UACM

Diseño de software

Diagramas de clases

SAR

Valadez Carmona Guadalupe Yamileth

Rodríguez Cervantes Kevin Manzur

Cruz Ovando Cristela Adelaida

HISTORIAL DE VERSIONES

FECHA	VERSIÓN	DESCRIPCIÓN	AUTOR@S
30/01/2025	4.00	⊦ Nombre del sistema. ⊦ Actualización de los integrantes. ⊦ Logo para el proyecto. ⊦ Diagrama de clases. ⊦ Diagrama de controladores. ⊦ Diagrama de modelos.	Guadalupe Yamileth, Manzur Rodriguez, Cristela Adelaida
31/01/2025	4.10	⊦ Nuevo formato del documento.	Guadalupe Yamileth, Manzur Rodriguez, Cristela Adelaida
21/02/2025	4.20	⊦ Actualización del documento, siguiente el 'Estándar de Documentación V - 2.0'	Manzur Rodriguez

INDICE

1. Clases

1.1. 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.

Clase		Base()
Objetos		Descripción
-		-
Modificadores	Atributo	Descripción
private	host	Host del servidor.
private	usuario	Nombre de usuario de la DB.
private	password	Contraseña de la DB.
private	nombre_base	Nombre de la DB.
private	dbh	database hander.
private	stmt	Stmt
private	error	Error de la conexión.

Método	Descripción
`__construct($host, $usuario, $password, $nombre_base)`	Constructor que permite seleccionar la base de datos dinámicamente.
`query($sql)`	Preparamos la consulta.
`bind($parametro, $valor, $tipo = null)`	Vinculamos la consulta con bind.
`execute()`	Ejecuta la consulta.
`registros()`	Obtiene todos los registros de la consulta actual.
`registro()`	Obtener un solo registro.
`rowCount()`	Obtener cantidad de filas con el método rowCoun
`close()`	Cierra la conexión con la base de datos.

Requisitos

- Conexión a las diferentes DB de una manera dinámica.
- Antes de ejecutar la consulta, es necesario que los datos sean formateados, para evitar una inyección SQL.
- Evitar inyecciones o algún ataque a la DB.

Riesgos

- No existe un manejo de errores al conectarte a la DB.
- El usuario asignado para realizar consultas en la base de datos, no tiene permisos por parte de los administradores.
- Acceder a una tabla, la cual no tenemos acceso.
- Manejo de eventos, en caso que alguna DB no se encuentre disponible.

1.2. Cifrado()

Ciframos y desciframos la información de la DB.

Clase		Cifrado()
Objetos		Descripción
-		-
Modificadores	Atributo	Descripción
private	clave	Clave de 32 caracteres para el cifrado OpenSSL.
private	iv	Vector de inicialización (IV), para cifrado OpenSSL.
Método		Descripción
`cifradoSha1($data)`		Cifrado irreversible con sha512
`cifradoSha512($data)`		Cifrado único con sha512, con una longitud de 129 caracteres.
`cifradoOpenSSL($data)`		Cifra la información con OpenSSL, esta es reversible.
`decifraOpenSSL($data)`		Desciframos la información, de cifrado OpenSSL.
Requisitos		
› Tipo de cifrado para almacenar la información de los visitantes. › Método de cifrado, para comparar la información "usuario" y "contraseña", con el que se crea una sesión. › Clave y vector de inicialización (IV) estáticos para el método de cifrado OpenSSL.		
Riesgos		
› Clave y vector de inicialización (IV) estáticos para cifrado OpenSSL (asegúrate de usar las longitudes correctas). › Información cifrada con un método diferente, el cual no este implementado. › Se cifre la información de visitantes, con un método no reversible.		

1.3. Controlador()

Clase controlador principal. Se encarga de poder cargar los modelos y las vistas.

Clase		Controlador()
Objetos		Descripción
-		-
Modificadores	Atributo	Descripción
-	-	-
Método		Descripción
`modelo($modelo)`		Carga el modelo.
`vista($vista, $datos = [])`		Carga a vista.
Requisitos		
› Cargar el modelo correcto.		

› Si no existe la vista, deberá cargar una vista de error. › Array para mostrar los datos en la vista.
Riesgos
› Mostrar un mensaje de error, en caso que la vista no exista. › No cargar los datos a mostrar para las diferentes vistas. › Error al establecer la ruta de la carpeta de las vistas.

1.4. Core()

Controla el sistema (núcleo del frame-work), establece la vista por defecto del sistema.

Clase		Core()
Objetos		**Descripción**
-		-
Modificadores	**Atributo**	**Descripción**
private	$controladorActual	Controlador por defecto.
private	$metodoActual	Método por defecto.
private	$parametros	Array vacío por defecto, para los parámetros.
Método		**Descripción**
`__construct()`		Establecemos el controlador y el método por defecto, en caso de que no se establezca nada por la url. Si se establece algo por la url, se redirecciona al modelo y la vista.
`getUrl()`		Comprobamos si hay algo en la url del sitio, si esta la ruta del sitio, no retorna nada. En caso de existir algo en la ruta de la url, esta se descompone en un array, utilizando el carácter '/' para separarla en el array.
Requisitos		
› Establecer la vista por defecto. › El controlador de la vista por defecto, tiene que tener el método index(). › Mapear la url ingresada en el navegador: controlador, método y parámetro por la url del navegador.		
Riesgos		
› Vista por defecto no existe. › Método no existe. › Error al mapear los datos de la url.		

1.5. Sanitiza()

Formatea (sanitiza) la información para guardarla en la DB, evitando inyecciones.

Clase	Sanitiza()

Objetos		Descripción
-		-
Modificadores	**Atributo**	**Descripción**
Método		**Descripción**
`sanitize_input($data)`		Formatea (Sanitiza) la información para realizar una consulta en la DB.
Requisitos		
› Formatear la información de manera correcta. › Formatear la información, de manera que se evite alguna inyección SQL. › Eliminar espacios vacíos al final de la cadena.		
Riesgos		
› Cambiar caracteres de la información.		

1.6. 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.

Clase		QR()
Objetos		**Descripción**
-		-
Modificadores	**Atributo**	**Descripción**
private	duracion	Duración por defecto de la visita, establece por defecto un valor de 4, el cual representa horas.
Método		**Descripción**
`newQr($identificador)`		Genera un QR, el cual almacena un enlace web.
Requisitos		
› Genera códigos QR únicos. › Los códigos QR son generados aleatoriamente, tomando la hora del sistema. › Códigos QR utilizados para visitantes. › Permite generar nuevos códigos QR para invitados con una duración de acceso de 4 hora.		
Riesgos		
› Error a generar un QR para el visitante. › Le asigna una duración mayor a 4, para el QR. › Los QR se repiten, ya que no toma la hora del sistema para generarlos aleatoriamente.		

1.7. Codigo()

Genera códigos aleatorios de tipo numéricos utilizando combinaciones.

Clase		Codigo()
Objetos		Descripción
-		-
Modificadores	Atributo	Descripción
-	-	-
Método		Descripción
`codeRandom($inicio, $limite)`		Genera un código numérico de una longitud definida.
Requisitos		
› Repetición de los códigos.		
Riesgos		
› Los códigos generados se repiten. › Genera números de manera lineal, y no aleatoria.		

2. Controladores

2.1. Paginas()

Controlador por defecto del sistema, se muestra al iniciar al sistema.

Clase - Controlador	Paginas extends Controlador
Vista	Index
Modelo	Descripción
Login	Crear una sesión en el sistema, con el "usuario" y "contraseña" del vigilante.
Parámetro	Descripción
-	-
Método	Descripción
__construct()	Inicializa el modelo (Login).
index()	Muestra la vista principal del sistema.
exit()	Cierra la sesión del sistema.
Requisitos	
› Formulario para crear una sesión en el sistema. › El formulario requiere los datos "usuario" y "contraseña". › El dato "usuario" y "contraseña", tiene una longitud máxima de 20 caracteres. › Verificar si la información ingresada coincide con la DB. › Función para cerrar-eliminar la sesión iniciada en el sistema. › Método para enviar los datos ingresados del formulario a través de POST.	
Riesgos	
› Se ingresen datos vacíos.	

› Posibles inyecciones SQL. › Error con la conexión de la DB. › No eliminar correctamente la sesión inicializada anteriormente. › No crear una sesión, cuando se ingresen los datos del formulario de manera correcta.

2.2. Home()

Muestra la información guardada en los códigos QR, trabaja con el escáner
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.

Clase - Controlador	Home extends Controlador
Vista	Index
Modelo	Descripción
Usuario	Se envía el identificador del QR, compara si pertenece a la comunidad o es visitante. Retorna la información del usuario o el mensaje de error.
Parámetro	Descripción
-	-
Método	Descripción
__construct()	Inicializa el modelo (Usuario).
index()	Formulario oculto para escanear el QR. Activa los campos para mostrar la información retornada del modelo.
Requisitos	
› Enviar el formulario sin la necesidad de recargar la página. › El formulario envía datos por el método POST. › El escáner, ingresara una url, con una longitud máxima de 80 caracteres.	
Riesgos	
› El escáner, ingrese la url mayor a la longitud establecida. › Error al escanear el QR.	

2.3. Registrar()

Para registrar a un visitante, o alguien de la comunidad que olvido su credencial.

Clase - Controlador	Registrar extends Controlador
Vista	Index historial
Modelo	Descripción
Usuario	Registrar a un visitante.

	Si un usuario pertenece a la comunidad.
Parámetro	Descripción
-	-
Método	Descripción
__construct()	Instanciamos un objeto a los parámetros creados.
index($datos)	Comprobamos si los datos ingresados en el formulario son correctos, con lo que podemos crear una nueva sesión, o en otro caso, retornar un mensaje de erro.
Requisitos	
› Diseñar un formulario, que permita registrar la información de un visitante, cumpliendo los requisitos establecidos. El formulario contiene los campos: ○ Entrada para el nombre. ○ Entrada para el apellido paterno. ○ Entrada para el apellido materno. ○ Entrada para establecer el motivo. ○ Botón que permita seleccionar una imagen de los archivos del dispositivo. El tipo permitido del archivo: ".jpg, .jpeg, .png", de un máximo de 5 megas. › Botón para mostrar el formulario de un visitante. › Botón para comprobar, si un usuario pertenece a la comunidad. › Botón para mostrar el formulario, para el caso que un menor de edad dese entrar al plantel, este tiene que venir acompañado de un tutor. › Botón para regresar a la vista home. › Se registrará el identificador de la sesión, para los registros de los visitantes. › Después de 3 segundos, limpia los campos de texto y el contenido de la imagen lo hace nulo. › Se le solicitara al visitante, que indique cuál es su motivo de visita.	
Riesgos	
› No pedir la identificación oficial al registrar a un visitante. › Seleccionar un archivo que no sea un formato de imagen valido.	

3. Modelos

3.1. Login()

Realiza una comprobación con los datos “usuario” y “contraseña”, de ser verdadera, crea una sesión y permite entrar al sistema. En caso contrario, retornara un mensaje de error.

Clase - Modelo	Login
Parámetro	Descripción
$db	Conexión con la DB.
$sat	Formatea la información recibida.

$cif	Cifra la información.
Método	**Descripción**
__construct()	Instanciamos un objeto a los parámetros creados.
acceder($datos)	Comprobamos si los datos ingresados en el formulario son correctos, con lo que podemos crear una nueva sesión, o en otro caso, retornar un mensaje de error.
Requisitos	
› Cifrar los datos ingresados en el formulario, para proceder a compararlos con los almacenados en la DB. › Formatear de manera correcto los datos ingresados. › Retornar una respuesta, para indicarle al usuario si sus datos son correctos. › Asegurar que los datos ingresados, no están vacíos.	
Riesgos	
› Error al cifrar los datos. › No retornar una respuesta valida. › Error al crear una sesión valida.	

3.2. Usuario()

Realiza una comprobación, para establecer si un usuario pertenece a la comunidad o un visitante. Esta retorna información diferente, dependiendo del resultado de la consulta.

Clase - Modelo	Usuario
Parámetro	**Descripción**
$db	Conexión con la DB.
$sat	Formatea la información recibida.
$cif	Cifra la información.
$qr	Generar un nuevo QR.
$cod	Genera un código aletario.
Método	**Descripción**
__construct()	Instanciamos un objeto a los parámetros creados.
guardaVisitante($datos)	Realizamos una comprobación, para evitar guardar información con campos vacíos. Si todos los campos de texto están llenos y la imagen ya fue seleccionada, se guarda la información en la DB de visitantes.
setComunidad($datos)	Si alguien de la comunidad olvido su credencial, se registra como visitante, pero no es necesario la identificación oficial.

buscaMat($identificador)	Realiza una consulta en las diferentes bases, determina si el identificador aparece en una determinada base de datos. La información obtenida la retornamos a la vista.
getVisitante($identificador)	Retorna la información encontrada de un Visitante, la muestra en la vista.
getPdf($data)	Obtiene la información de un visitante, y genera un PDF con los datos (Nombre completo, motivo y QR).

Requisitos

- Cifrar los datos ingresados en el formulario, para proceder a compararlos con los almacenados en la DB.
- Formatear de manera correcto los datos ingresados.
- Retornar una respuesta, para indicarle al usuario si sus datos son correctos.
- Asegurar que los datos ingresados, no están vacíos.
- Se mostrará información de los visitantes registrados en el día, la información a mostrar es el nombre, hora en que se registró, además, cada registro debe contener el botón para generar un PDF con la información registrada.
- Las imágenes escaneadas, se le aplicara un formato de compresión de imágenes. Con el objetivo de disminuir su peso.
- Antes de guardar la información de un visitante, se tiene que generar un QR, el cual se almacenará en la carpeta designada.
- Designar la carpeta, en donde se guardará la identificación oficial de un visitante.
- Designar el formato de carpeta, en el cual se guardarán los códigos QR.
- Establecer la dirección de la carpeta, en donde se almacenan las imágenes de la comunidad.

Riesgos

- Error al descifrar los datos.
- No retornar una respuesta valida.
- No retornar la información de un usuario a la vista.
- Datos de la DB, más grandes de los contemplados.
- El archivo de imagen, pesa más de lo requerido.
- Corrompe el archivo, al realizar a compresión de la imagen.
- No guardar le archivo en a carpeta.

4. Diagrama de clases

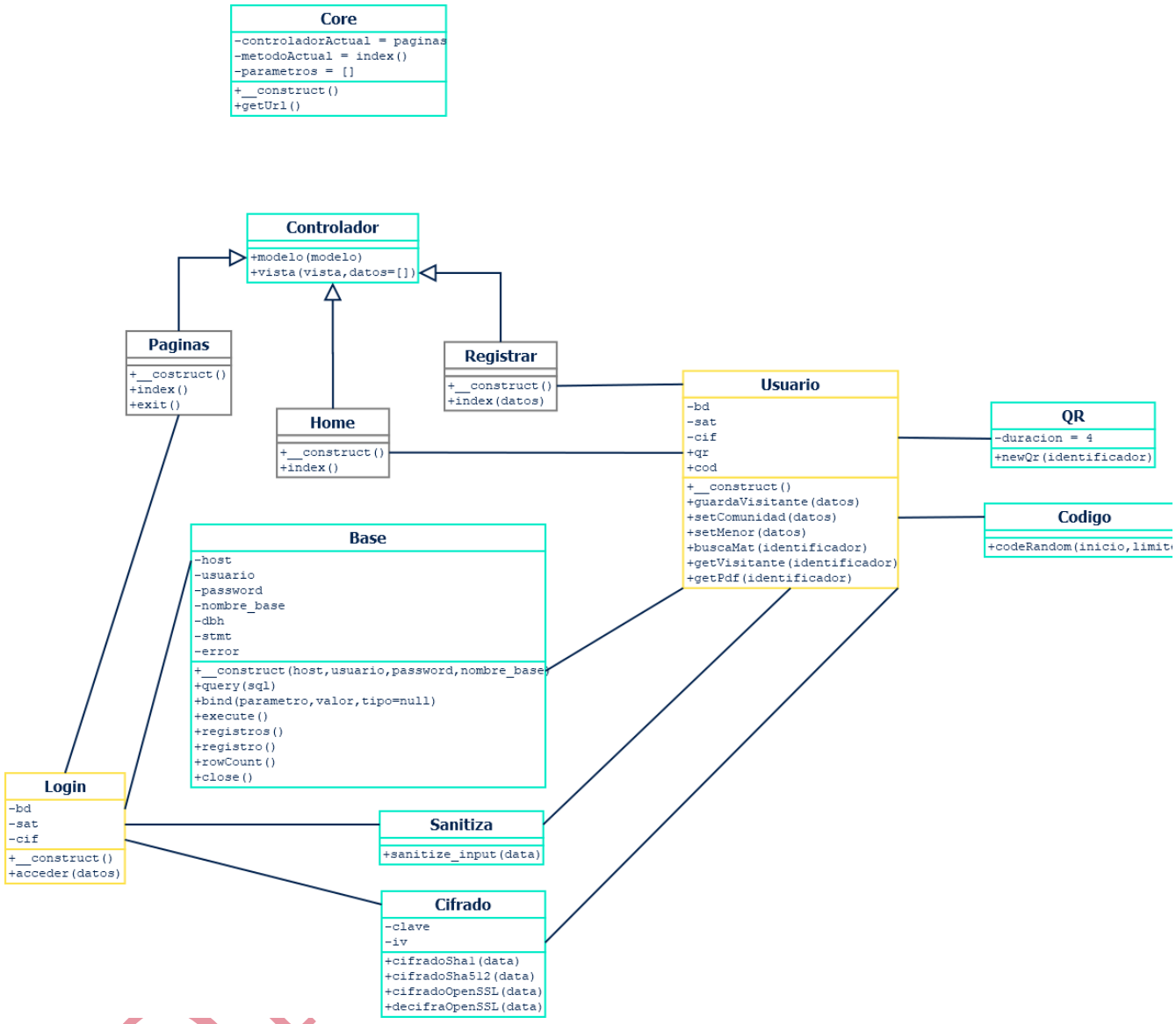

Diagrama de clases V-4.1

5. Herramientas

5.1. Diagrama de clases

Se utiliza la herramienta siguiente, Dia.

5.2. Diagrama se secuencia

Se utiliza la herramienta siguiente, Dia.

6. 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.

7. 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.