# UACM

Diseño de software

## Diagramas

SAR

Valadez Carmona Guadalupe Yamileth

Rodríguez Cervantes Kevin Manzur

Cruz Ovando Cristela Adelaida

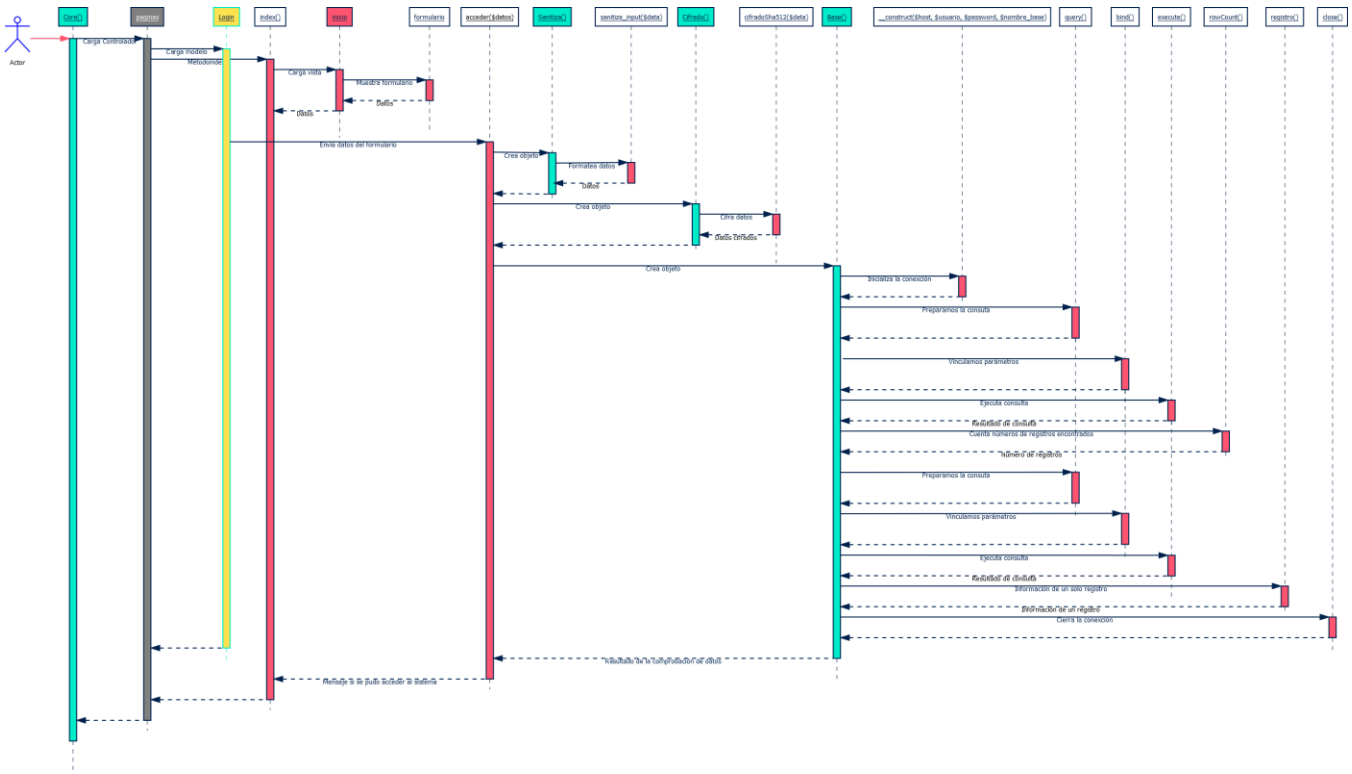
# HISTORIAL DE VERSIONES

| FECHA | VERSIÓN | DESCRIPCIÓN | AUTOR@S |
|---|---|---|---|
| 30/01/2025 | 4.0 | ⊦ Nombre del sistema.<br>⊦ Actualización de los integrantes.<br>⊦ Logo para el proyecto.<br>⊦ Diagrama de clases.<br>⊦ Diagrama de secuencia. | Guadalupe Yamileth, Manzur Rodriguez, Cristela Adelaida |
| 31/01/2025 | 4.1 | ⊦ Nuevo formato del documento. | Guadalupe Yamileth, Manzur Rodriguez, Cristela Adelaida |
| 21/02/2025 | 4.2 | ⊦ Actualización del documento, siguiente el 'Estándar de Documentación V - 2.0' | Manzur Rodriguez |

# INDICE

# 1. Diagrama de secuencia

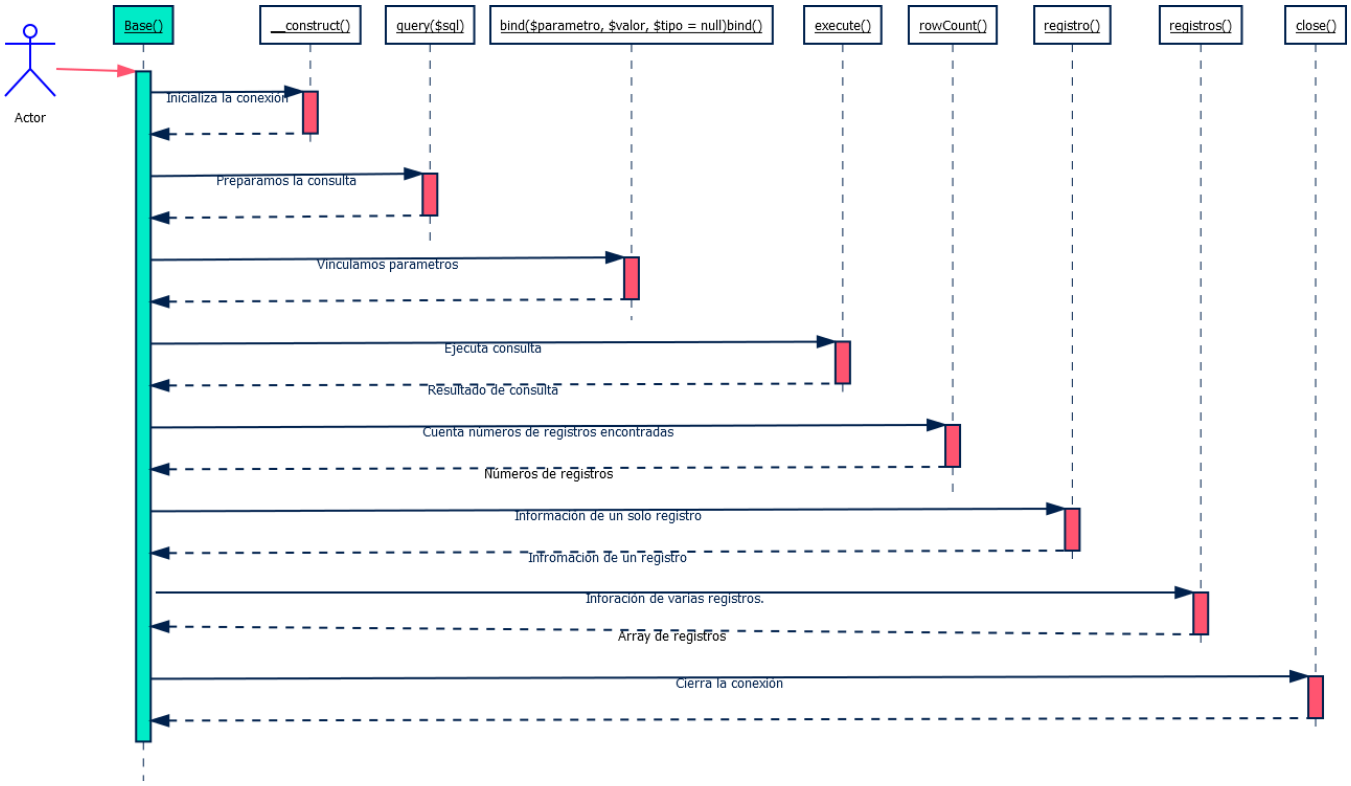
## 1.1. Login()




| Diagrama | Login |
|---|---|
| El vigilante, tiene que iniciar sesión en el sistema. | |
| **Descripción** | |
| 1. La clase Core, carga por defecto al controlador "Paginas".<br>2. El controlador "Paginas", inicializa el modelo "Login".<br>3. El controlador "Paginas", inicializa el método "index()".<br>4. El método "index()", carga la vista "inicio".<br>5. La vista "inicio", muestra el formulario de acceso.<br>6. Accedo al metodo "acceder ($datos)", del modelo "Login". El parámetro son los datos recibidos del formulario.<br>7. Inicializo la clase "Sanitiza()".<br>8. De la clase "Sanitiza()", accedo al método "sanitize_input($data)", está fórmate la información. El parámetro "$data", son los datos que serán formateados y retornados de nuevo.<br>9. Inicializo la clase "Cifrado()".<br>10. De la clase "Cifrado()", accedo al método "cifradoSha512($data)", está cifra la información, para compararla con la información cifrada de la DB. El parámetro "$data", son los datos que serán cifrados, y después retornados para su uso.<br>11. Inicializo la clase "Base()". | |

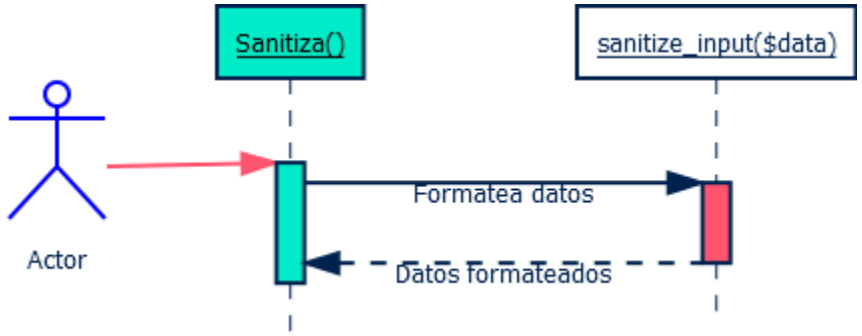
12. De la clase "Base()", acceso al método "__construct($host, $usuario, $password, $nombre_base)", con la que inicializo la conexión a la base de datos que almacena la información de acceso. El parámetro "$host", es la ruta del servidor, "$usuario", usuario con el que se realizara consultas, "$password", es la contraseña del usuario y "$nombre_base", es el nombre de la base de datos a la cual nos conectaremos.
13. De la clase "Base()", acceso al método "query($query)", establece la consulta.
14. De la clase "Base()", acceso al método "bind()", vincula el valor de la consulta con el parámetro recibido de formulario.
15. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.
16. De la clase "Base()", acceso al método "rowCount()", obtiene el número de filas obtenidas de la consulta.
17. Si el número de filas es igual a 1, continua con las consultas, en caso contrario, retornamos mensaje de error.
18. De la clase "Base()", acceso al método "query($sql)", establece la consulta. El parámetro "$sql" es la consulta que se va a realizar.
19. De la clase "Base()", acceso al método "bind($parametro, $valor, $tipo = null)", vincula el valor de la consulta con el parámetro recibido de formulario. El parámetro "$parametro", es el parámetro establecido en la consulta, "$valor", es el valor que tendrá el parámetro y "$tipo", es el tipo de valor.
20. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.
21. De la clase "Base()", acceso al método "registro()", obtenemos el "id" del vigilante que ingreso los datos. Creamos una sesión, con el "id" del vigilante.
22. De la clase "Base()", acceso al método "close()", con esto cerramos la conexión a la DB.

## 1.2. Base()



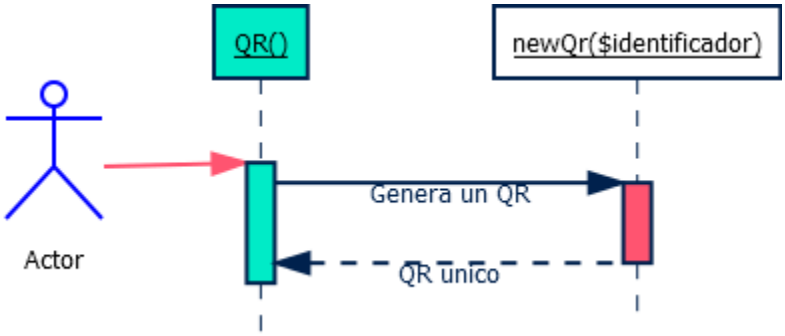
| Diagrama | Base |
|---|---|
| Realiza la conexión a la base de datos, implementada para realizar una conexión PDO. Se encuentra configurada para conectarse a diferentes DB, al crear el objeto se establece la conexión. | |
| **Descripción** | |
| 1. De la clase "Base()", acceso al método "__construct()", con la que inicializo la conexión a la base de datos de manera dinámica.<br>2. De la clase "Base()", acceso al método "query($sql)", establece la consulta. El parámetro "$sql" es la consulta que se va a realizar.<br>3. De la clase "Base()", acceso al método "bind($parametro, $valor, $tipo = null)", vincula el valor de la consulta con el parámetro recibido de formulario. El parámetro "$parametro", es el parámetro establecido en la consulta, "$valor", es el valor que tendrá el parámetro y "$tipo", es el tipo de valor.<br>4. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.<br>5. De la clase "Base()", acceso al método "rowCount()", obtiene el número de filas obtenidas de la consulta.<br>6. De la clase "Base()", acceso al método "registro()", obtiene solo un registro.<br>7. De la clase "Base()", acceso al método "registros()", obtiene un array con varios registros.<br>8. De la clase "Base()", acceso al método "close()", cierra la conexión con la base de datos. | |

## 1.3. Sanitiza()



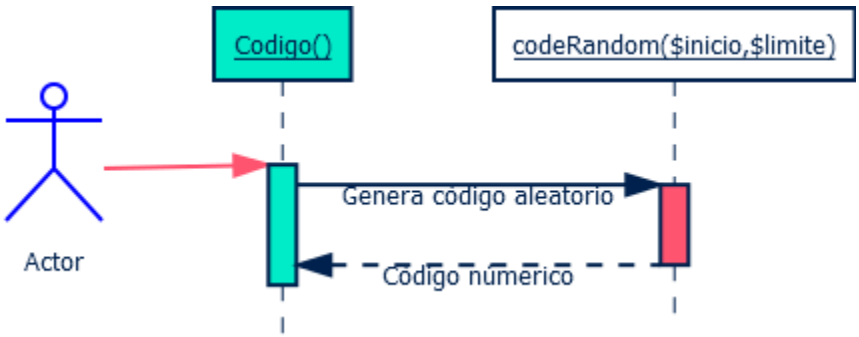

| Diagrama | Sanitiza |
|---|---|
| Formatea (sanitiza) la información para guardarla en la DB, evitando inyecciones. | |
| **Descripción** | |
| 1. De la clase "Sanitiza()", accedo al método "sanitize_input($data)", está fórmate la información. El parámetro "$data", son los datos que serán formateados y retornados de nuevo. | |

## 1.4. QR()



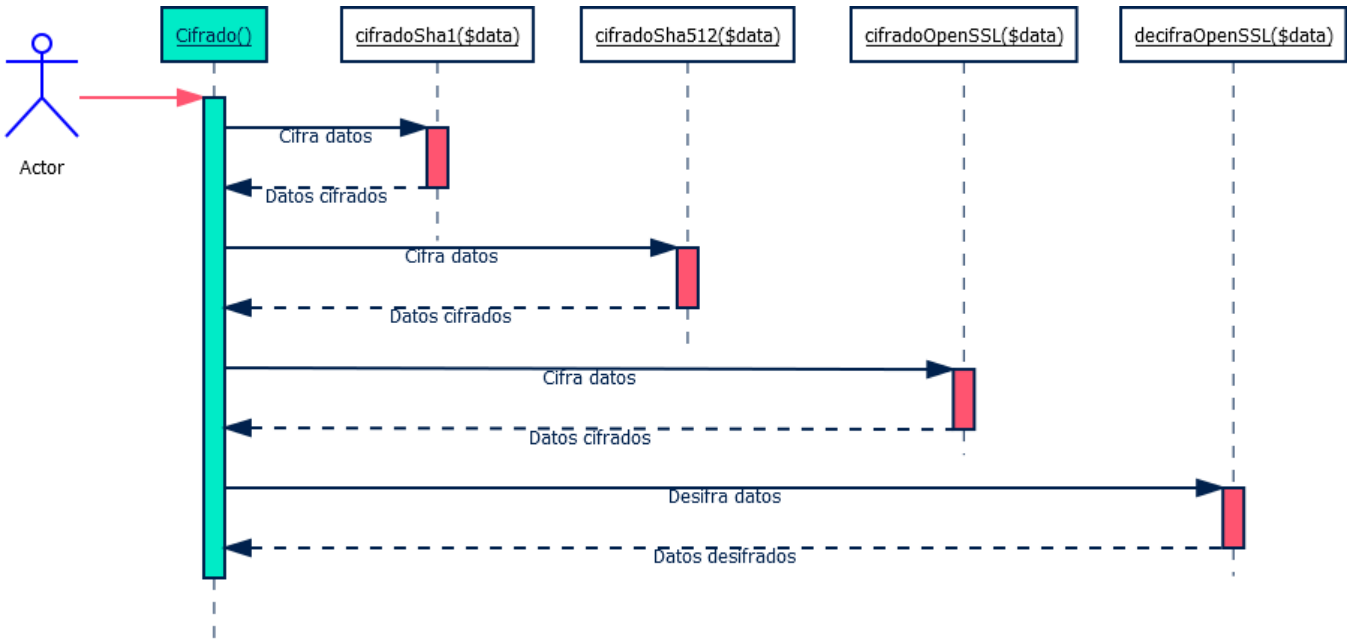

| Diagrama | QR |
|---|---|
| Permite la generación de un código QR único para un visitante. Los códigos QR almacena un enlace web, el cual tiene como método GET, un identificador, este se ha proporcionado como parámetro. Se debe crear el QR como una imagen, la cual será almacenada en la base de datos. | |
| **Descripción** | |
| 1. De la clase "QR()", accedo al método "newQr()", este genera un QR, el cual almacena un enlace web.. | |

## 1.5. Codigo()



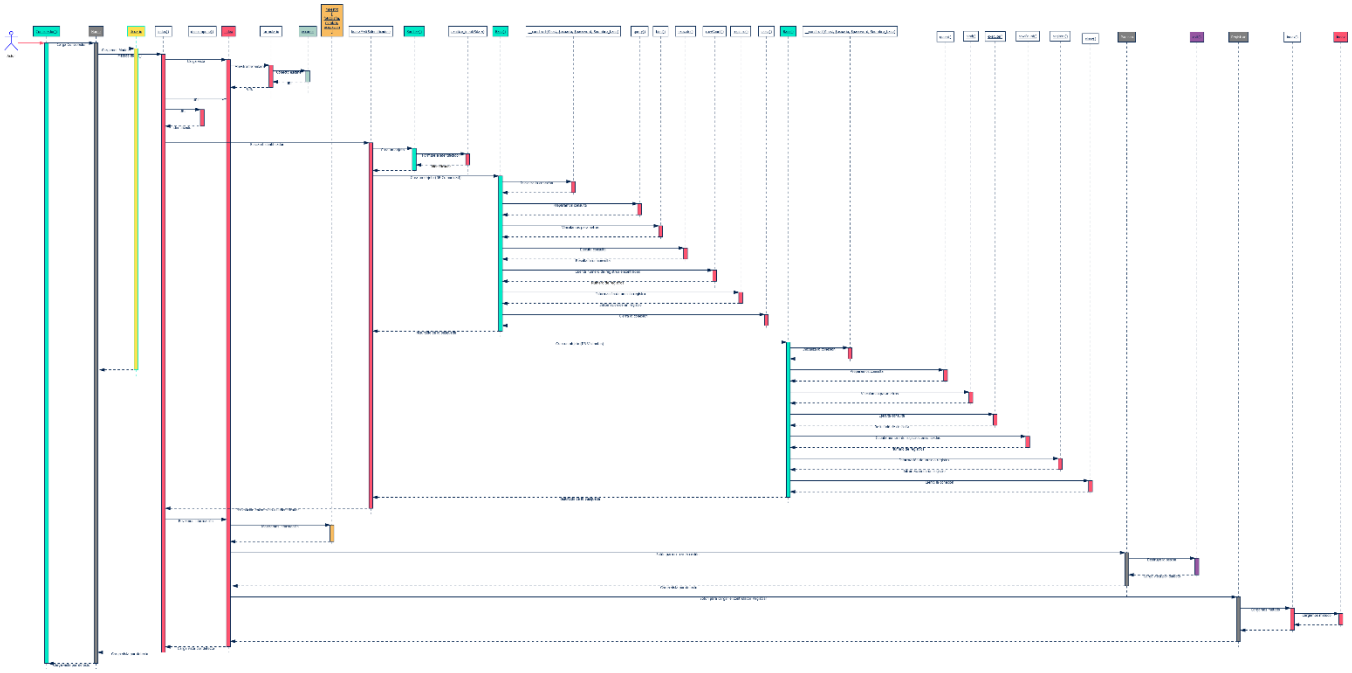

| Diagrama | Codigo |
|---|---|
| Genera códigos aleatorios de tipo numéricos utilizando combinaciones. | |
| **Descripción** | |
| 1. De la clase "Codigo()", accedo al método "codeRandom($inicio,$limite)", este genera un QR, el cual almacena un enlace web. EL param. | |

## 1.6. Cifrado()




| Diagrama | Cifrado |
|---|---|
| Ciframos y desciframos la información de la DB. | |
| **Descripción** | |
| 1. De la clase "Cifrado()", accedo al método "cifradoSha1($data)", este es un cifrado irreversible con sha1. El parámetro "$data", son los datos que serán cifrados, y después retornados para su uso.<br>2. De la clase "Cifrado()", accedo al método "cifradoSha512($data)", este es un cifrado único con sha512, con una longitud de 129 caracteres. El parámetro "$data", son los datos que serán cifrados, y después retornados para su uso.<br>3. De la clase "Cifrado()", accedo al método "cifradoOpenSSL($data)", este cifra la información con OpenSSL, esta es reversible. El parámetro "$data", son los datos que serán cifrados, y después retornados para su uso.<br>4. De la clase "Cifrado()", accedo al método "decifraOpenSSL($data)", con este desciframos la información, del cifrado OpenSSL. El parámetro "$data", son los datos que serán cifrados, y después retornados para su uso. | |

## 1.7. Home

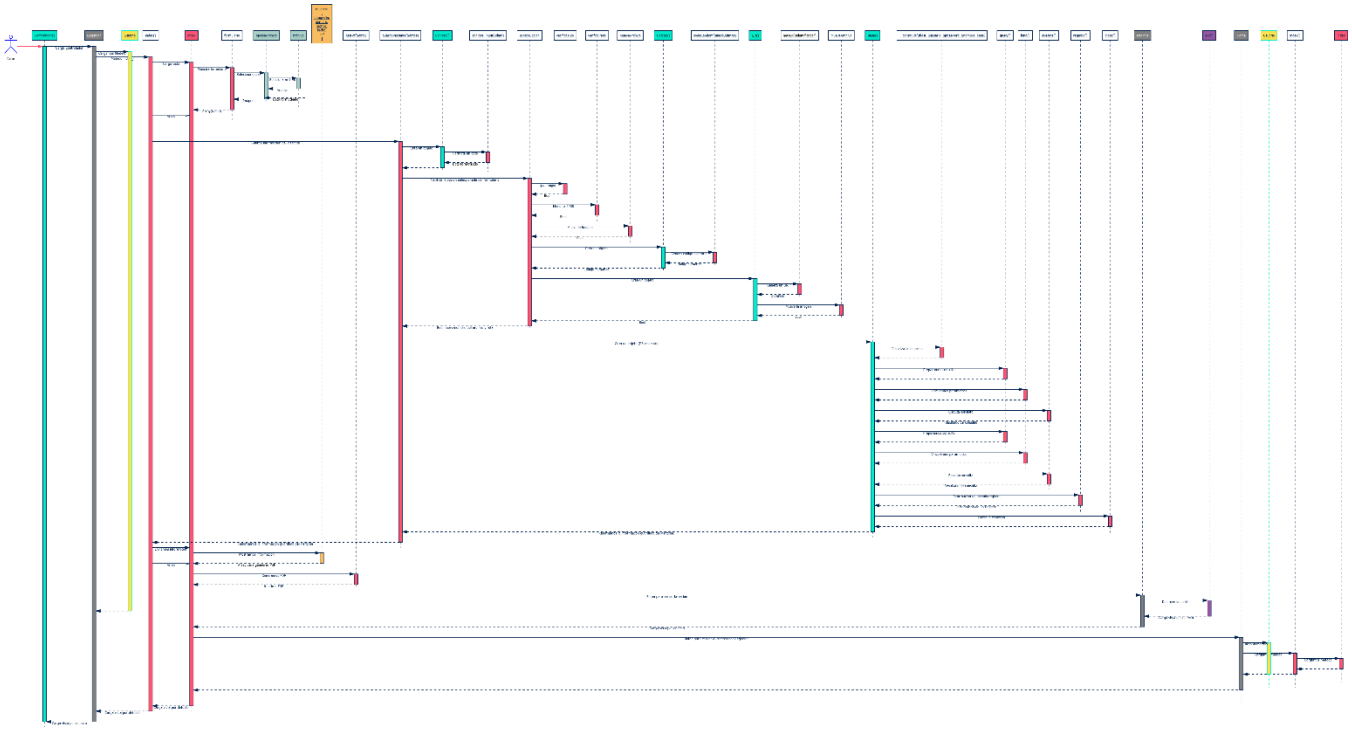


| Diagrama | Home |
|---|---|
| Muestra la información guardada en los códigos QR, trabaja con el escáner<br>En caso afirmativo, la vista mostrará los datos del usuario (fotografía, nombre, carrera) junto con la leyenda "Autorizado". En caso contrario, se presentará la leyenda "Denegado", lo que permitirá al personal de seguridad decidir si se le otorga acceso como visitante, considerando los requisitos necesarios para ello. | |
| **Descripción** | |
| 1. De la clase controlador, cargamos el controlador "Home".<br>2. Del controlador "Home", cargamos el modelo "Usuario".<br>3. Del controlador "Home", accedemos al método "index()".<br>4. Del método "index()", mostramos la vista "index".<br>5. De la vista "index", mostramos el formulario.<br>6. Dentro del formulario, se utiliza el escáner, el cual nos sirve para leer los códigos QR de las credenciales.<br>7. El escáner nos retorna la URL almacenada en el código QR.<br>8. El "URL", es retornado al método "index", del controlador "Home".<br>9. Del controlador "Home", enviamos la URL al método "descompone", el cual descompondrá la URL, y nos retornará el identificador almacenado.<br>10. Del método index(), utilizamos el modelo "Usuario", para enviar el identificado al método "buscaMat($identificador)".<br>11. Creamos un objeto de la clase "Sanitiza()".<br>12. Utilizando el objeto de la clase "Sanitiza()", enviamos el identificador al método "sanitiza_input($data)", el cual formateara la información enviada.<br>13. Creamos un objeto de la clase "Base", le enviamos por parámetros, los datos requeridos para conectarnos a la DB de la comunidad, la con la cual comprobaremos si existe el identificador. | |

14. De la clase "Base()", acceso al método "__construct($host, $usuario, $password, $nombre_base)", con la que inicializo la conexión a la base de datos que almacena la información de acceso. El parámetro "$host", es la ruta del servidor, "$usuario", usuario con el que se realizara consultas, "$password", es la contraseña del usuario y "$nombre_base", es el nombre de la base de datos a la cual nos conectaremos.
15. De la clase "Base()", acceso al método "query($query)", establece la consulta.
16. De la clase "Base()", acceso al método "bind()", vincula el valor de la consulta con el parámetro recibido de formulario.
17. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.
18. De la clase "Base()", acceso al método "rowCount()", obtiene el número de filas obtenidas de la consulta.
19. Si el número de filas es igual a 1, de a clase "Base()", accedemos al método "registro". Con esto retornamos la información de un usuario de la comunidad.
20. Si el número de filas es igual a 0, procedemos a cerrar la conexión.
21. De la clase "Base()", acceso al método "close()", cierra la conexión con la base de datos.
22. Si el identificador se encontró en la DB de la comunidad, se retorna su información. Si el usuario no existe en la DB, se realiza otra consulta, a la DB de las visitas.
23. De la clase "Base()", acceso al método "__construct($host, $usuario, $password, $nombre_base)", con la que inicializo la conexión a la base de datos que almacena la información de acceso. El parámetro "$host", es la ruta del servidor, "$usuario", usuario con el que se realizara consultas, "$password", es la contraseña del usuario y "$nombre_base", es el nombre de la base de datos a la cual nos conectaremos.
24. De la clase "Base()", acceso al método "query($query)", establece la consulta.
25. De la clase "Base()", acceso al método "bind()", vincula el valor de la consulta con el parámetro recibido de formulario.
26. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.
27. De la clase "Base()", acceso al método "rowCount()", obtiene el número de filas obtenidas de la consulta.
28. Si el número de filas es igual a 1, de a clase "Base()", accedemos al método "registro". Con esto retornamos la información de un usuario de la comunidad.
29. Si el número de filas es igual a 0, procedemos a cerrar la conexión.
30. De la clase "Base()", acceso al método "close()", cierra la conexión con la base de datos.
31. Si el identificador se encontró en la DB de la comunidad, se retorna su información. Si el usuario no existe en la DB, se realiza otra consulta, a la DB de las visitas.
32. Si el usuario no existe en ninguna de las DB, se retorna un mensaje de erro.
33. Del controlador "Home", enviamos la respuesta de la búsqueda del identificador a la vista.
34. La vista, muestra la información en pantalla, dependiendo de la respuesta de la consulta, esta puede mostrar "Fotografía, nombre completo y carrera", o en dado caso, mostrar "El QR no existe".
35. El vigilante puede cerrar su cesión. Esto se realiza al presionar un botón, el cual enviara por url, el controlador y el método que tiene que cargar, para destruir la sesión.
36. Se accede al controlador "Pagians".
37. Del controlador "Paginas", accedemos al método "exit()", este destruirá la sesión iniciada, y retornara a la vista por defecto.

38. El vigilante puede ir a la vista, para registrar a un visitante. Esto se realiza al presionar un botón, el cual enviara por url, el controlador y el método que tiene que cargar, para mostrara la vista con el formulario.
39. Se envía por url, el controlador y el método a cargar.
40. Cargamos el controlador "Registrar".
41. Dl controlador "Registrar", se carga el método "index()".
42. El método "index()", del controlador "Registrar", carga la vista index.

## 1.8. Registrar->index()



| Diagrama | Registrar->index() |
|---|---|
| Para registrar a un visitante, o alguien de la comunidad que olvido su credencial. | |

**Descripción**

1. De la clase controlador, cargamos el controlador "Registrar", y establecemos el método a cargar.
2. Del controlador "Registrar", cargamos el modelo "Usuario".
3. Del controlador "Registrar", cargamos el método "index()", establecido por la clase Controlador().
4. Del método "index()", cargamos la vista index.
5. De la vista "index", mostramos el formulario.
6. Dentro del formulario, habilitamos una opción para registro, con esta determinaremos si es necesario cargar el archivo de "Identificación oficial".
7. En caso de ser necesario, se habilita la opción para seleccionar un archivo de tipo imagen.
8. Del formulario, retornamos un array, este contiene los campos del formulario.

9. Del modelo "Usuario", enviamos el array del formulario al método "guardaVisitante($datos", donde el parámetro $datos, es el array.
10. Creamos un objeto de la clase "Sanitiza()".
11. Del objeto de la clase "Sanitiza()", enviamos uno por uno los datos de tipo texto al método "sanitize_input($data)", el parámetro $data, son los datos de tipo texto.
12. Con una condición, verificamos si se seleccionó la opción de imagen del formulario.
13. Si se selecciono la opción para registrar con imagen. Verificamos que el tipo de archivo es de tipo imagen (.jpg o .png). En caso contrario retornamos un valor false.
14. Verificamos que el peso del archivo sea menor a 5MB. En caso contrario retornamos un valor false.
15. Mueve la imagen a una carpeta del servidor. En caso contrario retornamos un valor false.
16. Si es requerido, creamos un código QR, en caso contrario, retornamos una respuesta true, indicando que la imagen se movió con éxito, el nuevo nombre de la imagen en el servidor y su ruta.
17. Solo si es necesario crear un QR, creamos un identificador único, el cual almacenara en el QR.
18. Creamos un objeto de la clase "Codigo()", este servirá para crear un identificador único para el visitante.
19. De la clase "Codigo", cargamos el método "codeRandom($inicio, $limite), mediante los parámetros, podremos establecer la longitud del identificador. Retornamos el identificador de tipo numérico.
20. Creamos un objeto de la clase "QR()".
21. De la clase "QR()", cargamos el método "newQR($identificador)", el parámetro, será el identificador único que tendrá el QR.
22. Mueve la imagen QR a una carpeta del servidor. Retornamos el nuevo nombre de la imagen en el servidor y su ruta.
23. De la clase "Base()", acceso al método "__construct($host, $usuario, $password, $nombre_base)", con la que inicializo la conexión a la base de datos que almacena la información de acceso. El parámetro "$host", es la ruta del servidor, "$usuario", usuario con el que se realizara consultas, "$password", es la contraseña del usuario y "$nombre_base", es el nombre de la base de datos a la cual nos conectaremos.
24. De la clase "Base()", acceso al método "query($query)", establece la consulta.
25. De la clase "Base()", acceso al método "bind()", vincula el valor de la consulta con el parámetro recibido de formulario.
26. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.
27. Una vez guardado los datos del visitante en la DB, volvemos a realizar una consulta, para obtener los datos almacenados en la DB y mostrarlos a la vista.
28. De la clase "Base()", acceso al método "query($query)", establece la consulta.
29. De la clase "Base()", acceso al método "bind()", vincula el valor de la consulta con el parámetro recibido de formulario.
30. De la clase "Base()", acceso al método "execute()", ejecuta la consulta.
31. De a clase "Base()", accedemos al método "registro". Con esto retornamos la información del visitante que se acaba de registrar.
32. De la clase "Base()", acceso al método "close()", cierra la conexión con la base de datos.
33. Retornamos la información previamente almacenada del visitante.
34. Del método "index()", mostramos la información a la vista index.

35. En la vista index, mostramos la identificación oficial (dependiendo el caso), nombre completo, motivo de visita, fecha en que se registro y la opción para imprimir su comprobante, el cual le permitirá acceder al plantel.
36. Del modelo "Usuario", cargamos el método "getPdf($data)", con el cual generaríamos un PDF del visitante. El parámetro $data, es un array, el cual contiene la información del visitante.
37. El vigilante puede cerrar su cesión. Esto se realiza al presionar un botón, el cual enviara por url, el controlador y el método que tiene que cargar, para destruir la sesión.
38. Se accede al controlador "Pagians".
39. Del controlador "Paginas", accedemos al método "exit()", este destruirá la sesión iniciada, y retornara a la vista por defecto.
40. El vigilante puede ir a la vista, escanear un código QR. Esto se realiza al presionar un botón, el cual enviara por url, el controlador y el método que tiene que cargar, para mostrar la vista con el formulario.
41. Se carga el controlador "Home".
42. Del controlador "Home", se carga el método "index()".
43. Del método "index()", se carga el modelo "Usuario".
44. Del método "index()", se carga la vista index.

## 2. Diagrama de clases

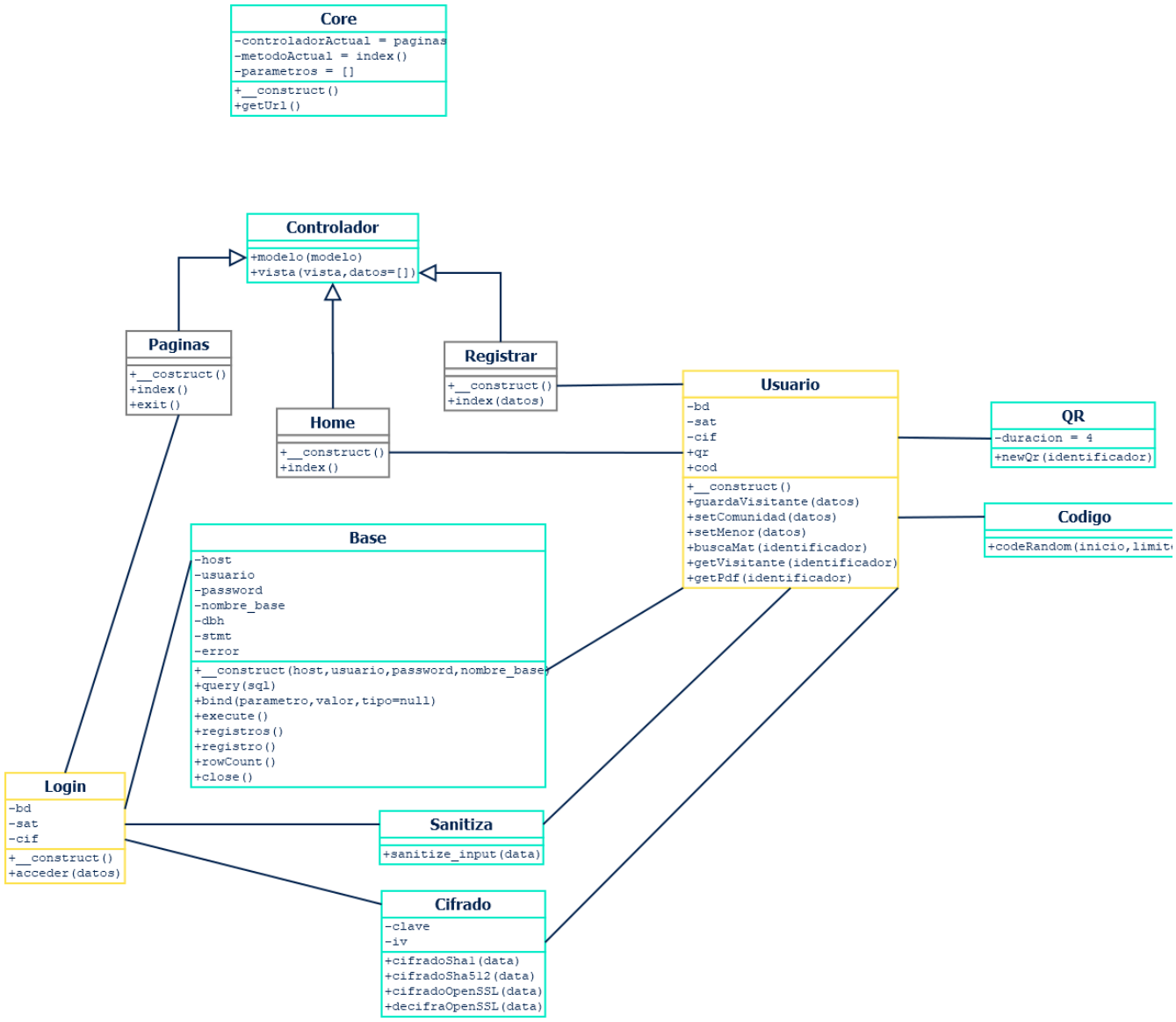



Diagrama de clases V-4.1

## 3. Herramientas

### 3.1. Diagrama de clases

Se utiliza la herramienta siguiente, Dia.

### 3.2. Diagrama se secuencia

Se utiliza la herramienta siguiente, Dia.

## 4. Definiciones, acrónimos y abreviaturas

- **SAR:** "Sistema de Acceso Rápido" (**SAR**).
- **UACM:** Universidad Autónoma de la Ciudad de México.
- **Comunidad:** Estudiantes y trabajadores.
- **Trabajadores:** Personal docente e investigador, Personal de administración y servicios y Personal de vigilancia.
- **Campus:** Área de instalaciones universitarias donde se realizan actividades académicas y administrativas.
- **Servidor:** Sistema informático que proporciona recursos y servicios a otros ordenadores a través de una red.
- **Base de Datos:** Conjunto organizado de datos almacenados electrónicamente, permitiendo su gestión y actualización.
- **Normativas:** Reglas y directrices establecidas por una autoridad para regular comportamientos y acciones.
- **Políticas:** Normas que regulan las actividades y comportamiento dentro de la institución.
- **UI (User Interface):** UI significa Interfaz de Usuario. Se refiere a la parte del software con la que los usuarios interactúan directamente. El diseño de UI se enfoca en la disposición visual y la presentación de los elementos en la pantalla.
- **UX (Experiencia de Usuario):** UX Se refiere a la experiencia general del usuario al interactuar con el software. El diseño de UX abarca aspectos más amplios que solo la apariencia y se centra en cómo se siente el usuario durante el uso del producto.
- **QA (Aseguramiento de la Calidad):** Es un proceso integral que se enfoca en asegurar que el software cumpla con los estándares de calidad y que funcione correctamente según los requisitos definidos.
- **Formador:** Es un profesional encargado de capacitar a los usuarios, desarrolladores, y otros miembros del equipo sobre el uso de software, herramientas o metodologías específicas.
- **Visitante:** Usuario final, el cual no pertenece de ninguna manera al plantel educativo.
- **DB:** Base de datos de la UACM.
- **Usuario:** Persona que desea acceder a la institución educativa.

## 5. Bibliografía


- A.U.S. Gustavo Torossi. Diseño de Sistemas. El proceso unificado de desarrollo de Software.
- Cervantes, Velasco, Castro; Arquitectura de Software. Conceptos y Ciclo de Desarrollo; Cengage Learning, 2016.