

# Login con QR

Samir Banquez Humanez - Diana Humanez - Santiago Quintero- Arianna Espitia - Steven Cuello Vásquez

**Tutor:** Alexander Toscano Ricardo.

---



## Reseña

La mayoría de los sistemas de inicio de sesión actuales dependen de unas credenciales (usuario y contraseña), que a menudo son olvidadas por los usuarios. El componente *Login con QR*, propone una solución amigable que satisface las necesidades y expectativas de los usuarios, en particular aquellos con limitaciones visuales y quienes tienden a olvidar sus contraseñas con facilidad. Este enfoque de inicio de sesión simplificado ofrece comodidad y accesibilidad a un amplio público, abriendo nuevas posibilidades para una experiencia de usuario más inclusiva y eficiente.

## Tabla de contenido

<b>ETAPA 1 DISEÑO DE LA APLICACIÓN Y ANÁLISIS DE REQUISITOS</b>	<b>4</b>
<b>1. INTRODUCCIÓN</b>	<b>4</b>
Propósito del proyecto	4
Alcance del proyecto	5
Definiciones y acrónimos	6
<b>2. DESCRIPCIÓN GENERAL</b>	<b>7</b>
Objetivos del sistema	7
Funcionalidad General	8
Usuarios del sistema	8
Restricciones	9
<b>3. REQUISITOS NO FUNCIONALES</b>	<b>9</b>
Requisitos de Desempeño	9
Requisitos de Seguridad	10
Requisitos de Usabilidad	10
Requisitos de Escalabilidad	10
<b>4. REQUISITOS FUNCIONALES</b>	<b>11</b>
Casos de uso	12
Diagramas de flujo de casos de uso	14
Descripción detallada de cada caso de uso	17
Prioridad de requisitos	24
<b>5. Modelado E/R</b>	<b>25</b>
<b>Etapas 3: Consumo de Datos y Desarrollo Frontend</b>	<b>27</b>
Introducción	27

# ETAPA 1 DISEÑO DE LA APLICACIÓN Y ANÁLISIS DE REQUISITOS

## 1. INTRODUCCIÓN

### **Propósito del proyecto**

"Login con QR" es un proyecto de diseño de software educativo I, II, III, que pretende mejorar el proceso de registro y acceso a la plataforma CREAVI a través de imágenes QR. La forma convencional de acceder a una cuenta en cualquier programa informático es mediante un nombre de usuario y una contraseña escrita, lo cual es problemático para los usuarios olvidadizos y, en particular, para aquellos con discapacidad visual.

Este documento, pretende documentar el proceso de diseño, análisis e implementación del componente de aplicación login con Qr en la plataforma CREAVI. Este proceso se organiza en tres fases o etapas, cada curso de diseño de software educativo desarrolla una de estas dichas fases:

### **- Fase 1: Diseño de la Aplicación y Análisis de Requisitos**

Esta fase desarrolla el diseño y documentación necesaria para llevar a cabo la propuesta del componente de logueo con QR. Este proceso se enmarca en la metodología Sprint de desarrollo de software, integrando las habilidades adquiridas en la Lic. informática y medios audiovisuales para desarrollar un producto innovador que vaya acorde a las necesidades del software CREAVI, que a su vez es un programa informático de carácter educativo, que busca que sus usuarios tengan experiencias amigables.

### **-Fase 2: Persistencia de Datos con Backend – Servidor**

Esta fase pone en marcha el diseño y la implementación de software desde la fase anterior, centrándose en la programación del componente de login con QR, desarrollándose la estructura, servidores o microservicios para respaldar las aplicaciones cliente del software educativo. Se cubren los conceptos de sistemas de bases de datos, incluido el diseño lógico, la organización de sistemas de gestión de bases de datos, así como los lenguajes de definición y manipulación de datos SQL y NoSQL.

### **- Fase 3: Consumo de Datos y Desarrollo Frontend – Cliente**

Aquí se seleccionan las herramientas de consumo de datos y técnicas más adecuadas para lograr un producto óptimo en términos de software o hardware, de acuerdo con los requisitos funcionales y no funcionales del componente de login con Qr. En esta fase el diseño visual o gráfico del componente, es un requisito esencial en la capa de presentación del producto.

## **Alcance del proyecto**

El objetivo principal del proyecto es diseñar, desarrollar e implementar un método de inicio de sesión basado en la lectura de códigos QR a través de la cámara del cliente. Esta innovadora aproximación permitirá a los usuarios acceder a su cuenta en CREAVI de manera eficiente, rápida y segura.

Este proyecto se enfoca en atender dos grupos específicos de usuarios: en primer lugar, las personas con discapacidades visuales, quienes se beneficiarán de una solución accesible y amigable que elimina las barreras asociadas con los métodos de inicio de sesión convencionales basados en nombre de usuario y contraseña. En segundo lugar, los usuarios propensos a olvidar sus contraseñas encontrarán en esta solución un proceso simplificado, ya que el escaneo de un código QR reemplazará la necesidad de recordar contraseñas específicas y simplifica el proceso de inicio de sesión,

Para lograr estos objetivos, el proyecto abordará todas las etapas esenciales, comenzando con la investigación y análisis de las tecnologías relacionadas con la lectura de códigos QR. A continuación, se llevará a cabo el diseño de la interfaz de usuario y la experiencia de usuario, priorizando la accesibilidad y la usabilidad. El desarrollo del sistema de inicio de sesión basado en códigos QR será una fase central del proyecto, seguida de pruebas exhaustivas para garantizar la funcionalidad y seguridad del sistema.

Finalmente, se llevará a cabo la implementación de la solución, en la plataforma de CREAVI. La solución propuesta no solo busca ser eficiente y segura, sino también contribuir al acceso igualitario de servicios en línea para una amplia gama de usuarios, sin importar sus limitaciones visuales o problemas de memoria, y buscar así ser un método distintivo y con menos barreras. La implementación de este método de inicio de sesión presenta las siguientes funcionalidades clave:

**Escaneo Eficiente de Códigos QR:** Los usuarios podrán acceder a sus cuentas de CREAVI de manera eficiente y segura mediante la lectura de un código QR a través de la cámara de su dispositivo. Este proceso simplificado elimina la necesidad de ingresar manualmente un nombre de usuario y una contraseña, agilizando el acceso a la plataforma.

**Autenticación en Tiempo Real:** La funcionalidad de escaneo de códigos QR garantiza una autenticación instantánea y en tiempo real. Una vez que se escanea el código QR, el sistema verifica la información y permite el acceso prácticamente de inmediato.

**Acceso Inclusivo:** Este método de inicio de sesión se ha diseñado teniendo en cuenta la inclusión y accesibilidad. Ofrece una solución particularmente beneficiosa para personas con discapacidades visuales y aquellos que tienden a olvidar sus contraseñas.

**Seguridad Reforzada:** A pesar de su simplicidad, el sistema de códigos QR genera códigos únicos y difíciles de falsificar, lo que garantiza la protección de las cuentas de usuario y mantiene altos estándares de seguridad.

**Registro y Seguimiento de Acceso:** El sistema mantendrá un registro de cada acceso exitoso a través del escaneo de códigos QR, lo que brinda a los usuarios un historial detallado de sus sesiones y una capa adicional de seguridad.

### **Funcionalidades Futuras (Propuestas):**

Aunque la implementación inicial se centrará en las funcionalidades anteriores, se proponen características adicionales para futuras versiones del proyecto:

**Opciones de Recuperación Avanzadas:** Se explorará la implementación de procedimientos de recuperación adicionales para abordar las posibles dificultades de los usuarios al escanear códigos QR, garantizando una experiencia sin obstáculos.

**Integración con Herramientas de Colaboración:** Se considerará la posibilidad de integrar el sistema de inicio de sesión con código QR con herramientas de colaboración para permitir el trabajo conjunto en proyectos y documentos compartidos.

**Personalización de Perfiles:** Se evaluará la inclusión de opciones de personalización de perfiles de usuario para mejorar la experiencia individual en CREAVI.

### **Definiciones y acrónimos**

Este proyecto utiliza varias definiciones y abreviaturas para comprender la tecnología y el concepto de un sistema de entrada de códigos QR. Aquí están las definiciones y abreviaturas:

1. **Código QR:** Un código QR es un código de barras bidimensional que almacena información en un formato legible por máquina. Normalmente se utiliza para almacenar URL, texto, números de serie y otros tipos de información.
2. **Inicio de sesión:** El proceso mediante el cual un usuario obtiene acceso a una cuenta o sistema seguro. Normalmente, proporcionará información como un nombre de usuario y contraseña para verificar su identidad.
3. **Accesibilidad:** La capacidad de las personas con discapacidad, incluidas las personas con discapacidad visual, para utilizar eficazmente un sistema, producto o entorno.
4. **Interfaz de usuario (UI):** la parte de un sistema o aplicación que permite a los usuarios interactuar con él. Esto incluye elementos visuales como botones, formularios y menús.

5. **Experiencia de usuario (UX):** La experiencia general del usuario al interactuar con un sistema o aplicación, incluidos los aspectos emocionales y prácticos.
6. **Privacidad:** proteja la información personal y permita a los usuarios controlar cómo se usa y comparte su información.
7. **Seguridad:** Proteger la integridad y confidencialidad de los datos y sistemas para evitar el acceso no autorizado o el uso indebido.
8. **CREAVI:** El nombre de la plataforma o sistema que crea este proyecto de comunicación en base al código QR.

## 2. DESCRIPCIÓN GENERAL

### Objetivos del sistema

El objetivo del sistema es mejorar la accesibilidad y la experiencia de usuario en el proceso de inicio de sesión en la plataforma CREAVI mediante la implementación de un sistema de login basado en QR. Este sistema está diseñado específicamente para beneficiar a los usuarios con discapacidad visual y a aquellos propensos a olvidar sus credenciales de acceso o de inicio de sesión. Al proporcionar este método de autenticación, se pretende simplificar el proceso de ingreso a la plataforma CREAVI, garantizando al mismo tiempo seguridad y eficiencia, mejorando a su vez, la interacción general con los servicios que la plataforma ofrece.

### Funcionalidad General

**Inicio de Sesión Rápido y Seguro:** La funcionalidad principal de este proyecto es permitir a los usuarios iniciar sesión de manera eficiente y segura utilizando la lectura de códigos QR, o con el método tradicional. Con el método de QR se elimina la necesidad de ingresar manualmente un nombre de usuario y contraseña, agilizando el proceso de autenticación y optimizando tiempos para ingresar a la plataforma.

**Escaneo de Código QR:** Los usuarios podrán escanear un código QR único generado por la plataforma para acceder a sus cuentas de CREAVI. Este proceso de escaneo se llevará a cabo a través de la cámara del dispositivo, lo que proporciona una experiencia de usuario rápida y sin fricciones.

**Generación de código QR:** Mediante la plataforma de CREAVI, se implementará dentro del componente la opción de generar un código QR único e intransferible por y para el usuario, para que así pueda realizar el ingreso a la plataforma mediado por su código QR.

**Registros de inicio de sesión:** El sistema registrará y documentará cada acceso exitoso mediante el escaneo de un código QR, lo que brinda a los usuarios un registro detallado de sus sesiones y una capa adicional de seguridad, además de información para mejorar versiones futuras.

**Generar nuevo código QR (recuperar):** Se proporcionarán procedimientos de recuperación en caso de que los usuarios tengan dificultades para escanear códigos QR, garantizando así una experiencia sin obstáculos, donde se brindará un nuevo código QR, en caso de pérdida o de cambio del mismo.

**Exportar el Código QR:** El método de ingreso ofrecerá la capacidad de descargar el código QR mediante archivos de tipo PDF, IMG, etcétera. Para así facilitar la accesibilidad al código y tenerlo a la mano de diversas maneras.

**Integración con la Plataforma CREAVI:** La funcionalidad de inicio de sesión con código QR se integrará de manera fluida con la plataforma CREAVI, permitiendo a los usuarios acceder a sus cuentas y recursos de forma rápida y sencilla.

### Usuarios del sistema

Los siguientes usuarios pueden interactuar con el inicio de sesión mediante QR, dependiendo de las funcionalidades.

Funcionalidad	Administrador	Docente Investigador	Estudiante
Iniciar Sesión			
Escanear Código QR			
Generar código QR			
Visualizar inicio de sesión			
Generar nuevo código QR (recuperar)			
Exportar el Código QR generado			

### Restricciones

El sistema de inicio de sesión con código QR puede estar sujeto a restricciones que incluyen el acceso limitado por roles específicos, la necesidad de registro y aprobación previa por parte de administrador o anfitrión, configuraciones personalizadas como horarios de acceso específicos, requisitos técnicos que implican la necesidad de dispositivos con cámaras, posibles pasos adicionales de autenticación en ciertos casos, límites en la cantidad de usos por período y políticas de seguridad como la caducidad de códigos QR o cambios regulares de contraseñas. Estas restricciones se implementarán de acuerdo con los requisitos del proyecto y las consideraciones de seguridad y accesibilidad.



### 3. REQUISITOS NO FUNCIONALES

#### Requisitos de Desempeño

- **Tiempo de Escaneo Rápido:** El sistema debe ser capaz de escanear y autenticar el código QR en un tiempo máximo de, por ejemplo, 2 segundos, para proporcionar una experiencia de usuario eficiente.
- **Compatibilidad Multiplataforma:** El sistema debe ser compatible con una variedad de dispositivos y sistemas operativos, incluyendo dispositivos móviles y de escritorio.
- **Tamaño de los Códigos QR:** El sistema debe ser capaz de escanear y procesar códigos QR de diferentes tamaños y resoluciones de manera eficiente.
- **Optimización de recursos:** El sistema debe ser capaz de solicitar al servidor los recursos necesarios para su ejecución, minimizando la exigencia del hardware empleado para su uso.

#### Requisitos de Seguridad

- **Integridad del código QR:** El componente generará código QR únicos y cifrados para prevenir el acceso no autorizado y posibles falsificaciones.
- **Autenticación segura:** garantizar la validación o autenticidad de los usuarios y prevenir accesos no autorizados, se implementará el uso de claves criptográficas.
- **Pruebas continuas de seguridad:** se realizarán pruebas de vulnerabilidad mediante falsificaciones de usuario ( Qr duplicados, similares , entre otros), para identificar posibles puntos débiles del componente de logeo , para su corrección oportuna, garantizando la fiabilidad y seguridad del sistema de logeo con imágenes Qr.
- **variables de entorno:** El componente de logeo cuenta con la capacidad de incorporarse con otros módulos, además, también podrá almacenar la información del mismo y migrar a otras plataformas.

#### Requisitos de Usabilidad

1. **Visibilidad:** El sistema debe ser totalmente visible para las personas con discapacidad visual. Proporciona soporte para lectores de pantalla para garantizar el cumplimiento de las pautas de contenido en línea.
2. **Navegación intuitiva.** La herramienta de Login con QR debe ser intuitiva y fácil para usuarios de todas las edades y niveles de experiencia. La necesidad de orientación externa debe disminuir.
3. **Respuesta del sistema:** Si no se establece ningún proceso, el sistema debe proporcionar comentarios precisos y oportunos a los usuarios después de cada escaneo de código QR.

## Requisitos de Escalabilidad

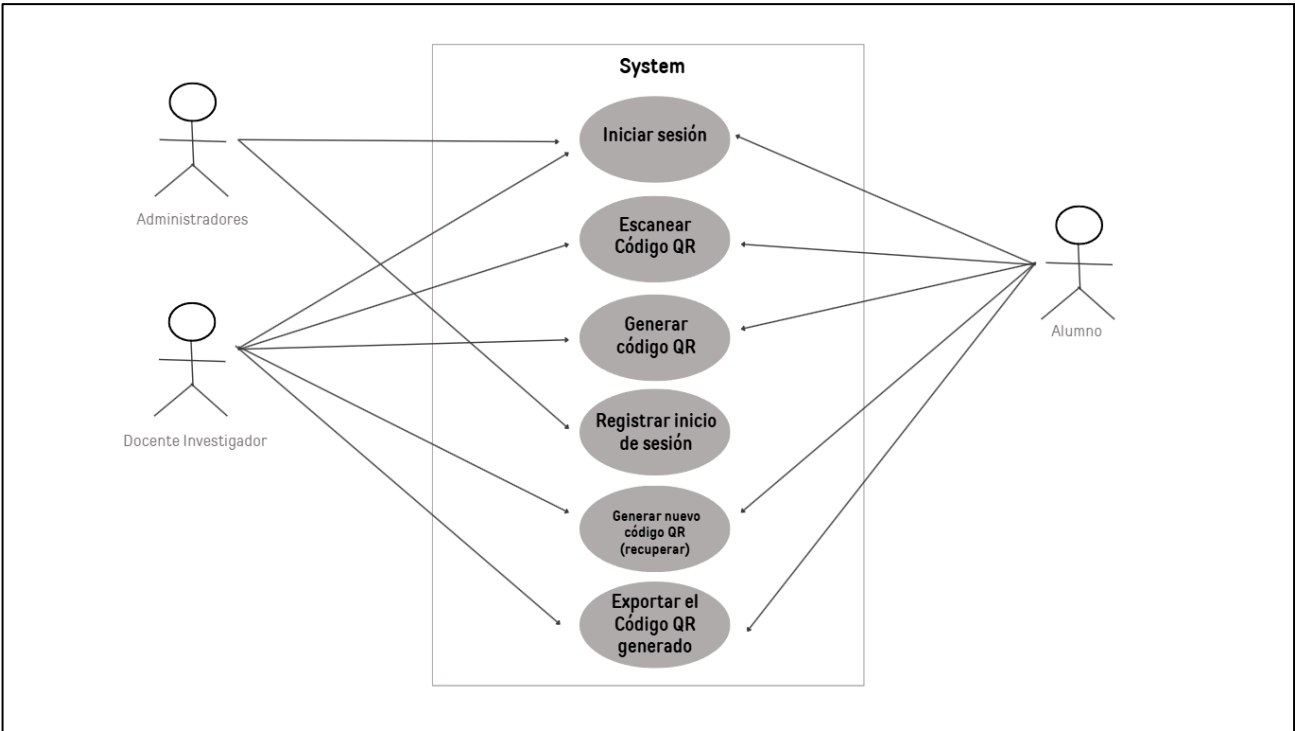
1. **La carga del usuario.** El sistema debería poder manejar una cantidad significativa de usuarios simultáneos sin una degradación significativa del rendimiento.
2. **Almacenamiento de datos:** el sistema debe ser escalable en términos de almacenamiento de datos. A medida que aumenta el número de usuarios y la cantidad de datos, el sistema de almacenamiento debe escalar de manera eficiente.
3. **Disponibilidad y Tolerancia a Fallos:** El sistema debe estar diseñado para garantizar la alta disponibilidad y la tolerancia a fallos. Debe ser capaz de mantener el servicio incluso en caso de fallas de hardware o software.

## 4.REQUISITOS FUNCIONALES

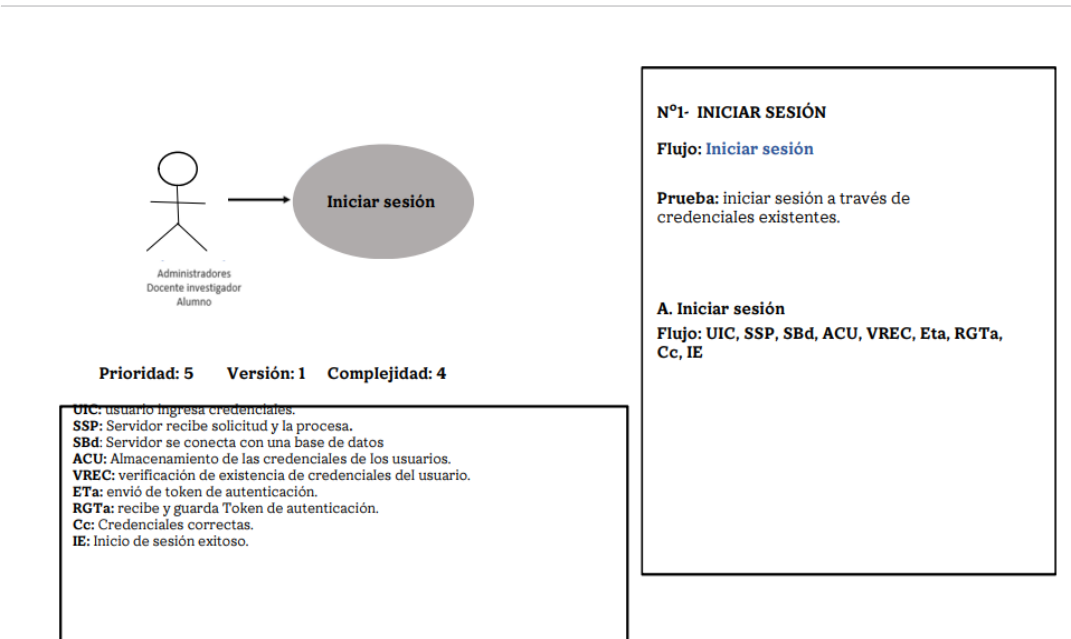
- **Iniciar sesión:**
  - Los usuarios ingresan sus credenciales (usuario y contraseña).
  - El servidor recibe la solicitud y la procesa.
  - La autenticidad de las credenciales es verificada.
  - Inicio de sesión exitoso.
- **Escanear Código QR:**
  - El usuario escanea su código QR delante una cámara.
  - La plataforma autentica el código QR escaneado.
- **Generar código QR:**
  - La plataforma generará códigos QR únicos asociados a cada cuenta de usuario registrada.
- **Registros de inicio de sesión:**
  - Registro de la hora de ingreso o acceso de cada usuario a la plataforma.
- **Generar nuevo código QR (recuperar):**
  - Los usuarios registrados podrán recuperar su cuenta en caso de olvido de credenciales y problemas de escaneo del código QR, con la generación de un nuevo código QR
- **Exportar el Código QR generado:**
  - Los usuarios podrán exportar o descargar el código QR asociado a su cuenta en diferentes formatos de imagen.

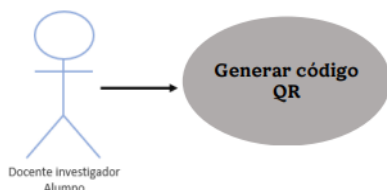
Casos de uso

Diagrama de casos de uso



Diagramas de flujo de casos de uso





**Prioridad: 5      Versión: 1      Complejidad: 5**

**SAaU:** Sesión activa y abierta del usuario.  
**CGCqr:** click en generar código QR.  
**SRCE:** Sistema redirige credenciales del usuario a una extensión  
**GCqrCE:** Generación de código QR con la credencial encriptadas del usuario.  
**SCET:** Sistema convierte credenciales encriptadas a texto.  
**SPC:** Sistema procesa las credenciales.  
**SRIEqr:** Sistema redirige la información a ESPORTAR QR.  
**UEF:** Usuario elige formato.  
**SERU:** Sistema envía resultado al usuario.  
**UDF:** Usuario descarga formato.

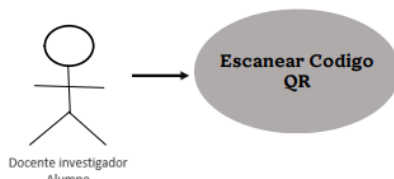
### Nº3- GENERAR CÓDIGO QR

**Flujo:** Generación de código Qr

**Prueba:** Generación de código Qr en formatos compatibles (pdf, IMG, GIF, etc.)

#### C. Generar código QR

**Flujo:** SAaU, CGCqr, SRCE, GCqrCE, SCET, SPC, SRIEqr, UEF, SERU, UDF



**Prioridad: 5      Versión: 1      Complejidad: 5**

**Iscqr:** Inicio de sesión mediante código Qr Escaneo de código QR.  
**SPC:** solicitud de permiso de cámara.  
**PA:** permiso aceptado.  
**Lqr:** lectura de QR.  
**ESDqr:** Envío de solicitud para descifrar Qr.  
**SDCqr:** Servidor descifra el código Qr.  
**SECqrdbd:** Servidor envía código Qr descifrado a la base de datos.  
**ScBd:** servidor se conecta con la base de datos.  
**VCBd:** Verificación de credenciales correctas en la base de datos.  
**NRES:** Navegador recibe respuesta y envía solicitud.  
**Cc:** Credenciales correctas.  
**UIP:** Usuario ingresa a la plataforma.

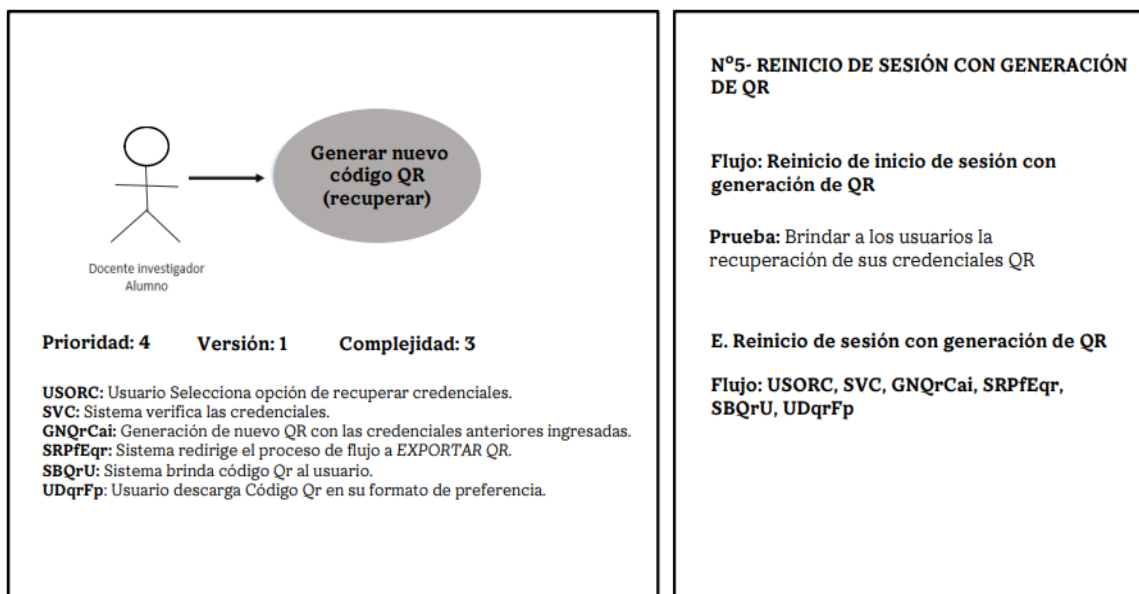
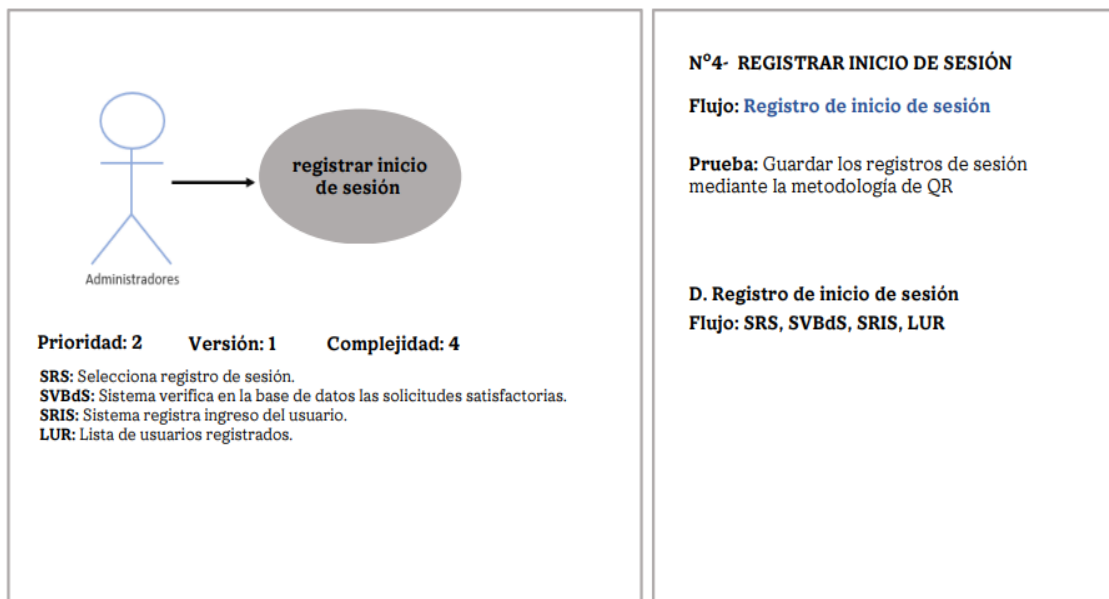
### Nº2- ESCANEAR CÓDIGO QR

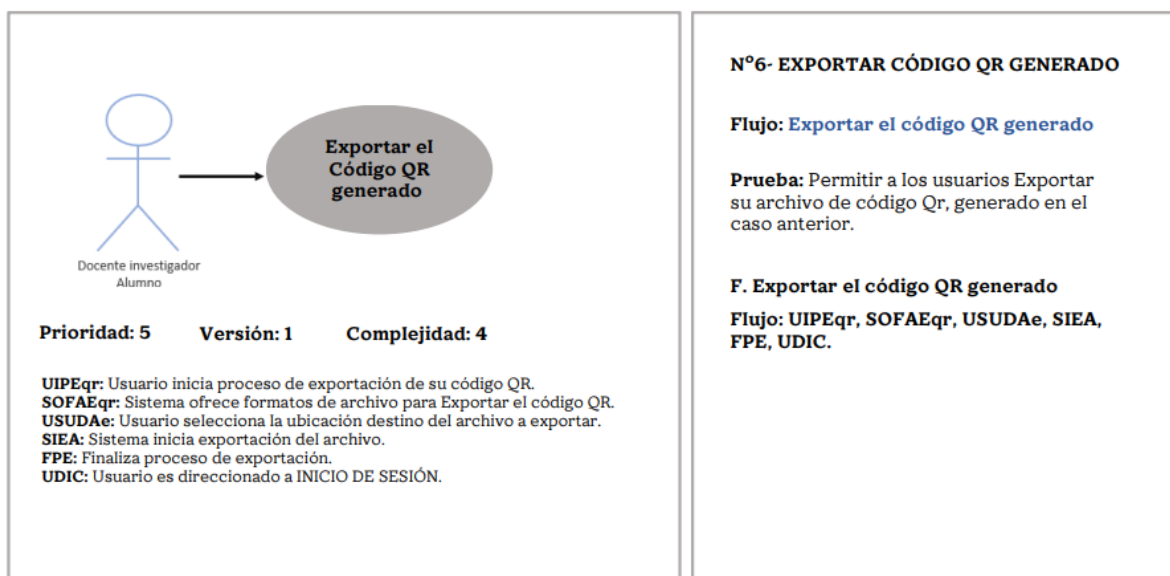
**Flujo:** Escaneo de código QR

**Prueba:** Escanear código QR a través de una cámara..

#### B. Escaneo de código QR

**Flujo:** Isqr, SPC, PA, Lqr, ESDqr, SDCqr, SECqrdbd, ScBd, VCBd, NRES, Cc, UIP





## Descripción detallada de cada caso de uso

### CASO No. 1 iniciar sesión

ID:	CU-1	
Nombre	Iniciar sesión	
Actores	Administrador, Docente investigador, Estudiante.	
Objetivo	Este caso debe permitir ingresar a la cuenta de CREAVI	
Urgencia	5	
Esfuerzo	4	
Pre-condiciones	- Debe haberse registrado satisfactoriamente en el sistema	
Flujo Normal	USUARIO	SISTEMA
	El usuario ingresa su usuario y contraseña	
		El servidor web recibe la solicitud

		El código del lado del servidor se conecta con una base de datos donde se almacenan los datos de los usuarios registrados, incluyendo sus nombres de usuario y contraseñas.
		El código del lado del servidor consulta la base de datos para verificar si el usuario y la contraseña ingresados coinciden con algún registro existente.
		El código del lado del servidor envía una respuesta HTTP al navegador web del usuario con el token de autenticación y una cookie que lo almacena.
		El navegador web recibe la respuesta y guarda la cookie con el token de autenticación.
	Si las credenciales son correctas, el usuario ingresa en la plataforma correctamente	
Flujo alternativo 1	Si la respuesta indica que el inicio de sesión fue fallido, el navegador web muestra un mensaje de error al usuario.	
	Si el sistema no encuentra en la base de datos el usuario y contraseña proporcionados, se pedirá que ingrese nuevamente o se registre si no lo está.	
	Post- condiciones El usuario ingresa satisfactoriamente porque ya tiene cuenta registrada en CREA VI	
Excepciones	El usuario no cuenta con Credenciales existentes (debe registrarse)	

## CASO No. 2 Escanear código QR

ID:	CU-2
Nombre	Escanear código QR
Actores	Docente investigador, Estudiante
Objetivo	Este caso debe permitir la lectura del código QR y posteriormente el ingreso a la plataforma mediante el mismo.

Urgencia	5	
Esfuerzo	5	
Pre-condiciones	<ul style="list-style-type: none"> <li>- Debió registrarse con el método tradicional en la plataforma</li> <li>- Debe contar con Cámara el cliente</li> </ul>	
Flujo Normal	USUARIO	SISTEMA
	Ingresar el método de inicio de sesión mediante código QR	
		Solicita permiso de cámara
	Acepta el permiso y se procede con la lectura del código QR	
		Se envía la solicitud para descifrar el código QR
		El servidor descifra el código QR y se conecta con la base de datos
		Se verifica en la base de datos si las credenciales son correctas
		El navegador recibe la respuesta y envía la solicitud al navegador
	Si las credenciales son correctas, el usuario ingresa a la plataforma	
Flujo alternativo 1	Ingresar el método de inicio de sesión mediante código QR	
		Sistema no detecta cámara (da mensaje de alarma) informando que no se detecta sistema de cámara
Post-condiciones		
Excepciones	El usuario no posee una cámara. La cámara está rota o defectuosa y no puede iniciar sesión mediante código QR (Se redirige al método tradicional)	



## CASO No. 3 Generar código QR

ID:	CU-3	
Nombre	Generar QR	
Actores	Docente investigador, Estudiante	
Objetivo	Este caso debe permitir Generar un código QR en un formato compatible (PDF, IMG, GIF, ETC)	
Urgencia	5	
Esfuerzo	5	
Precondiciones	<ul style="list-style-type: none"> <li>- Debe haberse autenticado de forma correcta en el sistema.</li> <li>- Ya debe tener la cuenta de CREA VI abierta</li> </ul>	
Flujo Normal	USUARIO	SISTEMA
	El usuario se encuentra con su sesión activa y abierta	
	El usuario pulsa “ generar Código QR”	
		El sistema redirige las credenciales del usuario a una extensión mediante una API para generar un código QR que incluya la información de usuario y contraseña con criptografía
		El sistema convierte las credenciales el texto criptográfico
		El sistema procesa las credenciales y redirige la información al CASO No 6
		Retorna al usuario una serie de opciones de descarga para el código QR
	El usuario escoge el formato deseado	
		El sistema prepara y envía el resultado al usuario con el mensaje de “se ha realizado satisfactoriamente”

	El usuario descarga el archivo	
Flujo alternativo 1		El sistema no encuentra credenciales en el sistema por lo que no puede generar un código QR
postcondiciones	El usuario debe tener el rol de docente investigador o estudiante para el correcto funcionamiento del código	
Excepciones	El usuario no es de los roles permitidos para generar el código QR	

## CASO No. 4 Visualizar inicio de sesión

ID:	CU-1	
Nombre	Visualizar inicio de sesión	
Actores	Administrador	
Objetivo	Este caso debe permitir observar y guardar los registros de los inicio de sesión mediante código QR	
Urgencia	2	
Esfuerzo	4	
Pre-condiciones	<ul style="list-style-type: none"> <li>- Debe haberse autenticado como administrador en el sistema</li> </ul>	
Flujo Normal	ADMINISTRADOR	SISTEMA
	Selecciona registros de sesión	
		El sistema verifica dentro de la base de datos todas las solicitudes satisfactorias de inicio de sesión por QR que han sucedido
		El sistema despliega un listado con fecha, hora y usuario de ingreso y si este fue o no correcto
	Observa lista de registrados, usuarios, hora y fecha	

Flujo alternativo 1		
Excepciones	No es usuario Administrador en el sistema	

## CASO No.5 Generar nuevo código QR

ID:	CU-5	
Nombre	Recuperar QR	
Actores	Docente investigador,Estudiante	
Objetivo	Este caso debe permitir recuperar el código QR	
Urgencia	4	
Esfuerzo	3	
Pre-condiciones	<ul style="list-style-type: none"> <li>- Debe haberse autenticado de forma correcta en el sistema con el Rol docente investigador o estudiante</li> </ul>	
Flujo Normal	<b>USUARIO</b>	<b>SISTEMA</b>
	Selecciona la opción de recuperación	
		Verifica las credenciales del sistema
		Genera nuevamente el código QR con las credenciales ingresadas anteriormente
		Redirige el proceso del flujo CU-6
		Brinda el código QR
	Descarga en el formato deseado	

Flujo alternativo	El usuario ingresa directamente a la opción de reiniciar sesión con QR	El sistema verifica y detecta que el usuario NO ha generado el QR previamente dentro de la plataforma creavi
-------------------	--	--

## CASO No. 6 Exportar QR generado

ID:	CU-6	
Nombre	Exportar QR generado	
Actores	Docente investigador, Estudiante	
Objetivo	Este caso debe permitir exportar el código QR generado anteriormente	
Urgencia	5	
Esfuerzo	4	
Pre-condiciones	<ul style="list-style-type: none"> <li>- Debe haberse autenticado de forma correcta en el sistema.</li> <li>- El usuario debió generar el código QR en el caso CU-3</li> </ul>	
Flujo Normal	<b>USUARIO</b>	<b>SISTEMA</b>
	Inicia el proceso de exportación	
		Se despliega la opción del formato de archivo (WEB, JPEG, PNG, BMP, GIF, etc.)
	Selecciona la ubicación de destino	
		Inicia la exportación
	Finaliza el proceso de exportado, el usuario verifica en el Caso N 1	
Excepción	El usuario no da permiso para descargar el archivo.	

## Prioridad de requisitos

A partir del análisis de requerimientos, funcionalidades y el proceso de design thinking, se concreta la siguiente matriz de prioridad de requerimientos.

Para la interpretación se tiene en cuenta la siguiente escala con sus valores.

Eje de Urgencia:

- Obligatoria (5)
- Alta (4)
- Moderada (3)
- Menor (2)
- Baja (1)

Eje de Esfuerzo:

- Muy alto (5)
- Alto (4)
- Medio (3)
- Bajo (2)
- Muy bajo (1)

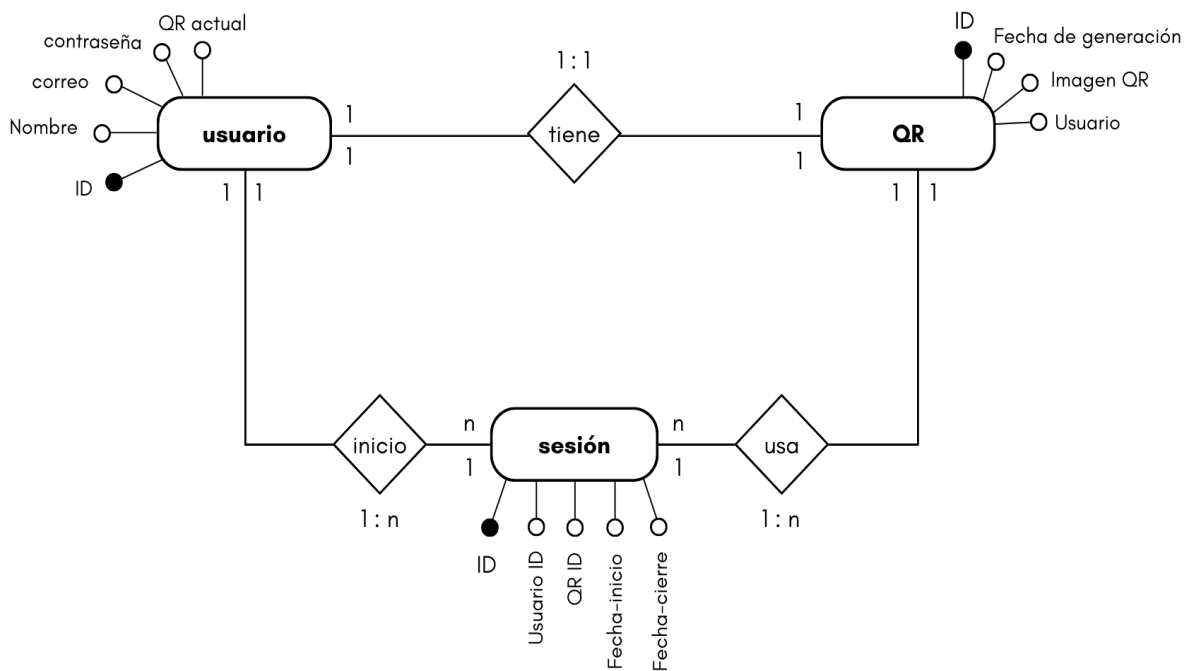
	Urgencia					
esfuerzo		1- Baja	2- Menor	3- Moderada	4- Alta	5- Obligatoria
	5-Muy alto	5	10	15	20	25
						CU-2 CU-3
	4-Alto	4	8	12	16	20
			CU-4			CU-1 CU-6
	3-Medio	3	6	9	12	15
					CU-5	
	2-Bajo	2	4	6	8	10
	1-Muy bajo	1	2	3	4	5

## 5. MODELADO E/R

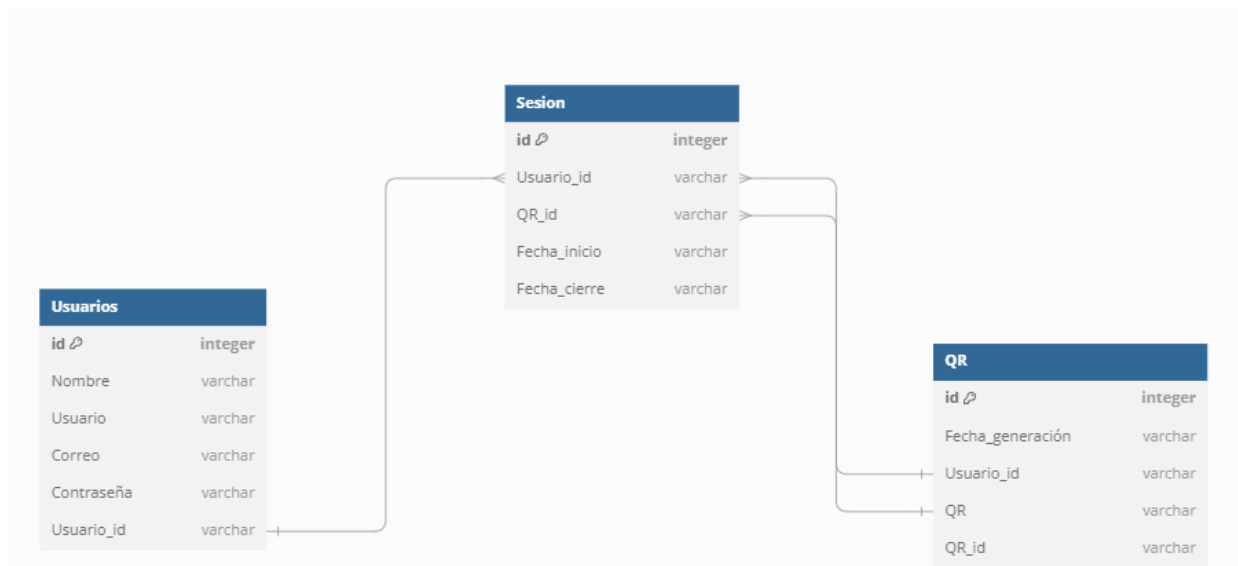
### Diagrama de entidad relación

A partir de los datos pertenecientes a las funcionalidades descritas anteriormente, se establece el siguiente modelado de datos.

<b>CU-1 Iniciar sesión</b>	Usuario	string
	Contraseña	string
<b>CU-2 Escanear código QR</b>	QR en formato físico	
	Cámara	
	Usuario	string
	Contraseña	string
<b>CU-3 Generar código QR</b>	Usuario	string
	Contraseña	string
<b>CU-4 Visualizar inicio de sesión</b>	Usuarios registrados	[“” ]
	Hora	string
	Fecha	string
	Códigos QR	[“” ]
<b>CU-5 Reinicio de sesión con generación de QR</b>	Usuario	string
	Contraseña	string
<b>CU-6 Exportar QR generado</b>	Código QR	string



## Diagrama relacional



## Script de modelo relacional

```

able Usuarios {
id integer [primary key]
Nombre varchar
Usuario varchar
Correo varchar
Contraseña varchar
Usuario_id varchar

```

```
}  
Table QR {  
id integer [primary key]  
Fecha_generación varchar  
Usuario_id varchar  
QR varchar  
QR_id varchar  
}
```

```
Table Sesion {  
id integer [primary key]  
Usuario_id varchar  
QR_id varchar  
Fecha_inicio varchar  
Fecha_cierre varchar  
}
```



## ETAPA 2: PERSISTENCIA DE DATOS CON BACKEND

### 1. INTRODUCCIÓN

#### Propósito de la Etapa

En esta sección describe la arquitectura del sistema de login mediante QR, exponiendo la estructura y organización de los componentes del backend, así como las tecnologías empleadas para su implementación. El propósito es proporcionar una visión clara y precisa del diseño del componente de logueo mediante QR y su interacción con la base de datos. A través de diagramas y descripciones detalladas, se ilustra la relación entre los diferentes componentes y se justifican las decisiones de diseño adoptadas.

#### Alcance de la Etapa

Esta etapa se enfoca en construir la base sólida del sistema de autenticación con QR, desarrollando la infraestructura de backend necesaria para almacenar, gestionar y recuperar los datos del sistema. Se diseñará y desarrollará una base de datos robusta, se establecerán el backend y conexión con la base de datos, y se implementarán las funcionalidades clave para el manejo de usuarios, generación de códigos QR y gestión de sesiones. Además, se garantizará la seguridad de los datos mediante la implementación de encriptados y mecanismos de autenticación y autorización. Todo lo anterior con el fin de sentar las bases para el posterior desarrollo del frontend, asegurando que el sistema sea confiable, escalable y seguro.

#### Definiciones y acrónimos

**Backend:** Parte de una aplicación que gestiona la lógica de negocio y la interacción con la base de datos, quedando oculta al usuario final.

**Frontend:** Parte visible de una aplicación con la que interactúa el usuario directamente.

**API (Interfaz de Programación de Aplicaciones):** Conjunto de reglas y especificaciones que software de diferentes aplicaciones pueden seguir para comunicarse entre sí.

**REST (Representational State Transfer):** Un estilo arquitectónico para el desarrollo de servicios web que se basa en la transferencia de representaciones de estado.

**CRUD:** Acrónimo de Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar), operaciones básicas para gestionar datos en una base de datos.

**SQL (Structured Query Language):** Lenguaje estándar para gestionar bases de datos relacionales.

**NoSQL:** Categoría de sistemas de gestión de bases de datos que no se ajustan al modelo relacional tradicional.

**JWT (JSON Web Token):** Un estándar abierto (RFC 7519) para transmitir información de forma segura entre dos partes como una cadena JSON.

**SGBD (Sistema de Gestión de Bases de Datos):** Software diseñado para organizar, almacenar y recuperar grandes cantidades de datos.

**Encriptado:** Proceso de cifrar información para protegerla de accesos no autorizados. Consiste en transformar datos legibles en una forma ilegible (texto cifrado) mediante un algoritmo y una clave. Solo aquellos que posean la clave correspondiente pueden descifrar los datos y recuperar la información original.

**Criptografía:** Técnica que se encarga de crear métodos para proteger la información, haciéndola ilegible para quienes no tienen autorización de acceso. Utiliza algoritmos matemáticos para cifrar y descifrar mensajes, asegurando la confidencialidad.

**Controller:** Gestiona las solicitudes del usuario y controla la lógica entre los datos y la interfaz.

**Mogno atlas:** Servicio en la nube para gestionar bases de datos MongoDB, que ofrece copias de seguridad, monitorización y alta disponibilidad.

**Servidor:** Dispositivo o software que proporciona recursos, servicios o datos a otros dispositivos, conocidos como clientes, a través de una red.

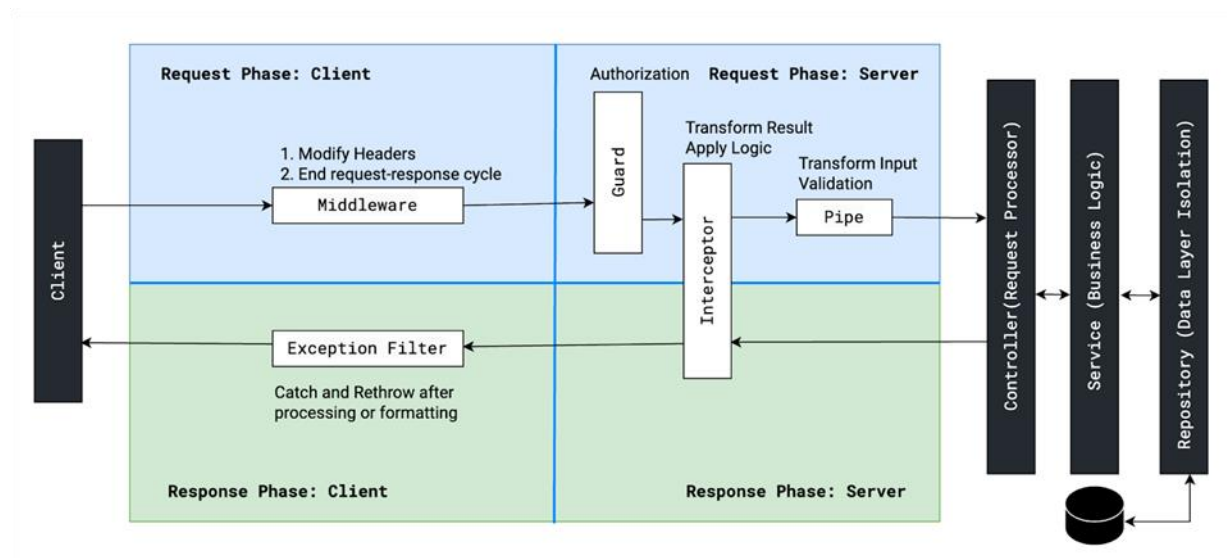
**Service:** Componente que encapsula la lógica de negocio y realiza tareas específicas, como la gestión de datos o la comunicación con otras partes de la aplicación. Se utiliza para estructurar y organizar el código, facilitando la reutilización y el mantenimiento.

## 2. DISEÑO DE LA ARQUITECTURA DE BACKEND

### Descripción de la Arquitectura Propuesta

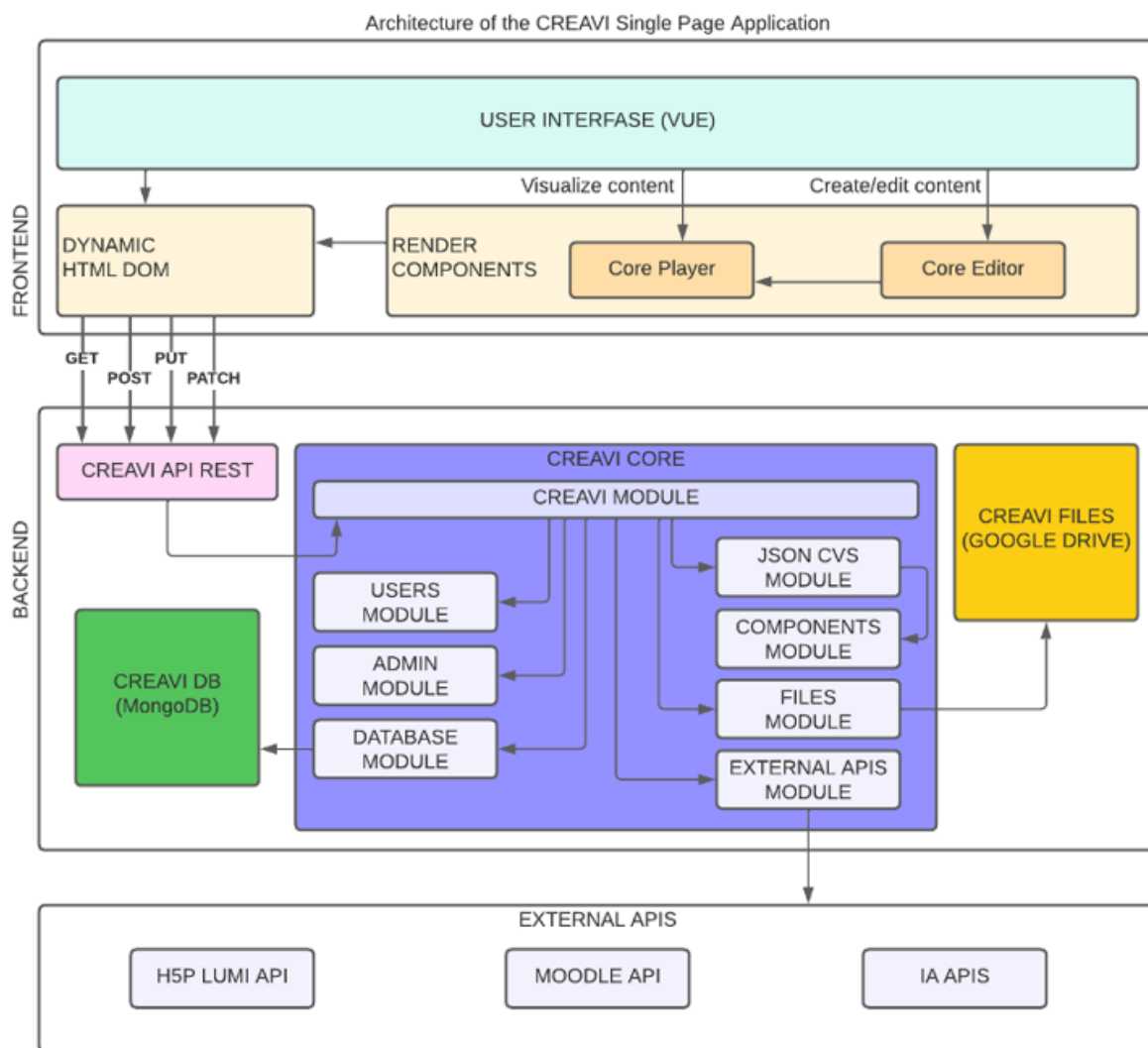
Este proyecto está basado en un sistema de Nest.js que cuenta con múltiples elementos que incluye: el cliente, el interceptor que proporciona una validación al controlador y existe una interacción entre el controlador, el servidor y el repositorio, al igual que con la base de datos.

### Componentes del Backend



En esta etapa del desarrollo del backend del sistema de login con QR, avanzamos desde la Capa de Servicio (Service Layer), donde se gestiona la lógica de negocio, incluyendo la generación y validación segura de códigos QR, así como la gestión de objetos de transferencia de datos (DTOs) para asegurar la correcta transmisión entre cliente y servidor. Luego, en la Capa de Repositorio (Repository Layer), se maneja la conexión a la base de datos y se ejecutan operaciones CRUD (Create, Read, Update, Delete) encapsulando toda la lógica de acceso a datos. Aquí se definen esquemas que estructuran los datos y se optimizan consultas para asegurar la eficiencia y escalabilidad del sistema. Finalmente, en la Capa de Controlador (Controller Layer), se reciben las solicitudes HTTP desde el cliente, dirigiéndolas a los servicios correspondientes, manejando las autenticaciones y respondiendo de manera adecuada, garantizando así una integración segura y eficiente de las capas del sistema en la plataforma CREAVI.

## Diagramas de Arquitectura



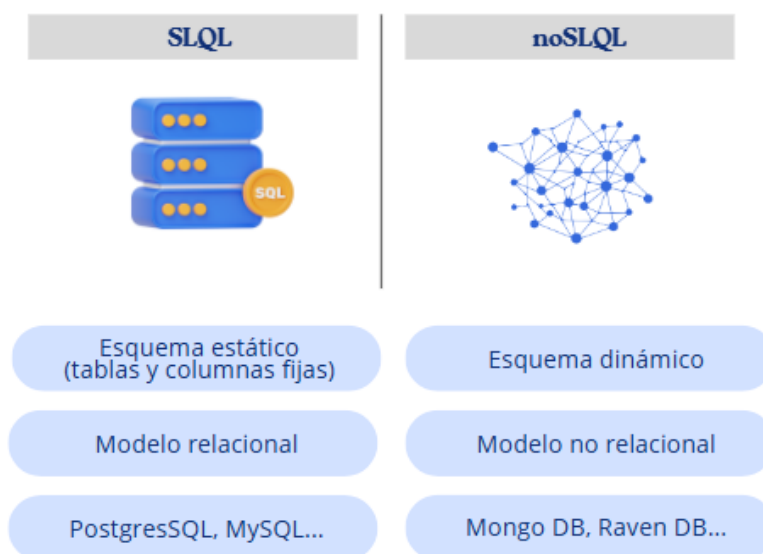
El diagrama anterior muestra la arquitectura de la aplicación de una sola página (SPA) de CREAVI. Esta aplicación cuenta con una interfaz de usuario que incluye componentes para visualizar y editar contenido, un backend que gestiona la lógica a través de distintos módulos y una base de datos MongoDB. En esta configuración del backend, se integrará el módulo de inicio de sesión con QR, el cual facilita el proceso de logueo mediante un código QR, principalmente para usuarios invidentes y personas olvidadizas.

Este módulo de autenticación no solo generará los códigos QR, sino que también verificará su validez para asegurar un acceso seguro a la plataforma. En el frontend, se implementarán componentes para escanear y validar los códigos QR, mientras que en el backend se desarrollará la lógica necesaria para interpretar, gestionar las sesiones de usuario y crear los códigos QR.

Este enfoque mejorará la accesibilidad y usabilidad de la plataforma, permitiendo a todos los usuarios, y especialmente a aquellos con limitaciones visuales, iniciar sesión de forma sencilla, rápida y segura mediante el escaneo del código QR.

## Elección de la Base de Datos

- **Evaluación de Opciones (SQL o NoSQL)**



El tipo de base de datos es una de las decisiones más importantes para el desarrollo del backend del proyecto, estamos hablando de la base de datos que se usará para el software. Entre los tipos de bases de datos más populares están las bases de datos SQL y NoSQL, la elección de una u otra, se basa en las necesidades que presente el software del proyecto.

- **Justificación de la Elección**

El componente login con QR optó por usar una base de datos no SQL, puesto que este tipo de bases de datos son más flexibles respecto al volumen de datos, tiene mayor escalabilidad y rendimiento, lo cual contribuye al alto rendimiento del software, lo que facilitará una respuesta al usuario con discapacidad visual total o parcial que responda a los requerimientos del mismo software.

- **Diseño de Esquema de Base de Datos**

Esquema de la colección Usuario

```

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { HydratedDocument } from 'mongoose';

export type usuarioDocument = HydratedDocument<Usuario>;

@Schema()
export class Usuario {
  @Prop()
  nombre: string;

  @Prop()
  usuario: string;

  @Prop()
  contraseña: string;

  @Prop()
  correo: string;
}

export const UsuarioSchema = SchemaFactory.createForClass(Usuario);

```

Esquema de la colección Docente

```

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { HydratedDocument } from 'mongoose';

export type docenteDocument = HydratedDocument<Docente>;

@Schema()
export class Docente {
  @Prop()
  nombre: string;

  @Prop()
  usuario: string;

  @Prop()
  contraseña: string;

  @Prop()
  correo: string;
}

export const DocenteSchema = SchemaFactory.createForClass(Docente);

```

## Implementación del Backend

- **Elección del Lenguaje de Programación**

Para el desarrollo del componente Login con QR, se optó por utilizar dos lenguajes de programación: Python y JavaScript. Esta elección de lenguajes responde a razones técnicas y funcionales que facilitan la implementación y optimización del desarrollo del componente.

**Python:** Elegido principalmente por su facilidad para integrarse con bases de datos, en este caso, MongoDB. Python al ser un lenguaje sencillo y versátil por sus características como bibliotecas y documentación actualizada, sintaxis clara y códigos concisos, permite una conexión eficiente y sencilla con MongoDB, facilitando la gestión y almacenamiento de los datos del usuario en el sistema.

**JavaScript:** Por otro lado, la elección de JavaScript se debe a su integración fluida con python y tecnologías web modernas, lo cual es fundamental para gestionar la interacción del usuario con la interfaz de la aplicación, como la generación y lectura del código QR. Además, su capacidad para trabajar en conjunto con frameworks y librerías. La combinación de JavaScript y Python se complementan muy eficientemente en la lógica.

## **Creación de la Lógica de Negocio**

- **Introducción:**

La lógica de negocio en este proyecto se implementa tanto en el backend como en el frontend. En el backend, se encarga de procesar las reglas definidas, manejar la validación de datos, gestionar errores y ejecutar procesos relacionados con la autenticación y la verificación de usuarios. Por otro lado, el frontend administra la interacción del usuario con la interfaz gráfica, gestionando la comunicación eficiente con el backend y garantizando que las operaciones se realicen según las reglas establecidas.

Este sistema tiene como propósito principal permitir a los usuarios autenticarse mediante códigos QR cifrados. Para lograrlo, se integraron funcionalidades que permiten generar, decodificar y verificar códigos QR, asegurando que los usuarios puedan iniciar sesión de forma segura. Asimismo, el manejo de errores y las validaciones son fundamentales para garantizar la integridad del sistema y la experiencia del usuario.

## **2. Lógica del Backend**

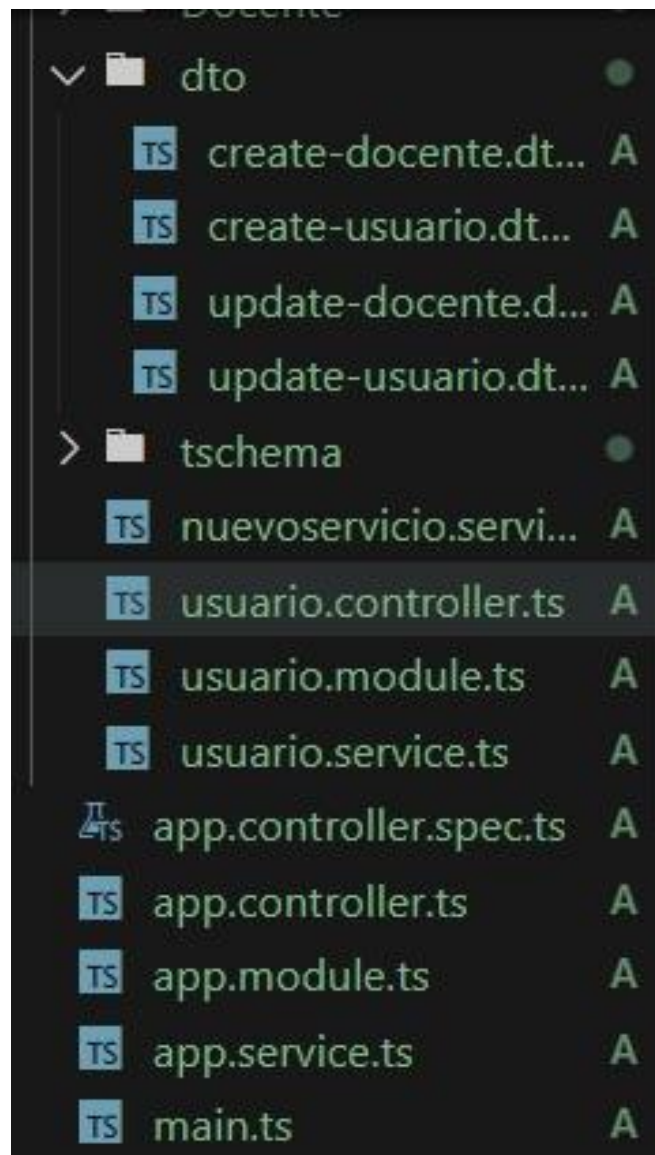
### **2.1 Servicios**

Ubicación de los archivos:

Los servicios relacionados con la lógica de negocio del backend se encuentran en la carpeta `src/services`. En esta carpeta se implementan las reglas principales, como la autenticación, la generación de códigos QR y la validación de datos.

Los servicios en el backend se encargan de:

- Validar datos de entrada: Revisan si los datos enviados por los usuarios cumplen con los requisitos (formato correcto, valores únicos, etc.).
- Gestionar la autenticación: Permiten que los usuarios inicien sesión mediante credenciales tradicionales o códigos QR.
- Procesar transacciones: Controlan la lógica relacionada con operaciones CRUD, como la creación de usuarios y la verificación de credenciales.
- Generar y descifrar códigos QR cifrados: Aseguran que los datos almacenados y procesados estén protegidos y puedan ser validados de manera eficiente.





## Fragmentos de código relevantes:

```
src > my_app > app.py > login
1 from flask import Flask, render_template, request, redirect, url_for, flash, send_file, jsonify
2 from pymongo import MongoClient
3 import qrcode
4 import io
5 from reportlab.lib.pagesizes import letter
6 from reportlab.pdfgen import canvas
7 import base64
8 import tempfile
9 from PIL import Image
10 import fitz # PyMuPDF
11 from pyzbar.pyzbar import decode
12 from cryptography.fernet import Fernet
13
14 app = Flask(__name__)
15 app.secret_key = 'supersecretkey' # Necesario para usar mensajes flash
16
17 # Configurar la conexión a MongoDB
18 mongo_uri = "mongodb://Steven:Steven123@cluster0.lqgtx8.mongodb.net/mydatabase?retrywrites=true&majority=1"
19 client = MongoClient(mongo_uri)
20 db = client['Exx']
21 users = db['usuarios']
22 teachers = db['docentes']
23
24 # Generar una clave de cifrado (deberías guardar esta clave de manera segura)
25 key = Fernet.generate_key()
26 cipher_suite = Fernet(key)
27
28 @app.route('/')
29 def login():
30     return render_template('login.html')
31
32 @app.route('/authenticate', methods=['POST'])
33 def authenticate():
34     usuario = request.form['usuario']
35     contraseña = request.form['contraseña']
36
37     # Buscar el usuario en la base de datos
```

```
33 def authenticate():
34     # Buscar el usuario en la base de datos
35     user = users.find_one({'usuario': usuario, 'contraseña': contraseña})
36     if not user:
37         user = teachers.find_one({'usuario': usuario, 'contraseña': contraseña})
38
39     if user:
40         return "Login successfull"
41     else:
42         return "Invalid usuario or contraseña."
43
44 @app.route('/authenticate_qr', methods=['POST'])
45 def authenticate_qr():
46     data = request.get_json()
47     encrypted_data = data['encrypted_data']
48
49     # Descifrar los datos del código QR
50     try:
51         decrypted_data = cipher_suite.decrypt(encrypted_data.encode()).decode()
52         usuario, contraseña = decrypted_data.split(',')
53         usuario = usuario.split(':')[1]
54         contraseña = contraseña.split(':')[1]
55     except Exception as e:
56         return jsonify({'success': False, 'message': 'Error decrypting data.'})
57
58     # Buscar el usuario en la base de datos
59     user = users.find_one({'usuario': usuario, 'contraseña': contraseña})
60     if not user:
61         user = teachers.find_one({'usuario': usuario, 'contraseña': contraseña})
62
63     if user:
64         return jsonify({'success': True, 'message': 'Login successfull'})
65     else:
66         return jsonify({'success': False, 'message': 'Invalid usuario or contraseña.'})
67
68 @app.route('/register', methods=['GET', 'POST'])
```

En este fragmento de código se muestra la lógica utilizada para autenticar a un usuario. La función `authenticateUser` recibe como parámetros el nombre de usuario y la contraseña ingresados. Primero, consulta la base de datos para verificar si existe un usuario con el nombre proporcionado; si no lo encuentra, lanza un error indicando que el usuario no fue encontrado. Luego, compara la contraseña ingresada con la almacenada en la base de datos utilizando `bcrypt`, una librería que permite manejar contraseñas cifradas de forma segura. Si la contraseña no coincide, lanza otro error indicando que es incorrecta. En caso de que ambas verificaciones sean exitosas, genera un token JWT con los datos del usuario, lo cual permitirá su autenticación en solicitudes futuras. Este token tiene un tiempo de expiración definido, en este caso, de una hora. Finalmente, la función devuelve el token generado. Este servicio es fundamental para la seguridad del sistema, ya que garantiza que solo los usuarios con credenciales válidas puedan acceder a las funcionalidades protegidas.

## 2.2 esquemas

```
// User.js (ubicado en src/models)

const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({

  username: { type: String, required: true, unique: true },

  email: { type: String, required: true, unique: true },

  password: { type: String, required: true },

  role: { type: String, enum: ['user', 'admin'], default: 'user' },

  qrCode: { type: String, required: false }, // Almacena el código QR generado

}); module.exports = mongoose.model('User', UserSchema);
```

Este esquema define la estructura de los datos de los usuarios en la base de datos. El modelo, implementado con Mongoose, especifica varios atributos clave para cada usuario. El campo `username` almacena el nombre de usuario y debe ser único y obligatorio. De manera similar, el campo `email` también es obligatorio y único, asegurando que no existan duplicados en la base de datos. La contraseña se almacena en el campo `password`, que es obligatorio y está cifrada antes de ser guardada. Adicionalmente, el campo `role` permite asignar un rol al usuario, que puede ser `user` o `admin`, siendo `user` el valor predeterminado. El esquema también incluye un campo opcional llamado `qrCode`, que almacena el contenido del código QR generado para cada usuario. Este esquema es crucial para el sistema, ya que define cómo se estructuran y almacenan los datos de los usuarios, asegurando la consistencia y la integridad en las operaciones relacionadas con el manejo de cuentas.

## 3. Lógica del Frontend

En el frontend, la lógica de negocio se centra en gestionar las interacciones del usuario con la interfaz y establecer una comunicación eficiente con el backend. Esta capa se encarga de procesar las entradas del usuario, enviar solicitudes al backend, recibir y manejar las respuestas, y actualizar la interfaz de usuario en consecuencia. En este proyecto, la lógica del frontend incluye la autenticación y verificación mediante códigos QR, así como la gestión de **credenciales** y la interacción fluida entre los componentes visuales y funcionales. Se ha diseñado para ser intuitiva y asegurar que los procesos de inicio de sesión, registro y escaneo de códigos QR se realicen sin contratiempos, proporcionando una experiencia eficiente y segura.

En cuanto a la interacción del usuario, se destacan los elementos del formulario, los botones para iniciar sesión, la opción de recordar al usuario y los enlaces de redes sociales. Todos estos

elementos permiten a los usuarios ingresar sus datos de inicio de sesión, verificar sus credenciales o incluso acceder a la plataforma mediante un código QR.

El formulario utiliza la librería Jinja2 para la renderización del frontend y el backend. El backend recibirá los datos enviados desde el formulario a través de la ruta configurada en el atributo `action="{{ url_for('authenticate') }}"`.

### 3.1 Autenticación y Verificación

En este fragmento de código se describe cómo se gestionan las credenciales y cómo se lleva a cabo la interacción entre el frontend y el backend para verificar la autenticación del usuario:

```
<form action="{{ url_for('authenticate') }}" method="post">

    <div class="input-box">

        <input type="text" placeholder="Usuario" id="usuario" name="usuario" class="form-control"
        required>

    </div>

    <div class="input-box">

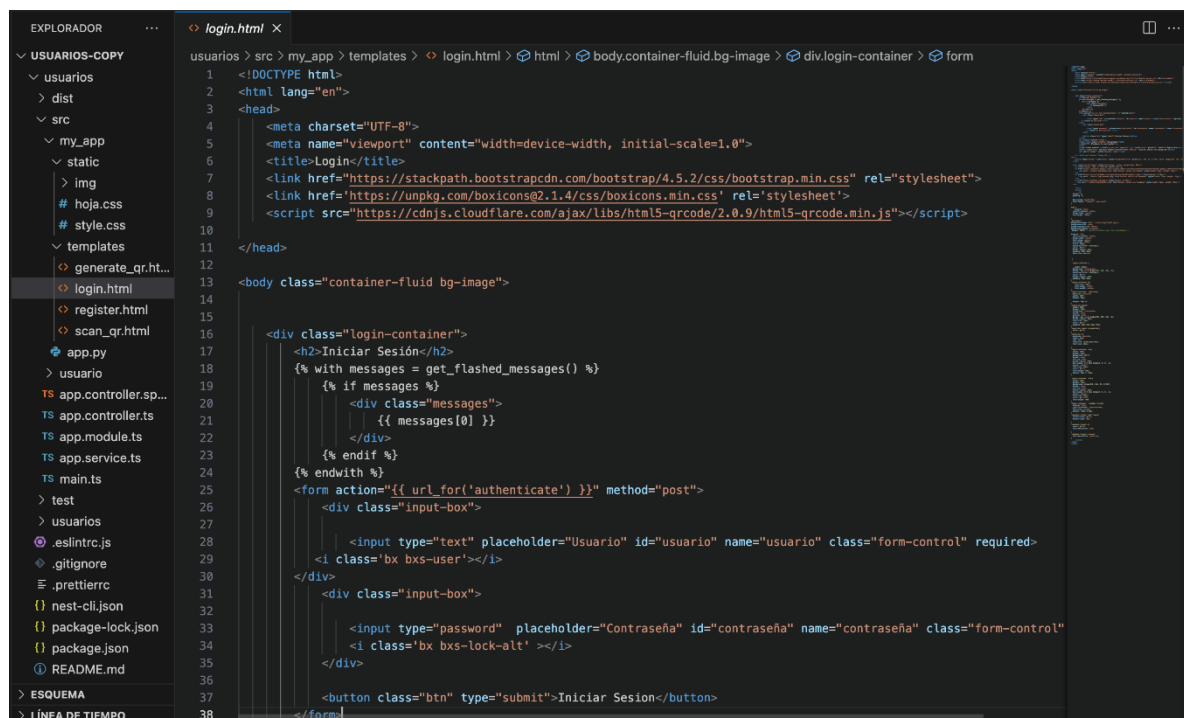
        <input type="password" placeholder="Contraseña" id="contraseña" name="contraseña"
        class="form-control" required>

    </div>

    <button class="btn" type="submit">Iniciar Sesión</button>

</form>
```

A continuación, se muestra la distribución de los códigos CSS y HTML, usados para aplicar el diseño de la interfaz, y vincularlo satisfactoriamente al backend.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Login</title>
7   <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
8   <link href="https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css" rel="stylesheet">
9   <script src="https://cdnjs.cloudflare.com/ajax/libs/html5-qrcode/2.0.9/html5-qrcode.min.js"></script>
10 </head>
11 <body class="container-fluid bg-image">
12
13   <div class="login-container">
14     <h2>Iniciar Sesión</h2>
15     <% with messages = get_flashed_messages() %>
16     <% if messages %>
17       <div class="messages">
18         <{{ messages[0] }}>
19       </div>
20     <% endif %>
21   <% endwith %>
22   <form action="{{ url_for('authenticate') }}" method="post">
23     <div class="input-box">
24       <input type="text" placeholder="Usuario" id="usuario" name="usuario" class="form-control" required>
25       <i class="bx bxs-user"></i>
26     </div>
27     <div class="input-box">
28       <input type="password" placeholder="Contraseña" id="contraseña" name="contraseña" class="form-control" required>
29       <i class="bx bxs-lock-alt"></i>
30     </div>
31     <button class="btn" type="submit">Iniciar Sesión</button>
32   </form>
```

## 4. Conclusión

En este proyecto, la lógica de negocio se centra en la correcta autenticación y gestión de la sesión de usuario, garantizando que el sistema responda adecuadamente a las interacciones del usuario con la interfaz de inicio de sesión. La implementación de estas reglas de negocio se realiza de manera eficaz gracias a la integración fluida entre el frontend y el backend.

El **frontend**, desarrollado en HTML y estilizado con CSS, permite a los usuarios interactuar con la aplicación de manera intuitiva. Los formularios y campos de entrada están diseñados para capturar las credenciales del usuario, y la visualización de mensajes (como errores de autenticación) se maneja dinámicamente mediante Jinja2, lo que asegura que la experiencia del usuario sea coherente y amigable.

El **backend**, utilizando Flask, gestiona la validación de las credenciales proporcionadas, autentica al usuario y maneja las respuestas a través de rutas específicas. El backend también maneja la seguridad del sistema, como el almacenamiento de tokens de sesión, asegurando que solo los usuarios autenticados puedan acceder a contenido protegido.

La comunicación entre ambos lados se realiza mediante solicitudes HTTP que permiten que el frontend envíe datos al backend (como las credenciales del usuario) y reciba respuestas (como

un mensaje de éxito o error), manteniendo la lógica de negocio alineada con las reglas definidas para la gestión de usuarios y sesiones.

Este enfoque modular y bien estructurado entre frontend y backend asegura que las reglas de negocio sean implementadas de manera efectiva, proporcionando una experiencia de usuario fluida y segura.

## Conexión a la Base de Datos

- **Configuración de la Conexión**

Para conectar Login Qr a la base de datos en MongoDB Atlas se deben cumplir 4 pasos principales que incluyen configurar el URI de conexión, instanciar el cliente de MongoDB, seleccionar la base de datos y definir las colecciones necesarias para la aplicación. Esta configuración asegura que el sistema pueda gestionar de manera eficiente la información requerida para su funcionamiento.

```
# Configurar la conexión a MongoDB
mongo_uri = "mongodb://[usuario]:[password]@cluster0.lxqrtx8.mongodb.net/mydatabase?retryWrites=true&w=majority"
client = MongoClient(mongo_uri)
db = client['Exx']
users = db['usuarios']
teachers = db['docentes']
```

De forma más detallada, login Qr realiza la conexión a la base de datos MongoDB utilizando un cliente que permite establecer comunicación con el servidor mediante un URI de conexión. Este URI contiene la información necesaria para la autenticación y ubicación del cluster de MongoDB, incluyendo el usuario, la contraseña, la dirección del cluster y el nombre de la base de datos. Una vez definido el URI, se utiliza la clase MongoClient para crear una instancia de cliente conectada al servidor especificado.

Posteriormente, se selecciona la base de datos deseada, 'Exx'. Dentro de esta base de datos se definen las colecciones relevantes para el sistema, como 'usuarios' y 'docentes', que representan las entidades principales manejadas por la aplicación. Estas colecciones permiten interactuar con los datos almacenados mediante operaciones CRUD (crear, leer, actualizar y eliminar).

- **Desarrollo de Operaciones CRUD**

### *Usuarios: Docente*

```
@Controller('docente')
export class DocenteController {
  constructor(private readonly DocenteService: DocenteService) {}

  @Post()
  create(@Body() createDocenteDto: CreateDocenteDto) {
    return this.DocenteService.create(createDocenteDto);
  }

  @Get()
  findAll() {
    return this.DocenteService.findAll();
  }

  @Get('/:id')
  findOne(@Param('id') id: string) {
    return this.DocenteService.findOne(id);
  }

  @Patch('/:id')
  update(@Param('id') id: string, @Body() updatedocenteDto: UpdateDocenteDto) {
    return this.DocenteService.update(id, updatedocenteDto);
  }

  @Delete('all')
  async deleteAll() {
    return await this.DocenteService.deleteAll();
  }

  @Delete('/:id')
  @HttpCode(200)
  async remove(@Param('id') id: string) {
    const result = await this.DocenteService.remove(id);
    return {
      message: 'docente borrado correctamente',
      count: result.deletedCount,
    };
  }
}
```

El controlador de docentes gestiona las acciones principales relacionadas con los datos de los docentes. Incluye métodos para crear nuevos docentes, listar todos los registros, buscar uno en particular mediante su ID, modificar información existente y eliminar registros de forma individual o masiva, facilitando la administración eficiente de la información.

### *Usuarios: usuario*

```
@Controller('usuario')
export class UsuarioController {
  constructor(private readonly usuarioService: UsuarioService) {}

  @Post()
  create(@Body() createUsuarioDto: CreateUsuarioDto) {
    return this.usuarioService.create(createUsuarioDto);
  }

  @Get()
  findAll() {
    return this.usuarioService.findAll();
  }

  @Get('/:id')
  findOne(@Param('id') id: string) {
    return this.usuarioService.findOne(id);
  }

  @Patch('/:id')
  update(@Param('id') id: string, @Body() updateUsuarioDto: UpdateUsuarioDto) {
    return this.usuarioService.update(id, updateUsuarioDto);
  }

  @Delete('all')
  async deleteAll() {
    return await this.usuarioService.deleteAll();
  }

  @Delete('/:id')
  @HttpCode(200)
  async remove(@Param('id') id: string) {
    const result = await this.usuarioService.remove(id);
    return {
      message: 'Usuario borrado correctamente',
      count: result.deletedCount,
    };
  }
}
```

El controlador de usuarios implementa las operaciones CRUD necesarias para gestionar usuarios, incluyendo la creación de nuevos usuarios, la obtención de todos los registros o de un usuario específico mediante su ID, la actualización de datos y la eliminación, ya sea de un usuario en particular o de todos los registros, garantizando una gestión eficiente y organizada de los datos

- **Manejo de Transacciones**

### **Descripción de las transacciones entre el Frontend y el Backend**

En un proyecto, una transacción entre el frontend y el backend ocurre cuando el frontend realiza una solicitud al backend para interactuar con los datos almacenados en la base de datos. Estas transacciones suelen ser de tipo request-response, donde el frontend envía datos (o solicita información) y el backend procesa la solicitud, realiza las operaciones necesarias (como leer, escribir o actualizar datos) y devuelve una respuesta al frontend.

### **Descripción del flujo de la transacción**

1. **Frontend a Backend:** El frontend inicia la transacción enviando los datos a través de una solicitud HTTP POST.
2. **Backend a Base de Datos:**
  - A. El backend valida y procesa los datos.
  - B. Realiza una operación atómica (en este caso, guardar un usuario en la base de datos).
  - C. Devuelve el resultado de la operación al frontend.
3. **Respuesta al Frontend:**
  - A. Si la operación fue exitosa, el backend envía un mensaje confirmando la creación del usuario.
  - B. Si ocurrió un error (p. ej., datos inválidos), se informa al frontend con un mensaje de error.

Supongamos que queremos registrar un usuario y asignarle un rol al mismo tiempo (dos operaciones interdependientes). Este flujo podría implementarse así:

### **Backend (transacción en servicio):**

## **CODIGO:**

```
async createUsuarioWithRole(createUsuarioDto: CreateUsuarioDto, role: string):
Promise<any> {

    const session = await this.usuarioModel.db.startSession();

    session.startTransaction();

    try {

        const user = new this.usuarioModel(createUsuarioDto);

        const savedUser = await user.save({ session });

        const roleDoc = { userId: savedUser._id, role };

        await this.roleModel.create([roleDoc], { session });

        await session.commitTransaction();

        return { message: "Usuario y rol creados exitosamente", user: savedUser };

    } catch (error) {

        await session.abortTransaction();

        throw new Error("Transacción fallida: " + error.message);

    } finally {

        session.endSession();

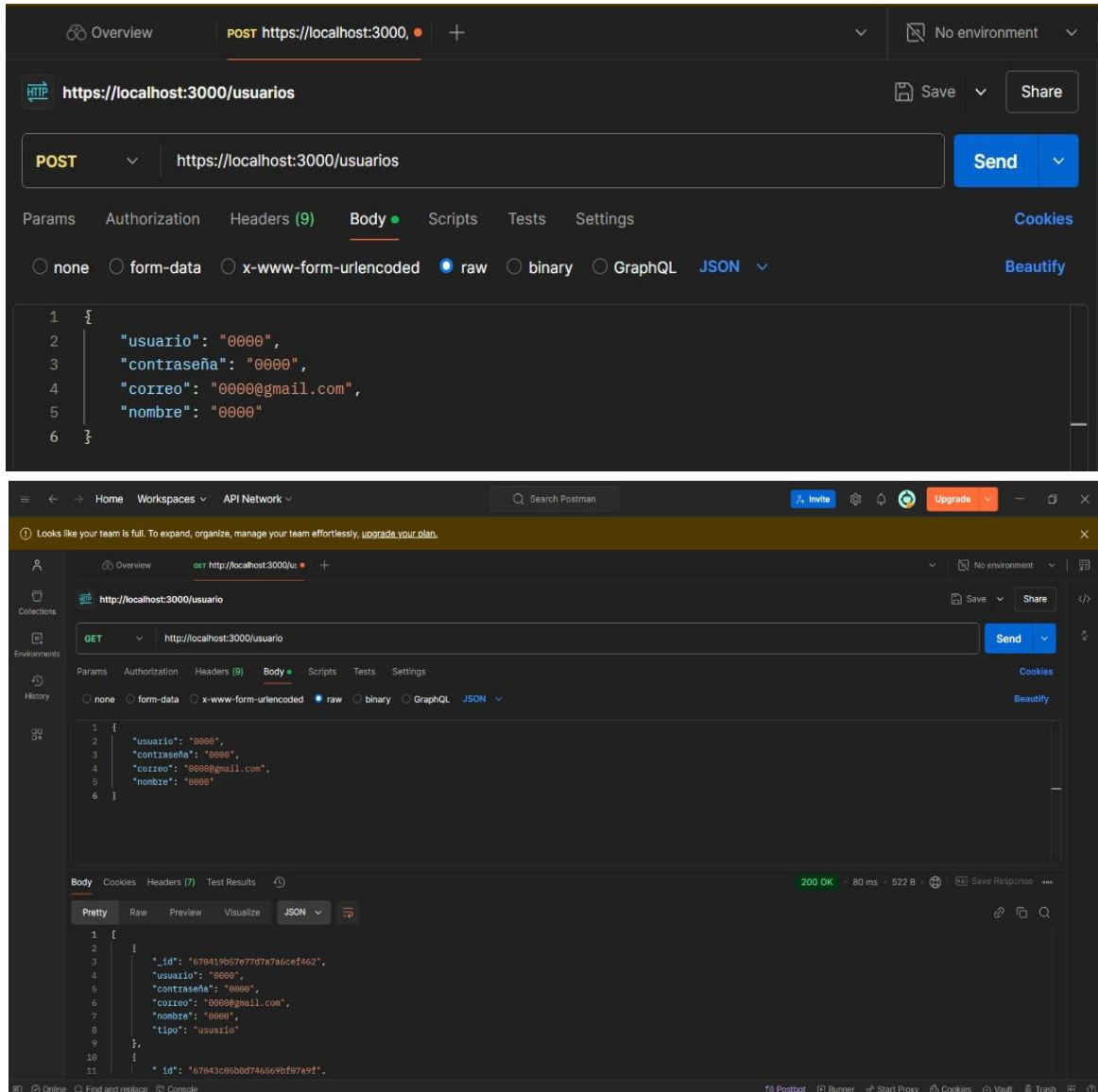
    }

}
```



- **Pruebas del Backend**

Estas pruebas fueron realizadas con postman para constatar y ajustar las posibles fallas y complicaciones en el componente.



## ETAPA 3: CONSUMO DE DATOS Y DESARROLLO FRONTEND

### 1.INTRODUCCIÓN

#### Propósito de la Etapa

El objetivo principal de esta etapa es crear una interfaz de usuario intuitiva y funcional que permita a los usuarios interactuar de manera fluida con el sistema de autenticación mediante códigos QR. Se busca desarrollar una experiencia de usuario óptima, donde los usuarios puedan escanear sus códigos QR de forma sencilla y segura, accediendo así a las funcionalidades del sistema. Además, se establecerá una conexión sólida entre la interfaz gráfica y la lógica del backend, garantizando la correcta visualización y manipulación de los datos.

#### Alcance de la Etapa

En esta etapa, nos concentramos en hacer realidad el diseño de la interfaz de usuario. La capa de presentación de la aplicación se desarrollará para proporcionar a los usuarios una interfaz fácil de entender y visualmente atractiva que les permita interactuar fácilmente con el sistema de autenticación mediante códigos QR. El diseño y desarrollo de pantallas, la lógica de interacción con el usuario, la integración con el backend para obtener y enviar datos, la realización de pruebas exhaustivas para garantizar la calidad y el rendimiento de la aplicación hacen parte de lo que se pretende lograr en esta etapa.

#### Definiciones y acrónimos

**Interfaz de Usuario (UI):** Es la interfaz visual, a través de la cual los usuarios interactúan con una aplicación. Incluye elementos como botones, formularios, menús y cualquier otro elemento con el que los usuarios puedan interactuar.

**Experiencia de Usuario (UX):** Se trata de la experiencia general que tiene un usuario al interactuar con una aplicación o sitio web. Incluye aspectos como la facilidad de uso, la accesibilidad, la eficiencia y la satisfacción del usuario.

**Maquetación:** Proceso de organizar y estructurar los elementos de una página web (como texto, imágenes y botones) para garantizar una presentación visual atractiva, coherente y funcional en diferentes dispositivos y tamaños de pantalla.

**Flujo de Navegación:** Conjunto de rutas o secuencia de pantallas que un usuario sigue al interactuar con una aplicación o sitio web, facilitando la realización de tareas y el acceso a diferentes funcionalidades.

**CSS (Cascading Style Sheets):** Es un lenguaje de hojas de estilo que se utiliza para describir la presentación visual de un documento HTML. Se utiliza para definir el diseño, el formato, los colores y otras características visuales de una página web.

**UI/UX Design (Diseño de Interfaz y Experiencia de Usuario):** Proceso de diseñar la interfaz visual y la experiencia de usuario de una aplicación o sitio web. Incluye la creación de wireframes, maquetas y prototipos, así como la iteración y mejora continua del diseño basada en la retroalimentación del usuario.

**API (Interfaz de Programación de Aplicaciones):** Conjunto de reglas y protocolos que permite a diferentes aplicaciones o sistemas comunicarse entre sí. En el contexto del desarrollo frontend, se refiere a las APIs utilizadas para obtener y enviar datos desde y hacia el backend de una aplicación.

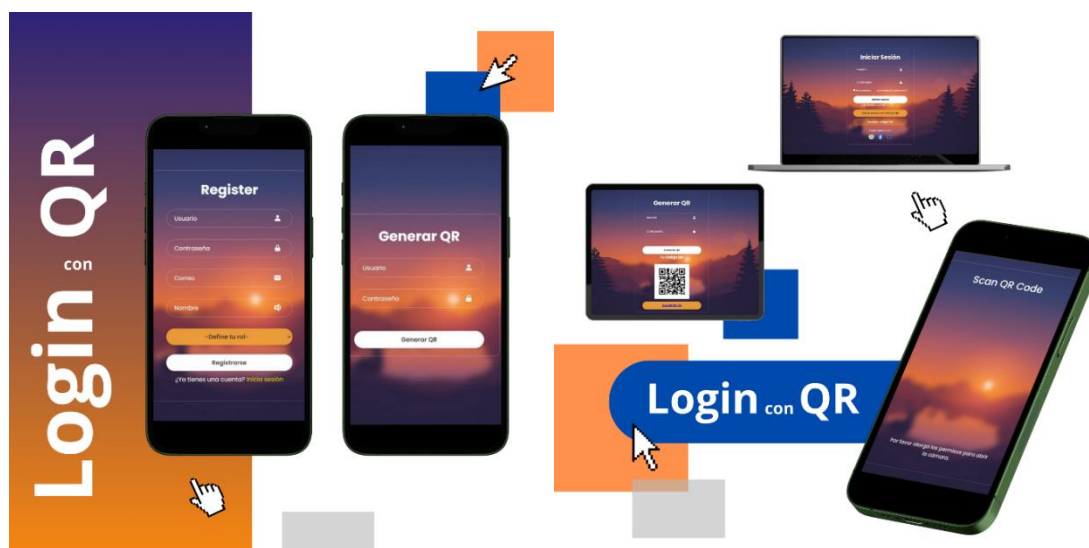
**HTML (Hypertext Markup Language):** Lenguaje estándar utilizado para crear y diseñar páginas web y aplicaciones web. Se utiliza para estructurar el contenido de una página web mediante el uso de etiquetas y elementos.

**Responsive:** Se refiere a la capacidad de un diseño web o aplicación de adaptarse automáticamente a diferentes tamaños de pantalla y dispositivos (desde computadoras de escritorio hasta teléfonos móviles). Un diseño responsive garantiza una experiencia de usuario óptima en cualquier dispositivo.

## 2.DISEÑO DE AMBIENTE

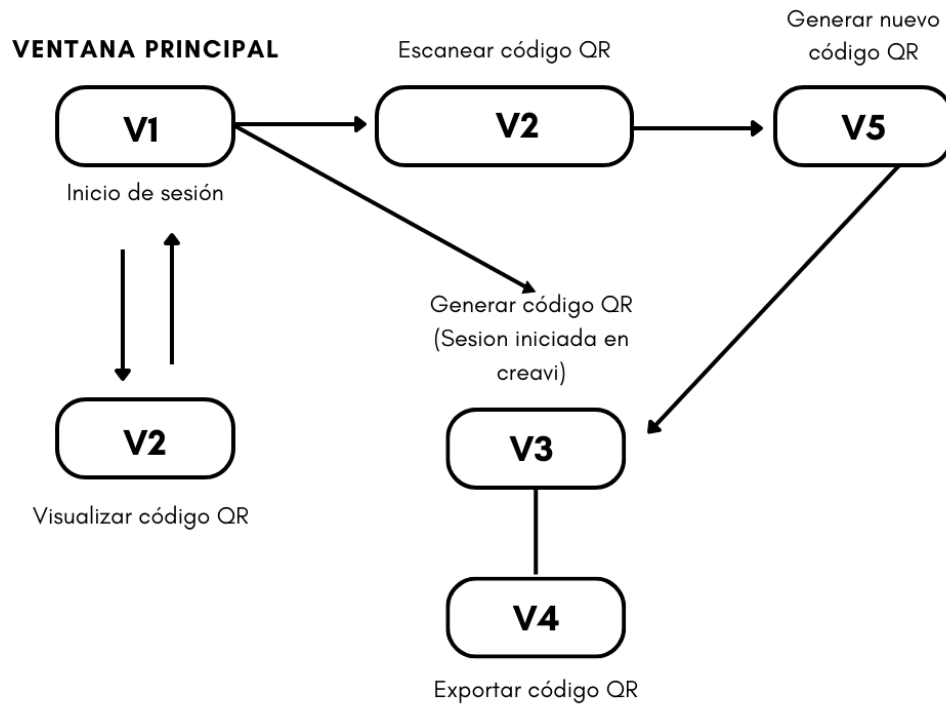
### Diseño de la Interfaz Gráfica de Usuario (GUI)

#### Mockups:

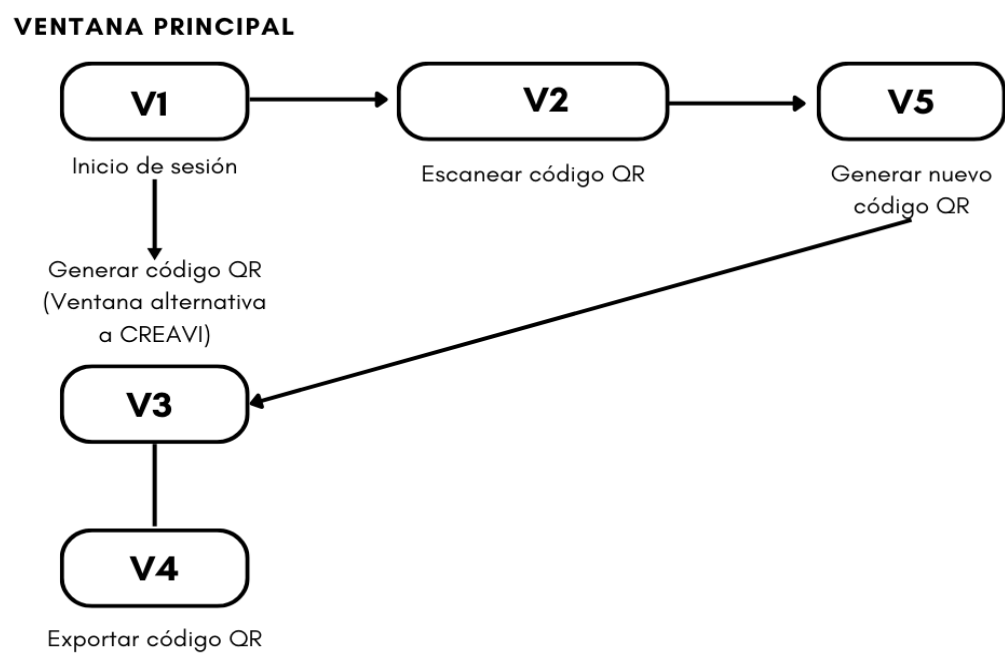


## Mapa de navegación

1-El siguiente mapa de navegación corresponde al usuario como administrador.

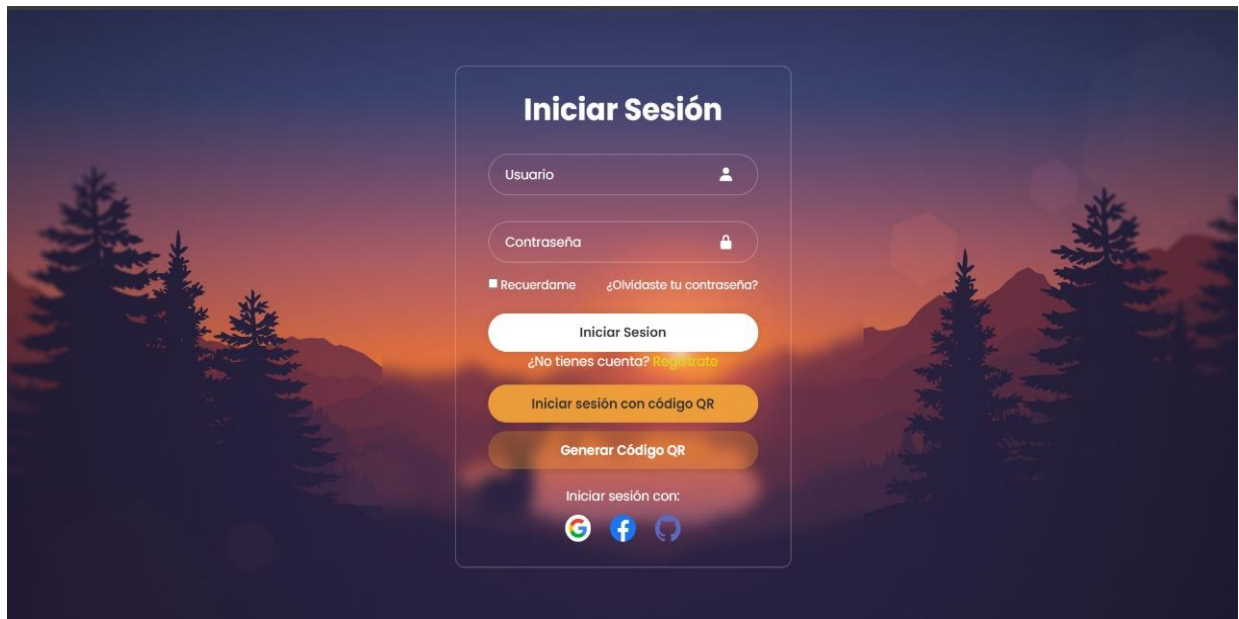


2-El siguiente mapa de navegación corresponde al usuario como estudiante.



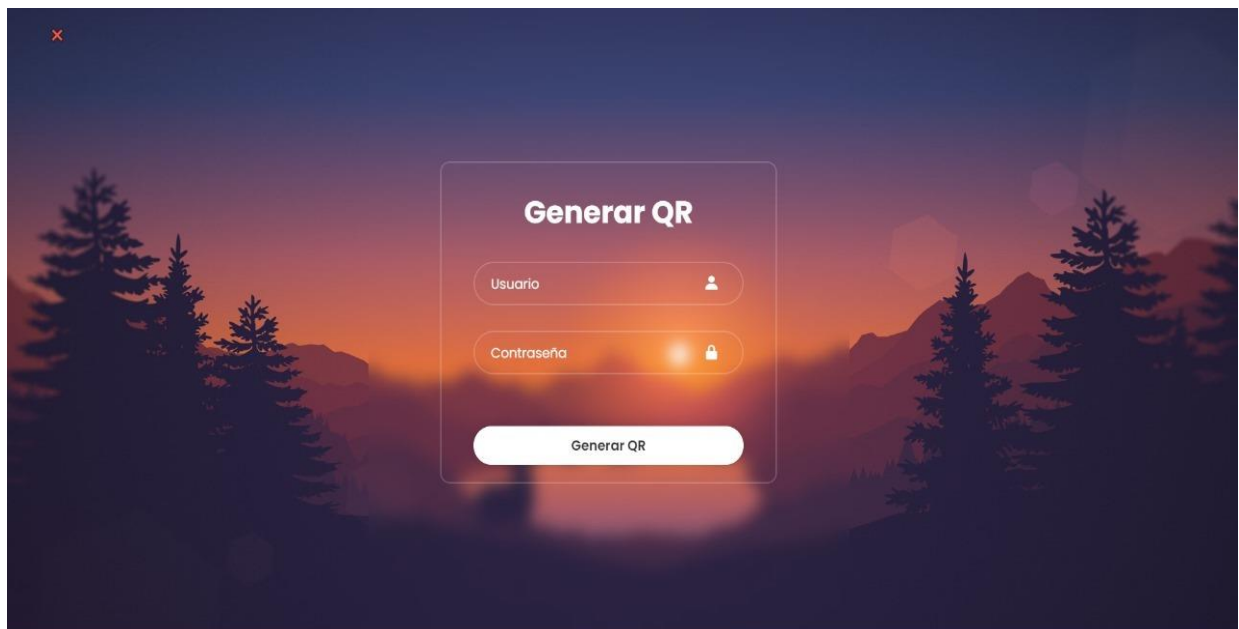
## Diseño de las ventanas

### Ventana I



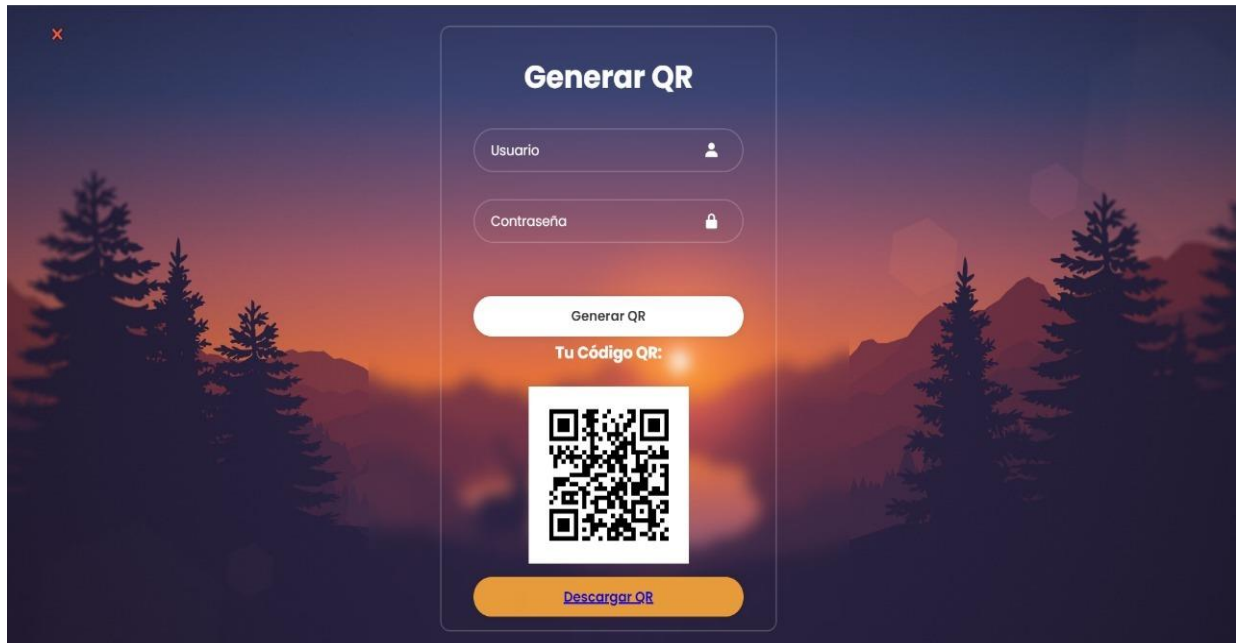
Pantalla principal de inicio de sesión donde el usuario puede ingresar su nombre de usuario y contraseña para acceder a la plataforma.

### Ventana II



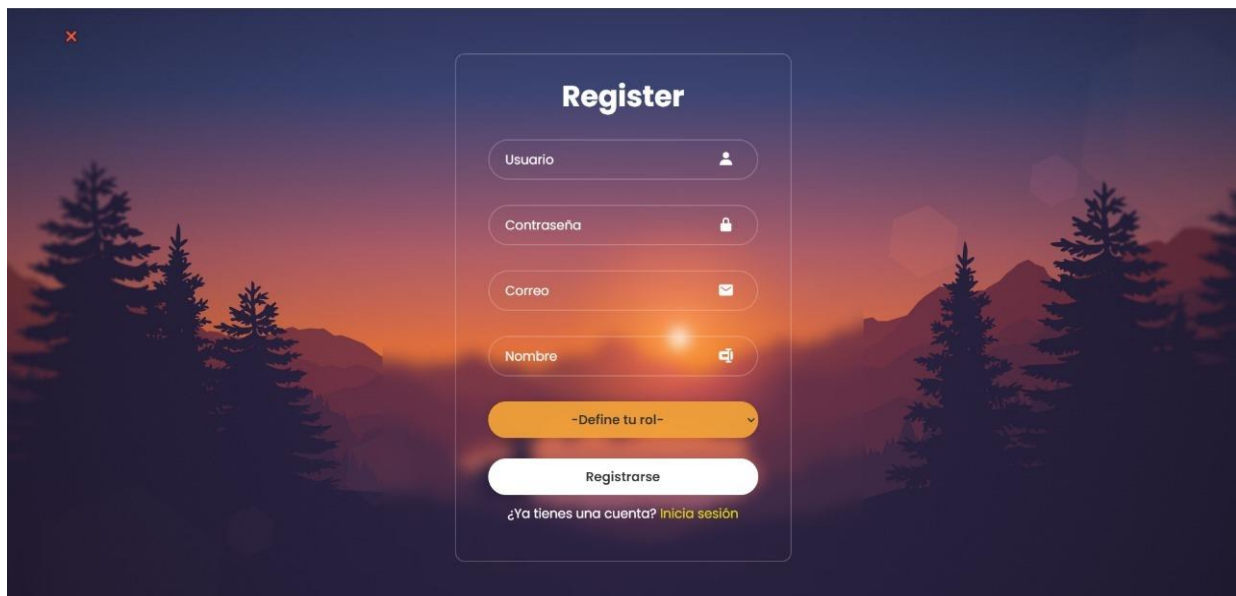
Esta ventana es similar a la ventana III, pero más simplificada en diseño. Permite generar un código QR ingresando el nombre de usuario y la contraseña, mostrando únicamente el botón de generación, sin opciones adicionales como la descarga.

### Ventana III



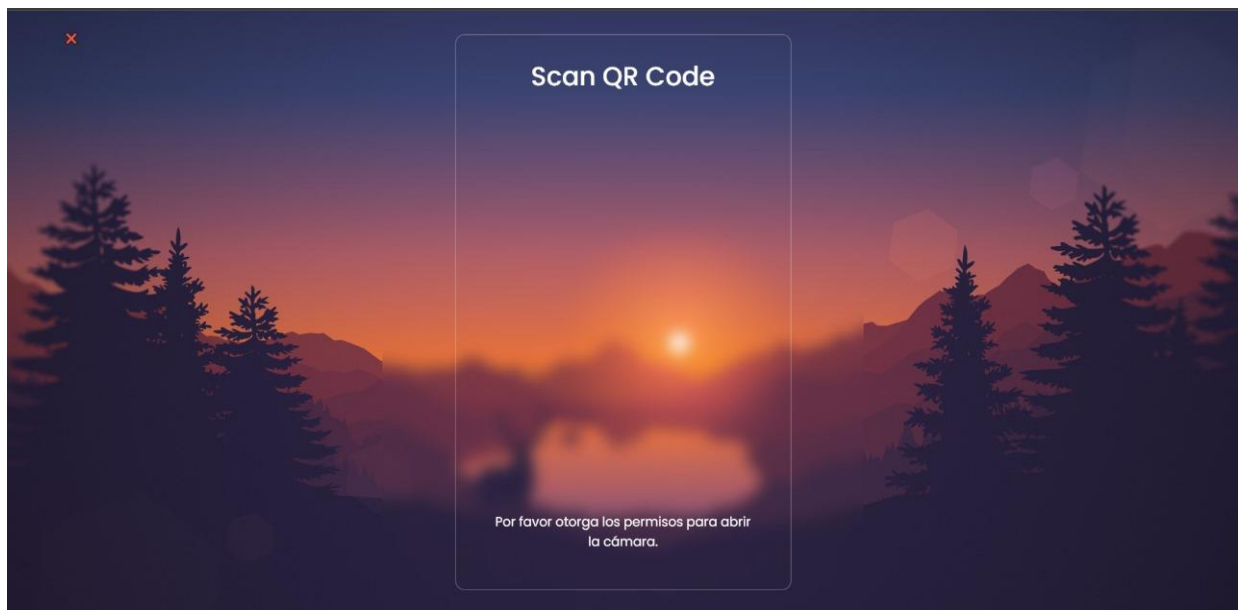
Esta ventana facilita la generación de un código QR personal, requiriendo que el usuario proporcione su nombre de usuario y contraseña; además, permite descargar el QR generado.

### Ventana IV



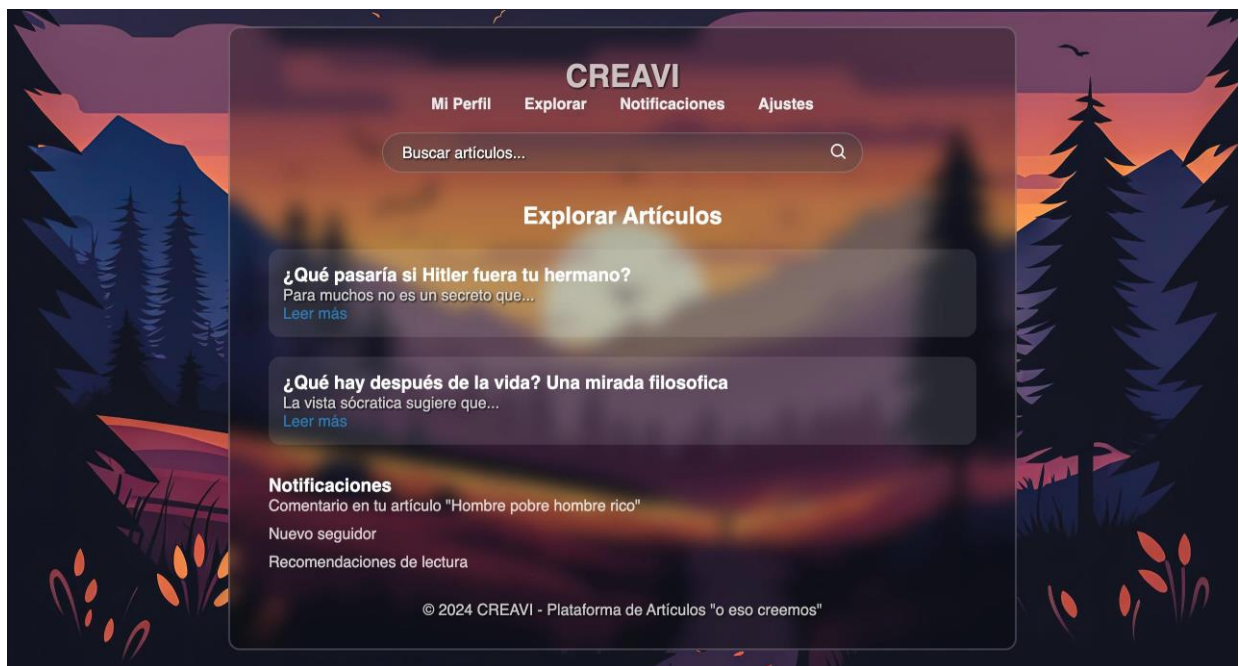
Esta ventana presenta un formulario de registro para nuevos usuarios, donde se solicitan datos como usuario, contraseña, correo electrónico, nombre completo y el rol que desean asignarse dentro de la plataforma; también incluye un botón para registrarse y un enlace para regresar al inicio de sesión en caso de ya tener una cuenta.

## Ventana V



Esta ventana permite a los usuarios escanear un código QR para autenticarse y solicita permisos para usar la cámara del dispositivo.


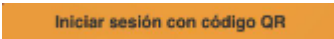




## Ventana VI





## Diseño de guía de metáforas


### Ventana número I:

Guia de metáforas		
Nombre	Imagen	Descripción
<b>Inicio de sesión</b>		Botón que permite acceder a la cuenta CREA VI
<b>Inicio de sesión con código QR</b>		Redirecciona al inicio de sesión con QR
<b>Facebook</b>		Botón que permite entrar por medio de la cuenta de Facebook del usuario
<b>Google</b>		Botón de Google, permite entrar por medio de la cuenta
<b>Github</b>		Botón de Github, permite entrar por medio de la cuenta del usuario
<b>Generar código QR</b>		Permite crear el código QR

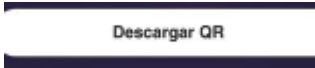
### Ventana número II, escanear QR:

Guia de metáforas		
Nombre	Imagen	Descripción
<b>Generar nuevo código QR</b>		Crear un código QR
<b>Cámara</b>		Botón para activar la cámara y escanear código QR

### Ventana número III, generar código QR

Guia de metáforas		
Nombre	Imagen	Descripción
<b>Generar QR</b>		Generar código QR




<b>Descargar código QR</b>		Botón que permite descargar el código QR generado
----------------------------	---	---

#### Ventana de registro IV

Guia de metáforas		
Nombre	Imagen	Descripción
<b>Define tu rol</b>		Al registrarse se le pide definir el rol, sea administrador, docente investigador o estudiante.
<b>Registrarse</b>		Botón que permite crear una cuenta.

#### Ventana de registro V

Guia de metáforas		
Nombre	Imagen	Descripción
<b>Buscar articulos</b>		Permite buscar articulos a tu disposición

### 3. CONEXIÓN DE FRONTEND Y BACKEND

La conexión entre el backend y el frontend se establece mediante el uso de plantillas HTML y el intercambio de datos a través de solicitudes HTTP (GET y POST).

- **Renderización de Plantillas HTML:**

El backend utiliza la función `render_template` de Flask para servir páginas HTML al navegador del usuario. Las rutas principales incluyen:

1. `/` (pantalla de inicio de sesión): Renderiza la plantilla [login.html](#).
2. `/register`: Renderiza la plantilla [register.html](#) para el registro de nuevos usuarios.
3. `/scan_qr`: Renderiza la plantilla [scan\\_qr.html](#) para la autenticación mediante código QR.
4. `/generate_qr`: Renderiza la plantilla [generate\\_qr.html](#) para generar un código QR.

#### Código:

```

@app.route('/')
def login():
    return render_template('login.html')

@app.route('/authenticate', methods=['POST'])
def authenticate():
    usuario = request.form['usuario']
    contraseña = request.form['contraseña']

    # Buscar el usuario en la base de datos
    user = users.find_one({'usuario': usuario, 'contraseña': contraseña})
    if not user:
        user = teachers.find_one({'usuario': usuario, 'contraseña': contraseña})

    if user:
        return "Login successful!"
    else:
        return "Invalid usuario or contraseña."

@app.route('/authenticate_qr', methods=['POST'])
def authenticate_qr():
    data = request.get_json()
    encrypted_data = data['encrypted_data']

    # Descifrar los datos del código QR
    try:
        decrypted_data = cipher_suite.decrypt(encrypted_data.encode()).decode()
        usuario, contraseña = decrypted_data.split(',')
        usuario = usuario.split(':')[1]
        contraseña = contraseña.split(':')[1]
    except Exception as e:
        return jsonify({'success': False, 'message': 'Error decrypting data.'})

```

- **Intercambio de Datos entre el Frontend y el Backend:**

### Registro de usuario

En /register, los datos del formulario de registro, como usuario, contraseña, correo, nombre y tipo de cuenta, se validan e insertan en la base de datos.

### Autenticación mediante Código QR:

La funcionalidad de autenticación con QR se maneja en la ruta /authenticate\_qr. El frontend envía datos cifrados en formato JSON al backend mediante una solicitud POST, donde se descifran y verifican contra la base de datos.

### Generación de Códigos QR:

En /generate\_qr, el backend recibe datos del usuario mediante un formulario. Estos datos se cifran y se utilizan para generar un código QR mediante la librería qrcode. La imagen del código QR se convierte a base64 y se envía al frontend para su visualización.

### Descarga de Archivos:

El backend permite la descarga de un PDF con el código QR generado a través de la ruta /download\_pdf, utilizando la función send\_file.

### Procesamiento de Archivos Subidos:

En /upload\_qr, el backend procesa archivos (imágenes o PDFs) cargados desde el frontend. Estos archivos se analizan para extraer y descifrar los datos del QR, que luego se verifican contra la base de datos.

```
@app.route('/authenticate', methods=['POST'])
def authenticate():
    usuario = request.form['usuario']
    contraseña = request.form['contraseña']

    # Buscar el usuario en la base de datos
    user = users.find_one({'usuario': usuario, 'contraseña': contraseña})
    if not user:
        user = teachers.find_one({'usuario': usuario, 'contraseña':
contraseña})

    if user:
        return "Login successful!"
    else:
        return "Invalid usuario or contraseña."

@app.route('/authenticate_qr', methods=['POST'])
def authenticate_qr():
    data = request.get_json()
    encrypted_data = data['encrypted_data']

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        usuario = request.form['usuario']
        contraseña = request.form['contraseña']
        correo = request.form['correo']
        nombre = request.form['nombre']
        tipo = request.form['tipo']
```

```

# Verificar si el usuario, correo o nombre ya existen
existing_user = users.find_one({'usuario': usuario})
existing_teacher = teachers.find_one({'usuario': usuario})
existing_email_user = users.find_one({'correo': correo})
existing_email_teacher = teachers.find_one({'correo': correo})
existing_name_user = users.find_one({'nombre': nombre})
existing_name_teacher = teachers.find_one({'nombre': nombre})

if existing_user or existing_teacher:
    flash("Username already exists. Please choose a different one.")
elif existing_email_user or existing_email_teacher:
    flash("Email already exists. Please choose a different one.")
elif existing_name_user or existing_name_teacher:
    flash("Name already exists. Please choose a different one.")
else:
    # Insertar el nuevo usuario en la base de datos
    if tipo == 'usuario':
        users.insert_one({'usuario': usuario, 'contraseña': contraseña,
'correo': correo, 'nombre': nombre, 'tipo': tipo})
    elif tipo == 'docente':
        teachers.insert_one({'usuario': usuario, 'contraseña':
contraseña, 'correo': correo, 'nombre': nombre, 'tipo': tipo})
    flash("Registration successful! You can now log in.")

# Generar el código QR
qr_data = f"usuario:{usuario},contraseña:{contraseña}"
encrypted_data = cipher_suite.encrypt(qr_data.encode())
qr = qrcode.QRCode(version=1, box_size=10, border=5)
qr.add_data(encrypted_data.decode())
qr.make(fit=True)
img = qr.make_image(fill_color="black", back_color="white")

# Guardar la imagen en un archivo temporal
temp_img_path = tempfile.mktemp(suffix='.png')
img.save(temp_img_path)

# Guardar la imagen en un buffer de bytes
img_byte_arr = io.BytesIO()
img.save(img_byte_arr, format='PNG')
img_byte_arr.seek(0)

# Generar el PDF con el código QR
pdf_byte_arr = io.BytesIO()
c = canvas.Canvas(pdf_byte_arr, pagesize=letter)
c.drawImage(temp_img_path, 100, 600, width=100, height=100)
c.save()
pdf_byte_arr.seek(0)

# Convertir la imagen QR a base64 para mostrarla en el HTML

```

```

        img_base64 = base64.b64encode(img_byte_arr.getvalue()).decode('utf-8')

        # Renderizar la página de registro con el código QR y el enlace de
        # descarga
        return render_template('register.html', qr_code=img_base64,
                               pdf_file=base64.b64encode(pdf_byte_arr.getvalue()).decode('utf-8'))

    return render_template('register.html')

@app.route('/download_pdf')
def download_pdf():
    # Obtener el PDF del contexto de la solicitud
    pdf_file = request.args.get('pdf_file')
    if pdf_file:
        pdf_byte_arr = io.BytesIO(base64.b64decode(pdf_file))
        return send_file(pdf_byte_arr, as_attachment=True,
                          download_name='qr_code.pdf', mimetype='application/pdf')
    else:
        return "Error: PDF file not found."

@app.route('/scan_qr')
def scan_qr():
    return render_template('scan_qr.html')

@app.after_request
def add_security_headers(response):
    response.headers['Content-Type'] = 'text/html'
    response.headers['X-Content-Type-Options'] = 'nosniff'
    return response

@app.route('/generate_qr', methods=['GET', 'POST'])
def generate_qr():
    if request.method == 'POST':
        usuario = request.form.get('usuario')
        contraseña = request.form.get('contraseña')

        # Buscar el usuario en la base de datos
        user = users.find_one({'usuario': usuario, 'contraseña': contraseña})
        if not user:
            user = teachers.find_one({'usuario': usuario, 'contraseña':
contraseña})

        if user:
            # Generar el código QR
            qr_data = f"usuario:{usuario},contraseña:{contraseña}"
            encrypted_data = cipher_suite.encrypt(qr_data.encode())
            qr = qrcode.QRCode(version=1, box_size=10, border=5)
            qr.add_data(encrypted_data.decode())

```

```

qr.make(fit=True)
img = qr.make_image(fill_color="black", back_color="white")

# Guardar la imagen en un archivo temporal
temp_img_path = tempfile.mktemp(suffix='.png')
img.save(temp_img_path)

# Guardar la imagen en un buffer de bytes
img_byte_arr = io.BytesIO()
img.save(img_byte_arr, format='PNG')
img_byte_arr.seek(0)

# Convertir la imagen QR a base64 para mostrarla en el HTML
img_base64 = base64.b64encode(img_byte_arr.getvalue()).decode('utf-
8')

# Renderizar la página con el código QR y el enlace de descarga
return render_template('generate_qr.html', qr_code=img_base64)
else:
    flash("Usuario o contraseña incorrectos.")
    return render_template('generate_qr.html')

return render_template('generate_qr.html')

@app.route('/upload_qr', methods=['POST'])
def upload_qr():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(request.url)
    if file:
        # Guardar el archivo en un archivo temporal
        temp_file_path = tempfile.mktemp(suffix='.' +
file.filename.split('.')[1])
        file.save(temp_file_path)

```