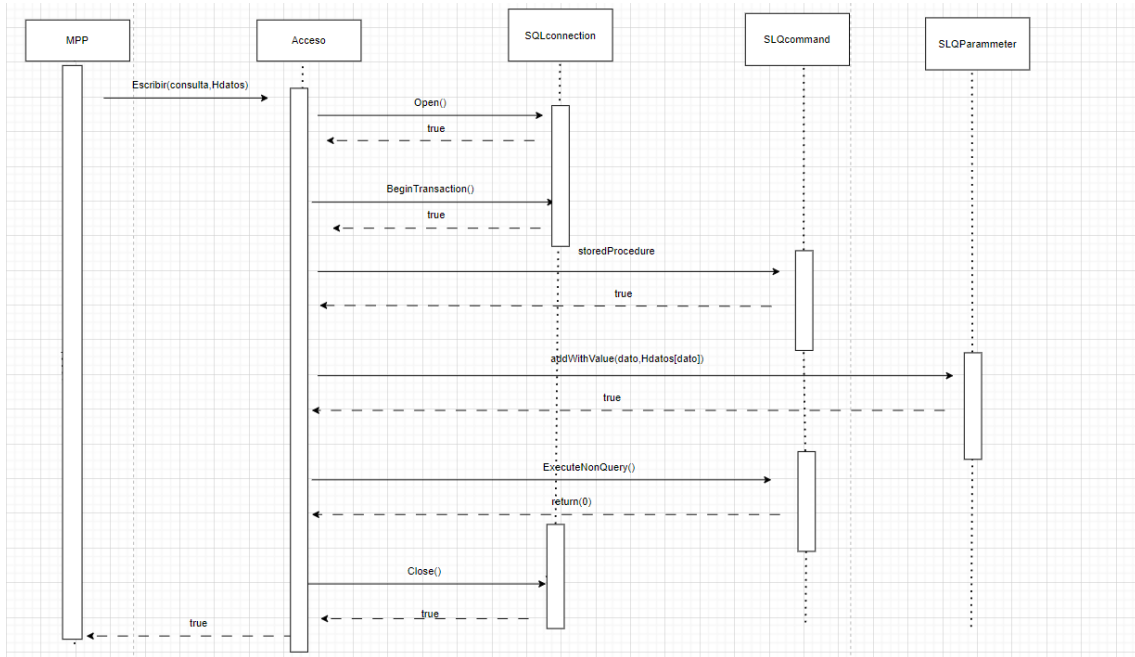
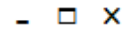


Diagrama de secuencia de escritura de datos

## Create Rol



select the permission you want to add to the role o seleccione desde aqui

Main

admin

gestor

traductor

name of the role

permissions

roles

select as father node

add

x

confirm

Formulario para crear roles

Assing role

user id

select the role

all roles

- Main
  - admin
  - gestor
  - traductor

user roles

- Main
  - admin
  - traductor

or select from here

permissions

roles

Formulario para crear y eliminar roles

## T05 Gestión de múltiples idiomas

¿Qué es?

La gestión de múltiples idiomas es la capacidad de un sistema para adaptarse y ofrecer contenido en diferentes idiomas, de acuerdo con las preferencias del usuario.

Explicación:

Esto lo podemos desarrollar a través del patrón observer.

Este es un patrón de diseño que se utiliza para establecer una relación de dependencia de uno a muchos entre objetos, de modo que cuando un objeto cambia su estado, todos los objetos dependientes son notificados y actualizados automáticamente.

Este patrón está compuesto por 3 métodos que son agregar observador, eliminar observador y notificar a los observadores.

En este caso los observadores son los formularios, dentro de la clase observer se instanciará una lista de observadores, cuando se abre un formulario, este será agregado a la lista, en caso de que el formulario se cierre, este se removerá de la lista.

Cuando se quiera cambiar el idioma, este se le notificará a todos los observadores que estén en la lista.

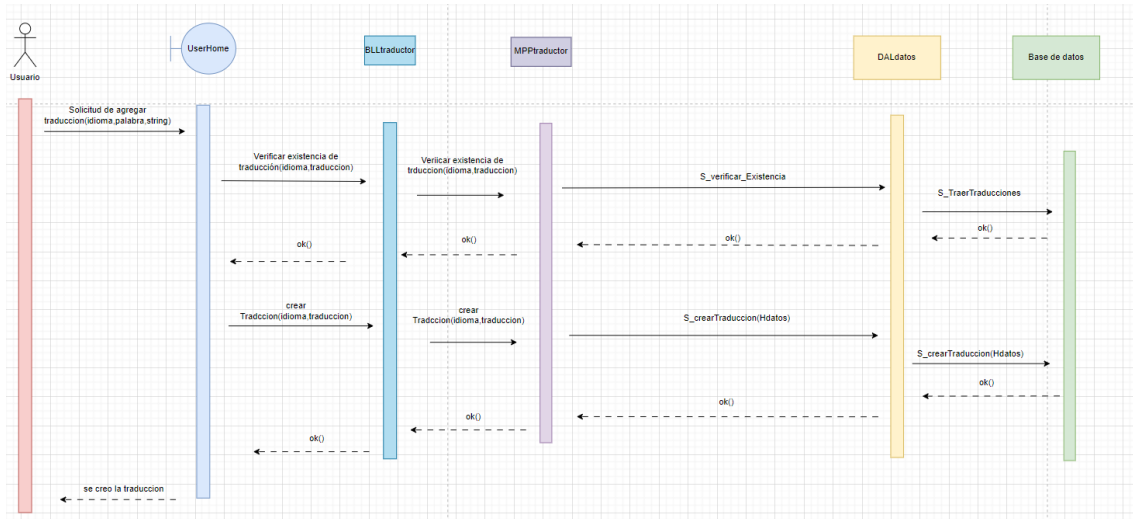


Diagrama de secuencia de agregar traducción

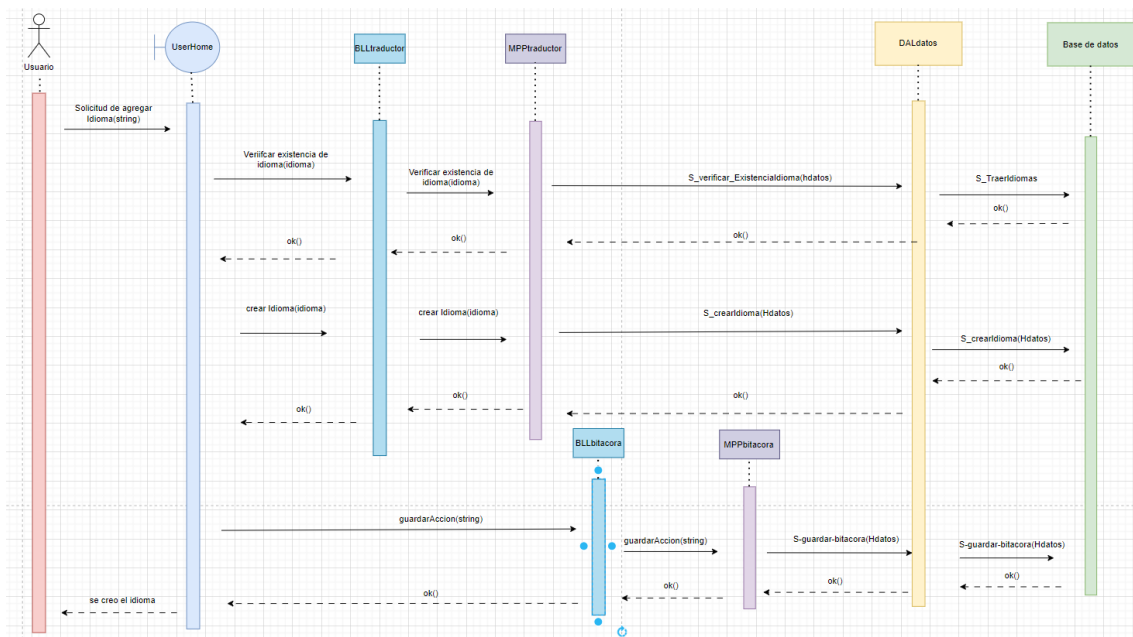


Diagrama de secuencia de agregar idioma

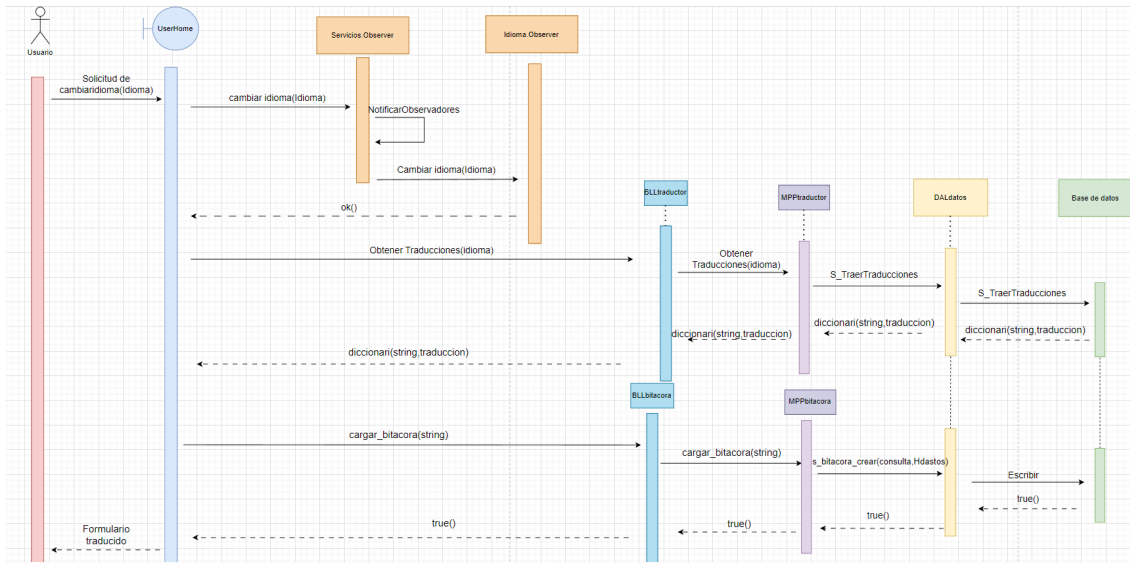


Diagrama de secuencia cambio de idioma

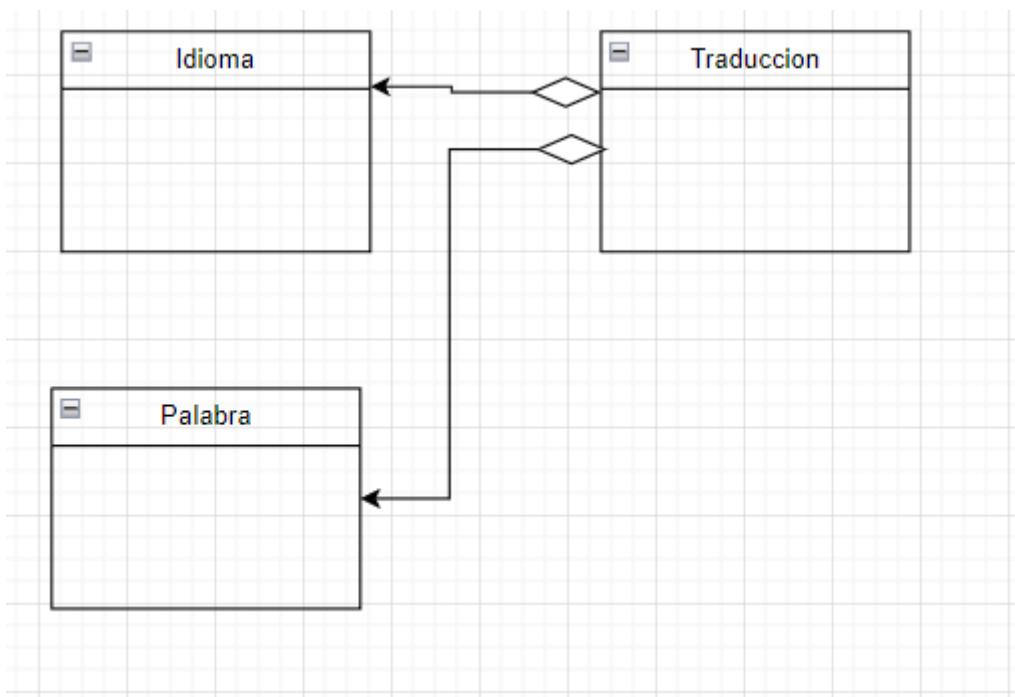
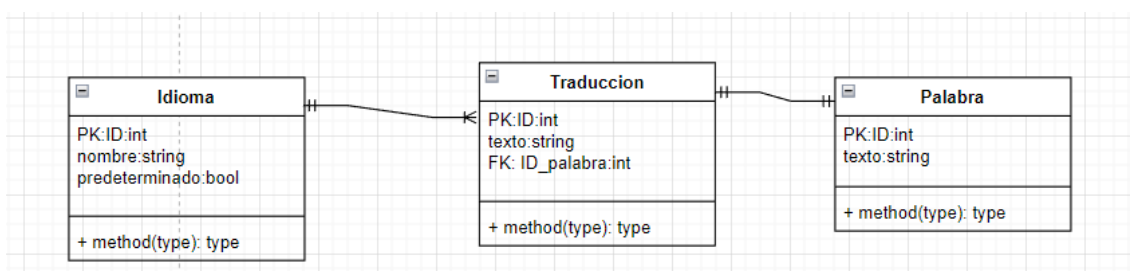


Diagrama de clases



## diagrama de modelo de datos

Traduccion	Palabra
casa do administrador	admin home
id	id
Bemvindo	welcome!
senha	password
linguagem	lenguaje

Formulario para crear un lenguaje

## T06 Gestión de bitácora y control de cambios

¿Qué es la bitácora?

La bitácora es el registro detallado y cronológico de los eventos, actividades y cambios que ocurren dentro del sistema. Esta es fundamental para el seguimiento y la resolución de los problemas, ya que nos puede proporcionar información histórica para encontrar la secuencia de eventos que llevaron a una situación en particular.

La bitácora incluye información como la fecha y hora de una actividad, el usuario que la realizó y cualquier tipo de resultado o error asociado a esa actividad. Volviéndose una herramienta esencial para el monitoreo y el mantenimiento del sistema, volviéndose de gran utilidad.

Explicación de código de la bitácora:

Para poder realizar una gestión de bitácora creamos dos clases, una se encuentra dentro de la capa BLL llamada “BLLbitácora” y otra dentro de la clase MPP llamada “MPPbitácora”.

Dentro de la clase BLLbitacora se encuentran tres métodos, estos son: “guardar\_accion”, “guardar\_logIn” y “guardar\_loggOut”.

El primer método “guardar\_accion” recibe un parámetro “acción” que es del tipo string, esta almacena información de la acción realizada por el usuario. Este método crea una instancia del objeto “MPPbitacora” de la capa MPP, carga el usuario actual y la fecha actual gracias a la sessionmanager, para después llamar al método “cargar\_bitacora” de la instancia creada anteriormente para almacenar la información en la base de datos.

Los otros dos métodos “guardar\_logIn” y “guardar\_LogOut” funcionan de la misma manera que el método explicado anteriormente solo que estos no reciben un parámetro, sino que la acción ya está asignado dentro del método. Por ejemplo en el de “guardar\_logOut” la acción ya almacenada dentro del método será “logged out”.

Una vez que estos parámetros lleguen a MPPbitacora a través de alguno de estos 3 métodos, el método “cargar\_bitacora” asigna los valores del parámetro a las claves de Hashtable. Después utiliza la clase “Acceso” para llamar al método “escribir” que ejecutará la consulta que está almacenada que es “S\_bitacora\_crear” con los parámetros correspondientes. Si todo se realiza de manera exitosa el método retorna un true sabiendo que los datos se han almacenado correctamente dentro de la base de datos, de caso contrario retorna un false.

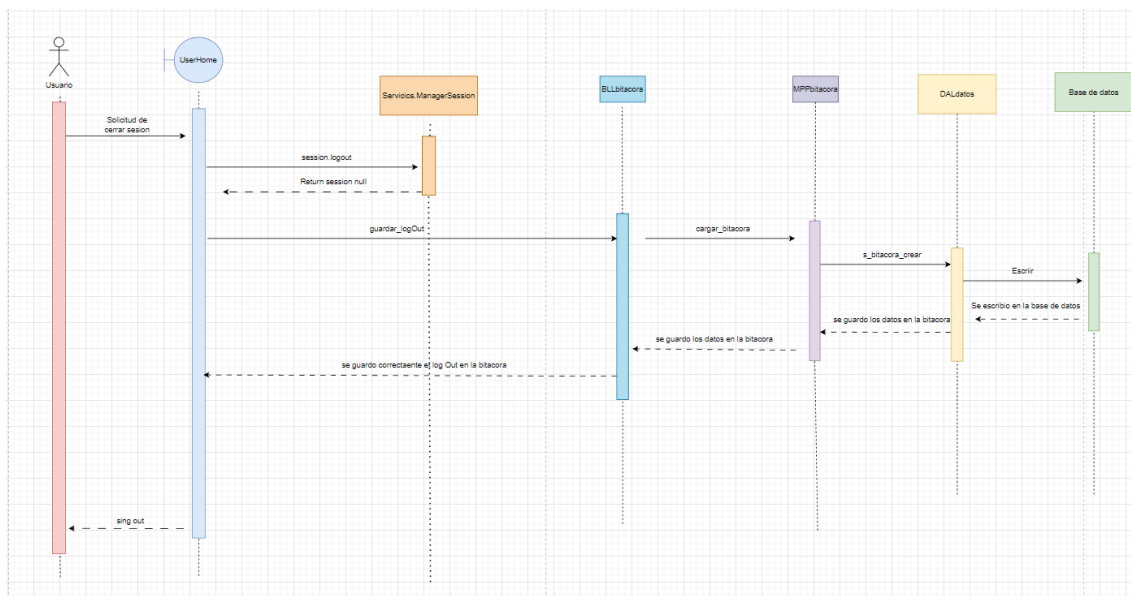


Diagrama de secuencia de Log out que también incluye el funcionamiento de la Bitácora



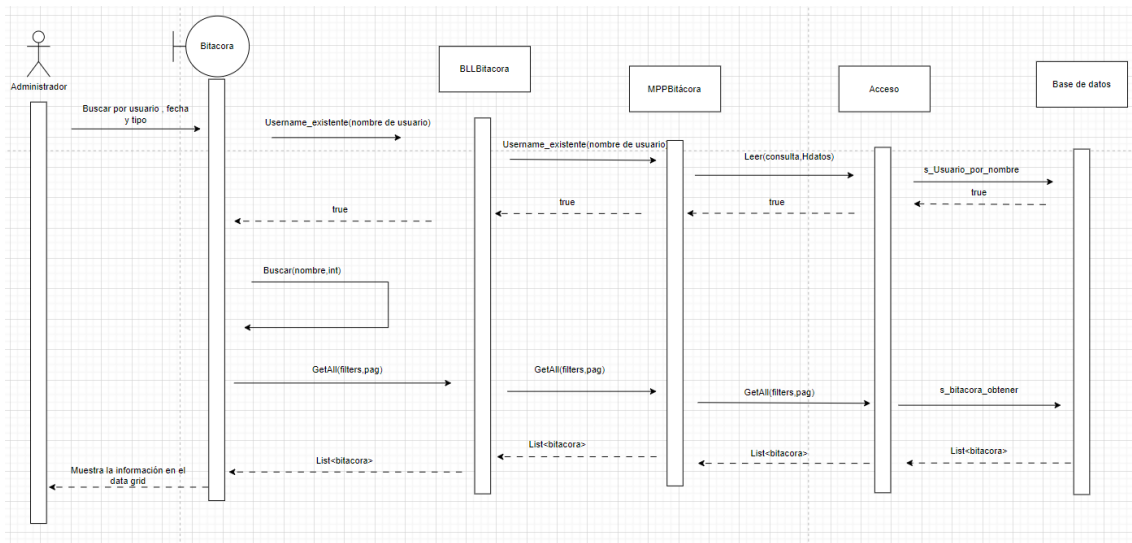


diagrama de secuencia de como funciona la búsqueda dentro de la bitácora

¿Qué es la gestión de cambios?

Es el seguimiento de las modificaciones realizadas en un objeto, en este caso, esta gestión de cambios se los hicimos a los usuarios. Esto implica registrar y controlar los cambios efectuados a sus datos, así como administrar las versiones del objeto a lo largo del tiempo.

¿Cómo funciona la gestión de cambios?

Dentro de la base de datos se guardan las versiones que va teniendo los datos del usuario a lo largo del tiempo, esto nos permite poder hacer una observación a las distintas versiones del mismo.

Dentro del programa hay un formulario que a través de un textbox nos permite ingresar el nombre de un usuario y poder observar sus cambios.

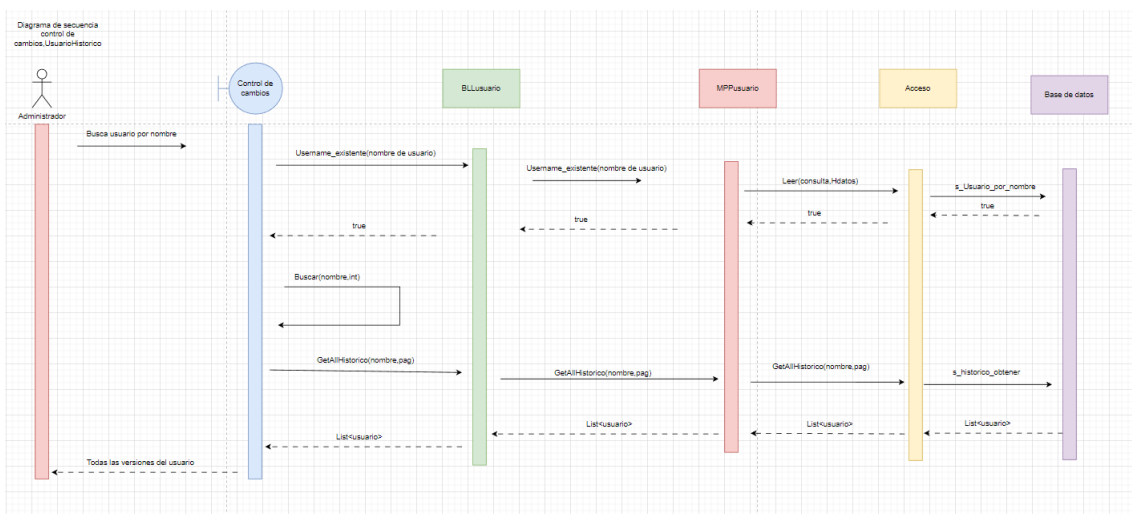


diagrama de secuencia de como se busca dentro del sistema todas las versiones históricas del usuario

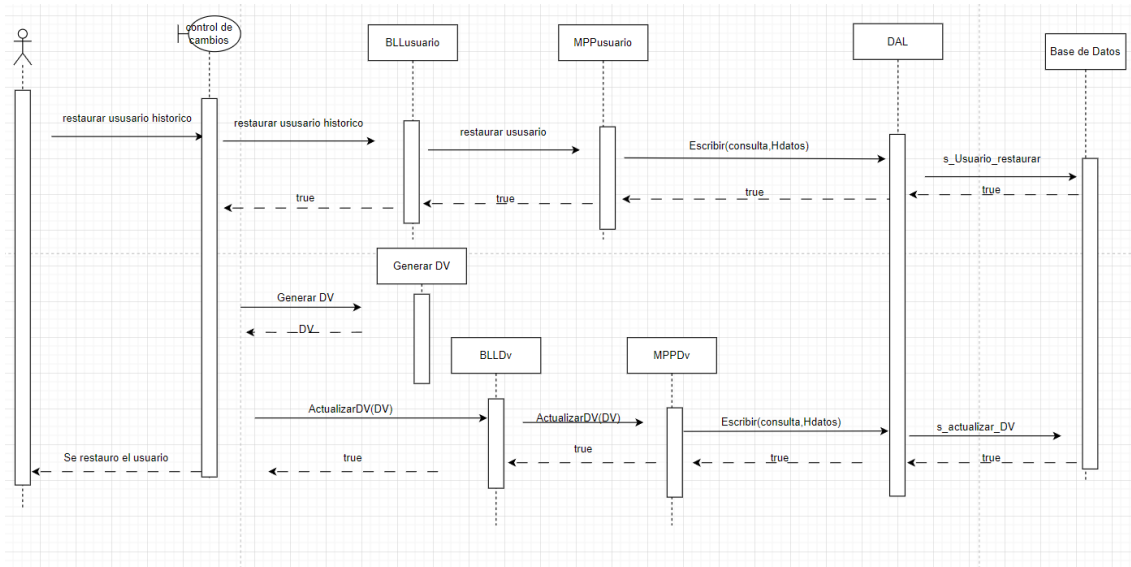


Diagrama de secuecnia de la restaurio del usuario

Binnacle

username  x type  x date from viernes, 14 de julio d▼ to viernes, 14 de julio d▼

	IdBitacora	Uusername	Date	Message	Type
▶	1657	gaston	14/7/2023 09:46	logged in	2
	1658	gaston	14/7/2023 09:46	logged out	2
	1659	gaston	14/7/2023 09:46	logged in	2
	1660	gaston	14/7/2023 09:46	logged out	2
	1661	renavidal	14/7/2023 09:46	logged in	2
	1662	renavidal	14/7/2023 09:46	logged out	2
	1663	renavidal	14/7/2023 09:47	logged in	2
	1664	renavidal	14/7/2023 09:48	logged in	2
	1665	renavidal	14/7/2023 09:48	dio privilegios de ...	2
	1666	renavidal	14/7/2023 09:50	dio al usuario 449...	2

Formulario para ver la bitácora del sistema

## T07 Gestión de dígito verificador

¿Qué es?

El dígito verificador es un dígito que tiene todos los usuarios, este se realiza a través de un HASH , en donde se encripta todos los datos del usuario formando un string, este es almacenado dentro de la base de datos, a su vez el sistema en general tendrá un dígito verificador , este se realiza también a través de un HASH , en donde se encripta todos los dígitos verificadores de los usuarios.

¿Qué ventajas trae aplicarlo?

Este Dígito nos permite mejorar la seguridad del sistema, debido a que cuando un usuario quiera entrar al mismo , se verificará que el Dígito verificador sea correcto, de no ser así, nos daremos cuenta de que hubo una alteración de datos dentro de la base de datos.

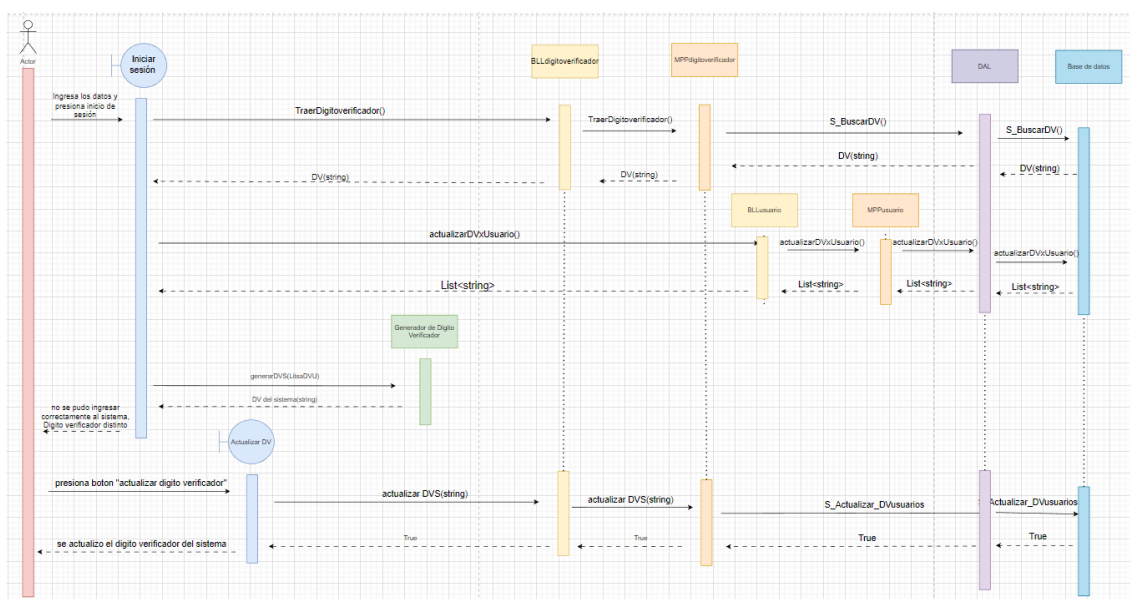
Gracias a la gestión de control de cambios , podremos volver los datos del usuario a la última vez que se haya guardado dentro de la gestión de cambios, modificando estos datos que fueron alterados al estado que tenían anteriormente.

¿cómo lo realizamos?

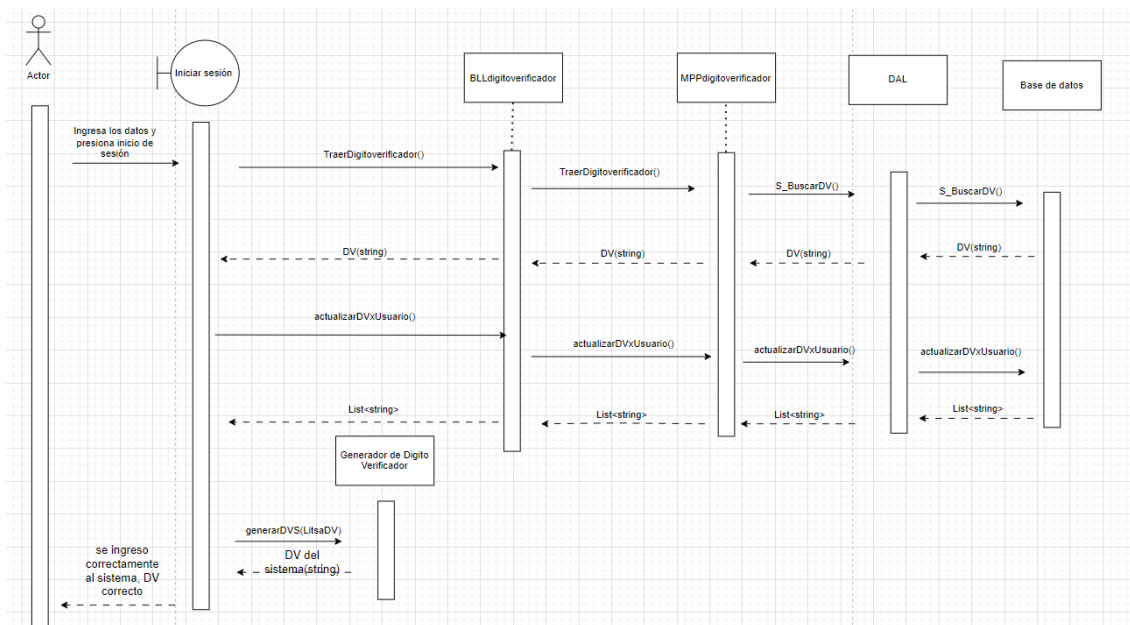
Para poder realizar el dígito verificador de la tabla usuario , primero reuno los datos que voy a utilizar del usuario y a través de un HASH lo transformó todo a una cadena de string. Esta cadena es almacenada dentro de la base de datos dentro del usuario.

Posteriormente se obtienen todos los dígitos verificadores de los usuarios y los encripto a través de un HASH obteniendo una cadena string, esta es almacenada en la base de datos dentro de la base de datos.

Cada vez que se quiera ingresar al sistema se creará una un nuevo dígito verificador con los datos de los usuarios y se comparará con el dígito verificador de la base de datos que se almacenó anteriormente, es caso de que estos dos sean distintos , nos daremos cuenta que hubo una alteración dentro de la base de datos.



## Diagrama de secuencia de error de dígito verificador y la actualización del mismo



## Diagrama de secuencia de verificación del dígito verificador