



## Proyecto 1

---

29-Diciembre-2025

---



## Índice

<b>Proyecto 1</b>	<b>1</b>
Índice	2
Introducción	5
Objetivos	6
Objetivo General	6
Objetivos Específicos	6
Nombre Del Proyecto	7
Descripción General de la aplicación	7
Responsables del proyecto	7
PALABRAS CLAVE	8
Requerimientos Técnicos	8
Herramientas y lenguajes utilizados para el desarrollo.	10
Backend 1 – Java (versión 21)	10
Backend 2 – Python (versión estable LTS, 20.2)	10
Frontend – Angular (versión 18.1)	11
Swagger	11
Postman	11
Git & GitHub	11
AWS (Amazon Web Services)	11
EC2 (Elastic Compute Cloud)	12
Diagramas	13
Casos de Uso	13
Registrar Usuario CU01	13
Iniciar Sesión (Autenticación 2FA) CU02	13
Recuperar Contraseña CU03	13
Agregar Productos al Carrito CU05	14
Realizar Compra CU06	14
Gestión de Inventario CU07	14
Generar Reportes de Ventas CU08	15
Consultar disponibilidad de citas CU09	15
Diagramas de Casos de Uso	16
Clases	24
Actividades	24
Arquitectura	32
Mysql AURORA AWS	32



---

Configuración para EC2	33
Bucket de archivos	33
Bucket de frontend	34
Configuración para DB	34
Target Group	35
LOAD BALANCER	35
Zonas para redundancia	35
USUARIOS IAM	36
Grupo de frontend	36
Grupo de backend	37
Diagrama Base de datos	39
Diagrama ER	39
Rutas y descripción de endpoint	40
Instalación Del Sistema	47
API REST	48
Despliegue y Mantenimiento	48
Despliegue	48
Backend	48
Mantenimiento	48
Mantenimiento	49
Backend (Java 21 y Python)	49
Frontend (Angular )	49
Base de Datos (RDS – MySQL)	49
Despliegue y Nube (AWS)	50
Gestión de Versiones (Git & GitHub)	50
Git y github	51
Referencias y recursos	52



## Introducción

El presente manual técnico documenta el desarrollo e implementación de la plataforma “**My PsiFirm**”, una plataforma para la gestión de citas y sesiones para la gestión integral de una firma de psicólogos que ofrece servicios clínicos, diagnósticos y actividades de apoyo terapéutico. La aplicación busca digitalizar y centralizar el manejo de empleados (psicólogos con distintas especialidades, personal administrativo y de mantenimiento), la nómina (pagos, bonos, descuentos de IGSS), la administración de pacientes (historial clínico, medicamentos, condiciones, tareas/actividades/advertencias y notas), facturación y control de pagos por los servicios, así como el inventario y venta de medicamentos o herramientas terapéuticas.

En el plano tecnológico, el sistema se despliega sobre **Amazon Web Services (AWS)**, aprovechando servicios como **S3** para almacenamiento de archivos y hosting, **EC2** para servidores de aplicación con balanceo de carga, **RDS** para la gestión de base de datos, y **IAM** para el control de accesos y seguridad. La arquitectura propuesta garantiza **escalabilidad, disponibilidad, seguridad y rendimiento**, ofreciendo una experiencia confiable tanto para clientes como para administradores.

Este manual servirá como guía para comprender la estructura del sistema, sus configuraciones principales, las tecnologías utilizadas y los procesos necesarios para mantener, actualizar y escalar la aplicación en un entorno de producción.



## Objetivos

### Objetivo General

Desarrollar y documentar una plataforma web segura y escalable que permita la gestión de citas y sesiones.

### Objetivos Específicos

1. Implementar un sistema de **autenticación y registro seguro**, incluyendo cifrado de contraseñas, validación por correo electrónico y recuperación de credenciales.
2. Desplegar la aplicación en **AWS**, configurando servicios como S3, EC2, Load Balancer, RDS e IAM para garantizar alta disponibilidad y seguridad.
3. Implementar un **módulo de reportes** para los administradores que permita analizar ventas, usuarios y medicamentos más relevantes.



---

## Nombre Del Proyecto

### FIRMA PSICOLÓGICA (PsiFirm)

## Descripción General de la aplicación

PsiFirm es una plataforma para la gestión integral de una firma de psicólogos que ofrece servicios clínicos, diagnósticos y actividades de apoyo terapéutico. La aplicación busca digitalizar y centralizar el manejo de empleados (psicólogos con distintas especialidades, personal administrativo y de mantenimiento), la nómina (pagos, bonos, descuentos de IGSS), la administración de pacientes (historial clínico, medicamentos, condiciones).

El sistema está diseñado para funcionar con dos tipos de usuarios principales:

- **Clientes:** pueden registrarse, autenticarse y acceder a funciones como navegación por catálogo, filtros avanzados, carrito de compras, preventas, participación en promociones, inscripción en eventos y aporte de reseñas mediante comentarios y calificaciones.
- **Administradores:** cuentan con privilegios para gestionar artículos (alta, edición y baja), administrar usuarios y comentarios, crear y controlar eventos de escucha colectiva, definir promociones y consultar reportes analíticos sobre ventas y desempeño de la plataforma.

## Responsables del proyecto

Jhony Roel Fuentes Lopez
--------------------------



---

## PALABRAS CLAVE

- **Java:** Entorno de ejecución para el backend
- **Mysql:** Base de datos relacional.
- **JWT:** Sistema de autenticación basado en tokens.
- **2FA:** Autenticación de dos factores para mayor seguridad.
- **AWS S3:** Almacenamiento de imágenes en la nube.
- **Angular:** Framework para el frontend.
- **Git:** Sistema de control de versiones.
- **GitHub:** Plataforma para alojar repositorios de Git.

## Requerimientos Técnicos

**Procesador:** core i5 gen o superior

**Memoria RAM:** 2 Gb

**Espacio de almacenamiento:** 10 Gb

**Sistema Operativo**

**Otros:** Tener instaladas las siguientes herramientas.

### Python

**Instalación en Windows:** Descarga el instalador desde [nodejs.org](https://nodejs.org) y sigue las instrucciones.

**Instalación en Linux:** Usa los siguientes comandos:

```
sudo apt update  
sudo apt install python3
```

### Mysql

Base de datos relacional para la gestión de datos como usuarios, artículos y preventas.

Instalación en Windows: Descarga e instala desde [postgresql.org](https://www.postgresql.org).

Instalación en Linux:

```
sudo apt update  
sudo apt install mysql-server
```

### Angular

Framework para el desarrollo de frontend en aplicaciones web.



---

Instalación en Windows y Linux: Necesitas tener Node.js instalado, luego ejecuta:

```
npm install react angular-dom  
npm install --save-dev @babel/preset-react
```

### **JWT (JSON Web Tokens)**

Método de autenticación basado en tokens para usuarios.

No requiere instalación, se implementa como una librería en el proyecto:

```
npm install jsonwebtoken
```

### **2FA (Two-Factor Authentication)**

```
npm install speakeasy qrcode
```

**Para java en el pom.xml:**

```
<dependency>  
  <groupId>io.jsonwebtoken</groupId>  
  <artifactId>jjwt-impl</artifactId>  
  <version>0.11.5</version>  
  <scope>runtime</scope>  
</dependency>
```

### **Swagger**

Descripción: Herramienta para documentar y probar APIs REST.

Instalación en Windows y Linux: Se usa como dependencia en el backend:

```
<dependency>  
  <groupId>org.springdoc</groupId>  
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
  <version>2.7.0</version>  
</dependency>
```

### **Git**

Sistema de control de versiones para gestionar el código.





Instalación en Windows:

Descarga desde [git-scm.com](https://git-scm.com).

Instalación en Linux:

```
sudo apt update  
sudo apt install git
```

### **Postman**

Plataforma popular utilizada por desarrolladores para probar y depurar APIs.

Instalación en windows:

Visita el sitio web oficial de Postman: [Postman](https://www.postman.com).

Descarga el instalador para Windows (archivo .exe).

Instalación en Linux:

```
sudo snap install postman
```

## Herramientas y lenguajes utilizados para el desarrollo.

### **Backend 1 – Java (versión 21)**

Java es un lenguaje de programación ampliamente utilizado en el desarrollo de aplicaciones empresariales. Su robustez, seguridad y soporte para programación orientada a objetos lo hacen ideal para servicios críticos. En este proyecto se utilizó para implementar uno de los servidores backend, encargado de gestionar la lógica de negocio y exponer endpoints RESTful.

### **Backend 2 – Python (versión estable LTS, 20.2)**

Python es un lenguaje de programación de alto nivel, interpretado y de código abierto, conocido por su sintaxis clara y legible, similar al lenguaje humano, lo que facilita su aprendizaje y uso. Es extremadamente versátil y se utiliza en desarrollo web, ciencia de datos, inteligencia artificial, automatización, ciberseguridad y más, gracias a su potencia y a un vasto ecosistema de bibliotecas.



## Frontend – Angular (versión 18.1)

React es una librería de JavaScript desarrollada por Facebook, orientada a la construcción de interfaces de usuario basadas en componentes. Permite el desarrollo de aplicaciones web dinámicas y reactivas (SPA – Single Page Applications). En el proyecto se utilizó para el frontend, ofreciendo una experiencia interactiva y fluida a los usuarios.

## Swagger

Swagger es una herramienta que permite documentar y probar APIs RESTful de forma interactiva. En este proyecto se utilizó para documentar los servicios backend y facilitar la comunicación entre los desarrolladores y testers.

## Postman

Postman es una herramienta utilizada para el diseño, prueba y documentación de APIs. Se empleó en el proyecto para verificar el correcto funcionamiento de los endpoints desarrollados en los dos backends antes de su integración con el frontend.

## Git & GitHub

Git es un sistema de control de versiones distribuido que permite gestionar cambios en el código fuente. GitHub, como plataforma de alojamiento, se utilizó para centralizar el repositorio, coordinar el trabajo en equipo y llevar control de versiones mediante ramas y pull requests.

## AWS (Amazon Web Services)

AWS fue el proveedor de servicios en la nube utilizado para el despliegue de la aplicación. La infraestructura incluye:

- **EC2:** se desplegaron dos instancias para los servidores backend (Java y Node.js). Cada instancia corre en un entorno independiente y se conectan a través de un balanceador de carga.
- **RDS:** como motor de base de datos relacional para almacenar usuarios, artículos, pedidos, comentarios y eventos.



- 
- **S3**: utilizado tanto para el **hosting del frontend** como para el **almacenamiento de archivos multimedia** (imágenes, audios y PDFs).

## **EC2 (Elastic Compute Cloud)**

EC2 es el servicio de AWS que permite ejecutar servidores virtuales en la nube. En este proyecto se utilizó para desplegar los dos backends (Java y Node.js), configurando los **Security Groups** para habilitar únicamente los puertos necesarios. Estas instancias se encuentran detrás de un balanceador de carga para garantizar disponibilidad y tolerancia a fallos.



---

## Diagramas

### Casos de Uso

#### Registrar Usuario CU01

- **Descripción:** Permite al usuario registrarse en el Sitio
- **Actores Involucrados:** Usuario
- **Flujo Principal:**
  - El usuario accede al formulario de registro.
  - El usuario completa la información requerida (nombre, apellido, email, contraseña, dirección, nit).
  - El sistema valida la información.
  - El sistema crea la cuenta para el usuario y lo redirige a la página de inicio.

#### Iniciar Sesión (Autenticación 2FA) CU02

- **Descripción:** Permite al usuario iniciar sesión en el sistema con autenticación de dos factores (2FA).
- **Actores involucrados:** Usuario, Comprador.
- **Flujo Principal:**
  - El usuario accede al formulario de inicio de sesión.
  - El usuario ingresa sus credenciales (correo electrónico y contraseña).
  - El sistema valida las credenciales.
  - El sistema solicita un código de verificación adicional (enviado por SMS o correo electrónico).
  - El usuario ingresa el código de verificación.
  - El sistema verifica el código 2FA.
  - El sistema autentica al usuario y lo redirige a su área de página principal.

#### Recuperar Contraseña CU03

- **Descripción:** Permite al usuario recuperar su contraseña en caso de olvido.
- **Actores involucrados:** Usuario.
- **Flujo Principal:**
  - El usuario selecciona la opción de "Recuperar contraseña".
  - El usuario ingresa su correo electrónico registrado.
  - El sistema envía un código de recuperación de contraseña al correo electrónico.
  - El usuario ingresa el código y restablecer su contraseña.
  - El sistema valida y confirma el restablecimiento de la contraseña.
  - El usuario puede iniciar sesión con la nueva contraseña.



### Agregar Productos al Carrito CU05

- **Descripción:** Permite al usuario agregar productos a su carrito de compras.
- **Actores involucrados:** Usuario.
- **Flujo Principal:**
  - El usuario navega por la tienda y selecciona un artículo.
  - El usuario ingresa la cantidad deseada.
  - El sistema agrega el producto al carrito.
  - El usuario continúa navegando o accede al carrito de compras.

### Realizar Compra CU06

- **Descripción:** Permite al usuario realizar una compra de los artículos agregados al carrito.
- **Actores involucrados:** Cliente.
- **Flujo Principal:**
  - El usuario accede al carrito de compras.
  - El usuario verifica los artículos seleccionados, las cantidades y el total a pagar.
  - El sistema procesa el pago y confirma la compra.
  - El usuario recibe su factura

### Gestión de Inventario CU07

- **Descripción:** Permite a los administradores gestionar el inventario de la tienda.
- **Actores involucrados:** Administrador, Ayudante.
- **Flujo Principal:**
  - El administrador accede a la sección de gestión de productos.
  - El sistema muestra el inventario actual.
  - El administrador selecciona un producto para actualizar la cantidad o modificar detalles.
  - El sistema guarda los cambios y actualiza el inventario.



---

### Generar Reportes de Ventas CU08

- **Descripción:** Permite al administrador generar reportes.
- **Actores involucrados:** Administrador.
- **Flujo Principal:**
  - El administrador accede al dashboard de administración.
  - El administrador selecciona la opción para generar reportes.
  - El administrador genera el reporte.
  - El sistema exporta el reporte en el formato solicitado

### Consultar disponibilidad de citas CU09

**Descripción:** Permite consultar horarios disponibles para agendar una cita por servicio y fecha.

**Actores involucrados:** Paciente.

**Flujo Principal:**

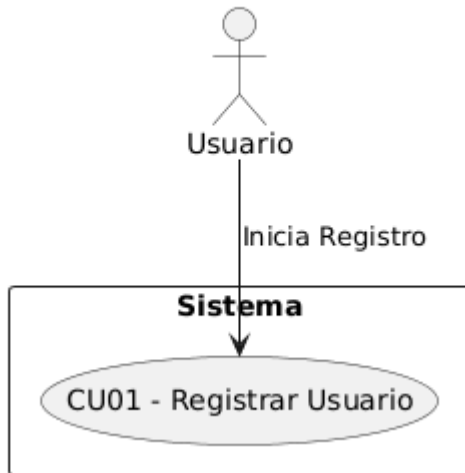
1. El paciente selecciona el servicio y la fecha.
2. El paciente solicita disponibilidad.
3. El sistema calcula/consulta los horarios disponibles.



## Diagramas de Casos de Uso

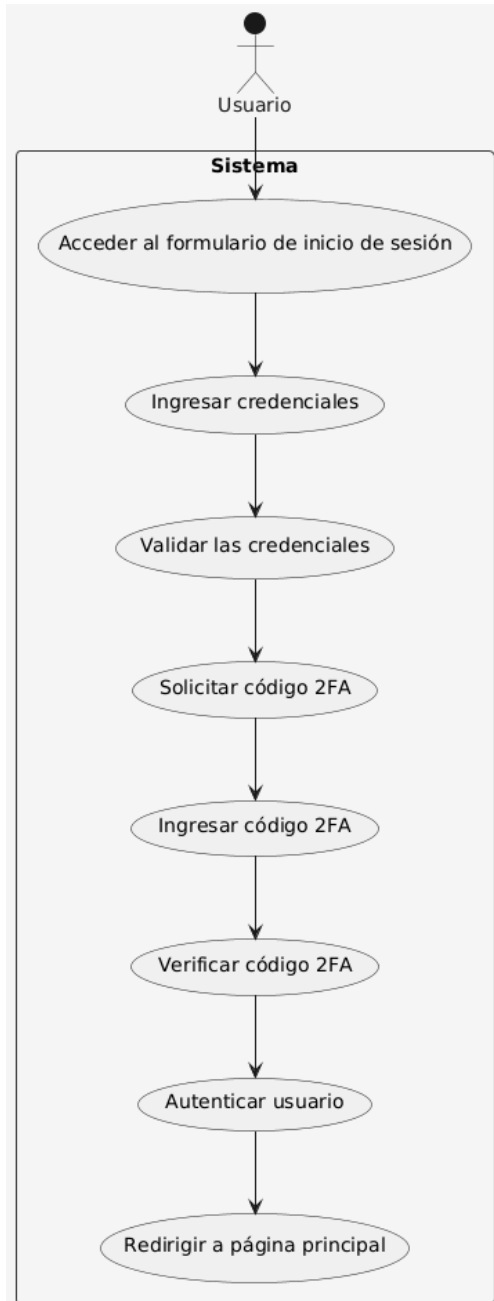
CU01

### Caso de Uso: CU01 - Registrar Usuario





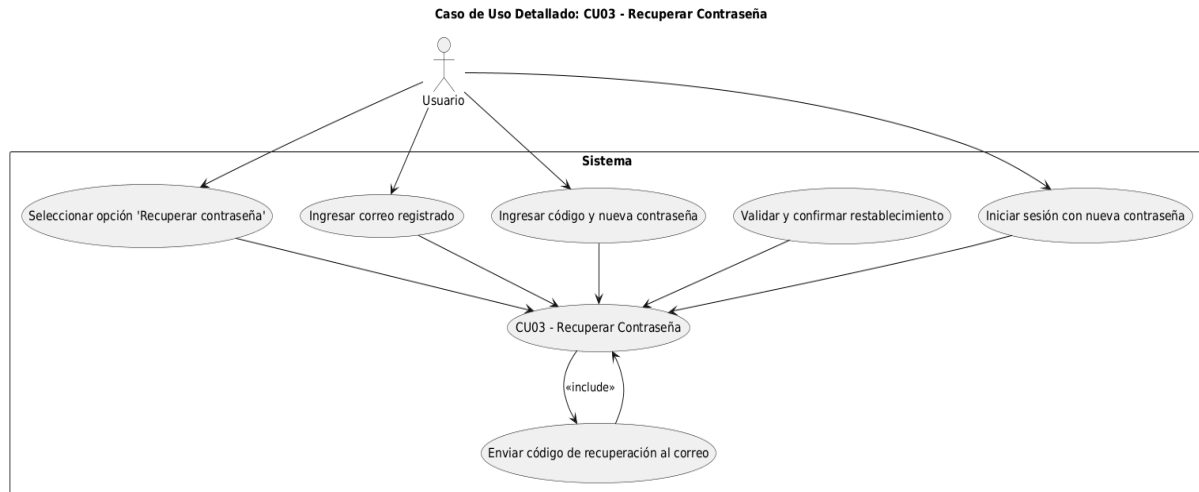
CU02





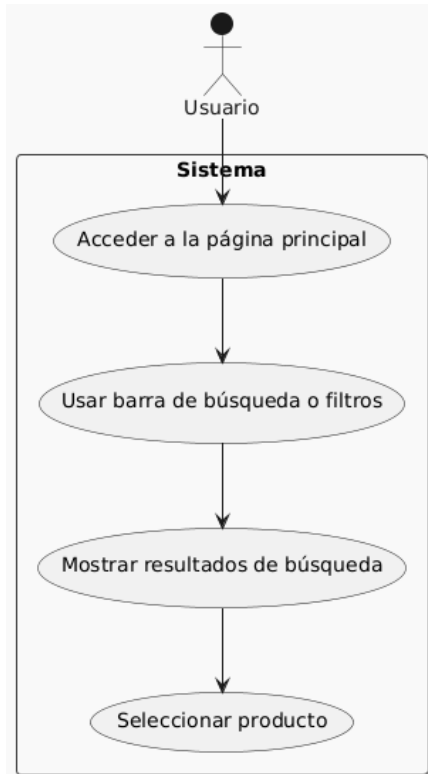


## CU03



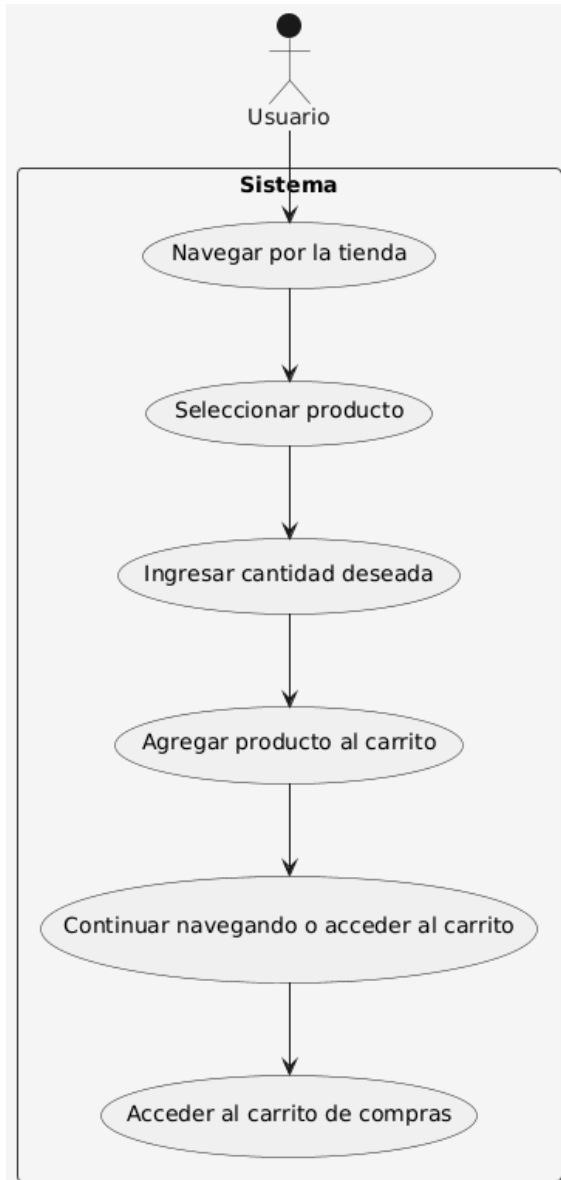


CU04





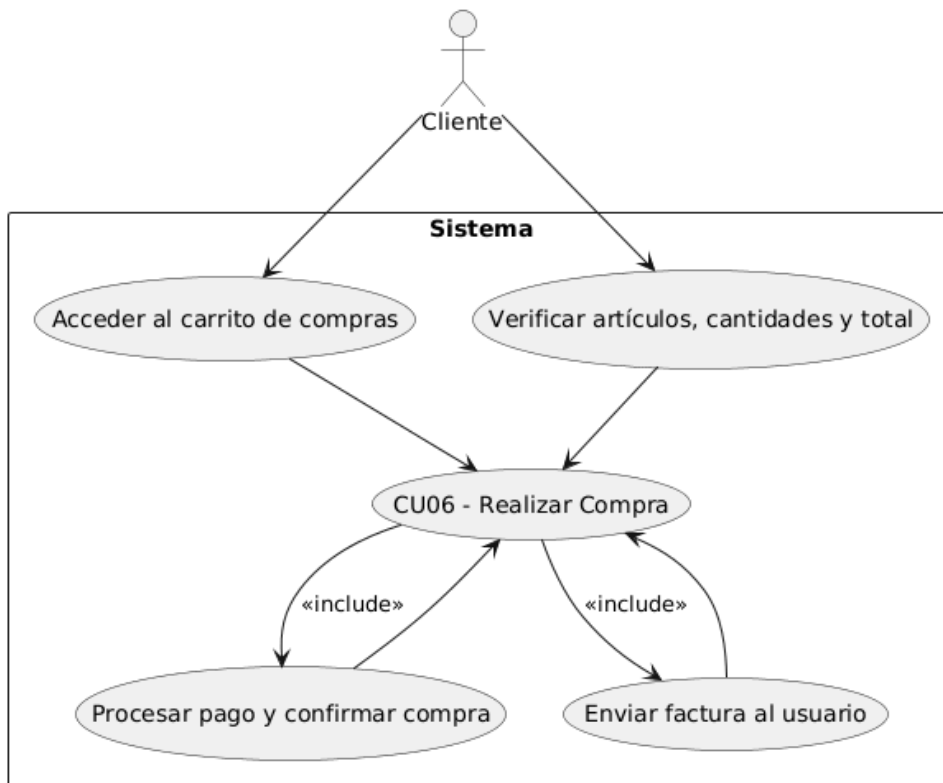
CU05





CU06

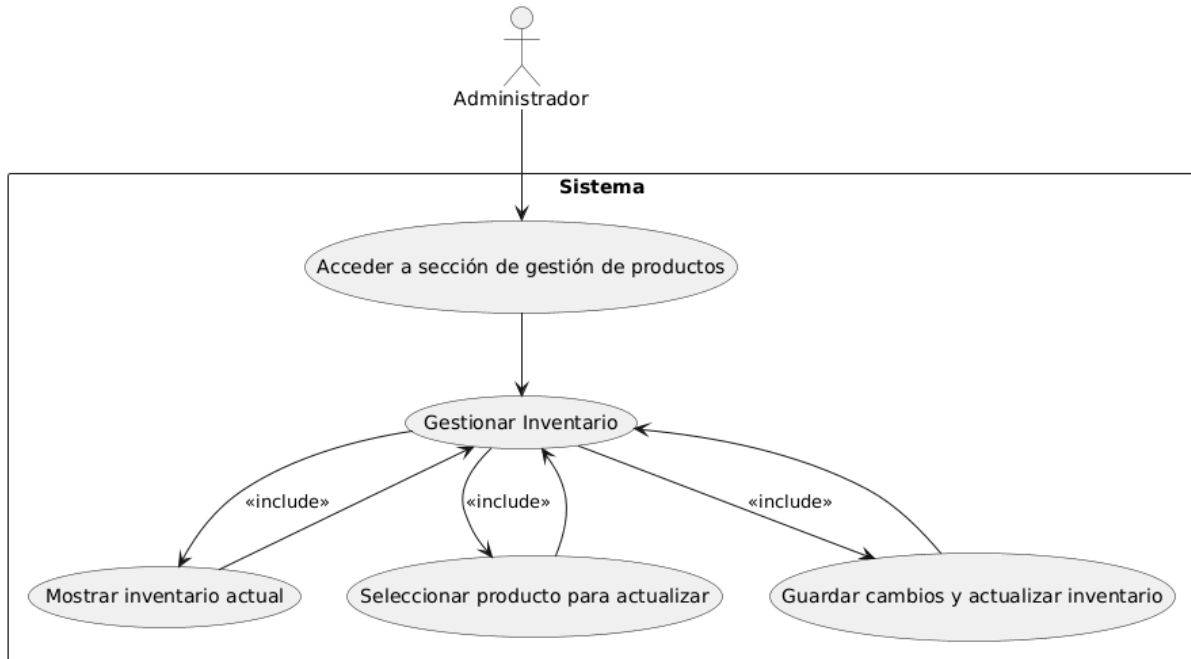
**Caso de Uso: CU06 - Realizar Compra**





CU07

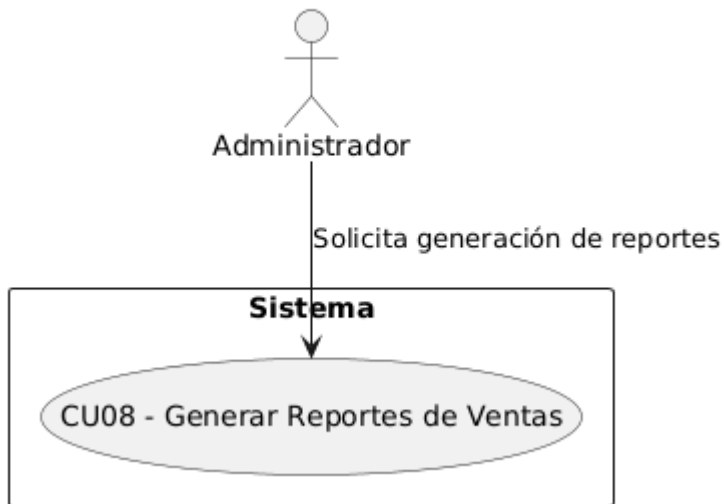
**Caso de Uso Detallado: Gestionar Inventario**



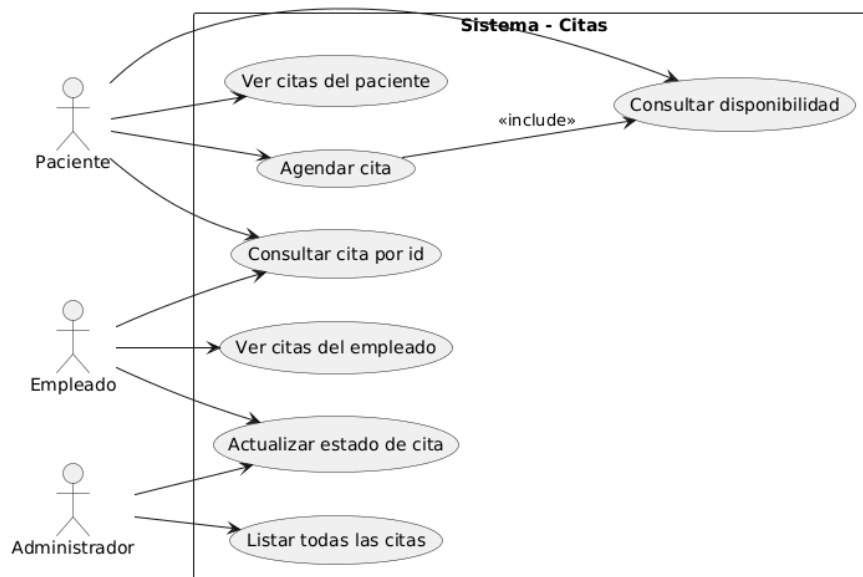


## CU08

### Caso de Uso: CU08 - Generar Reportes de Ventas

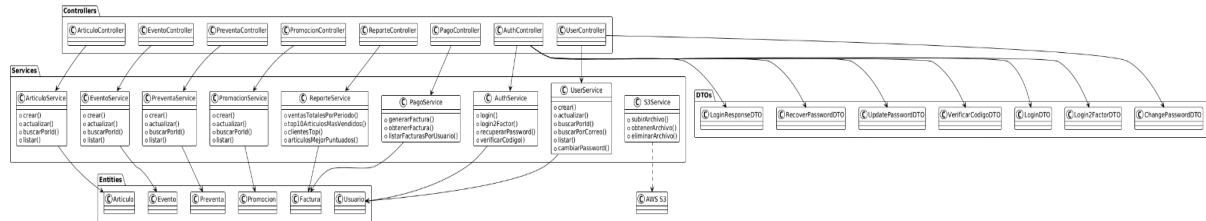


## CU09



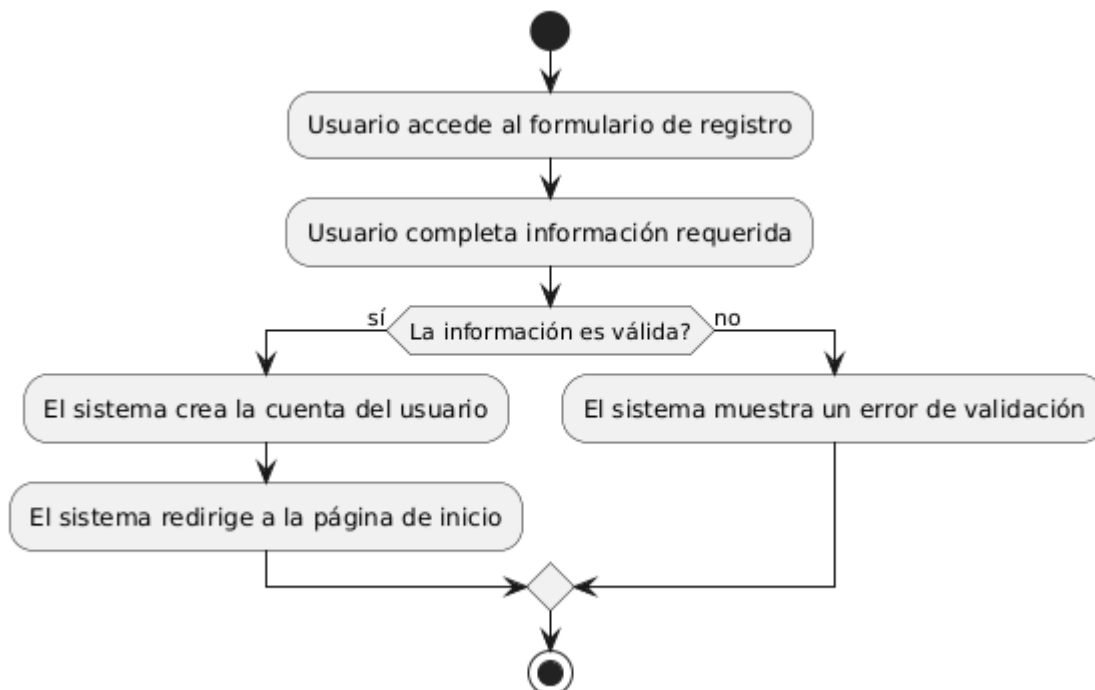


## Clases

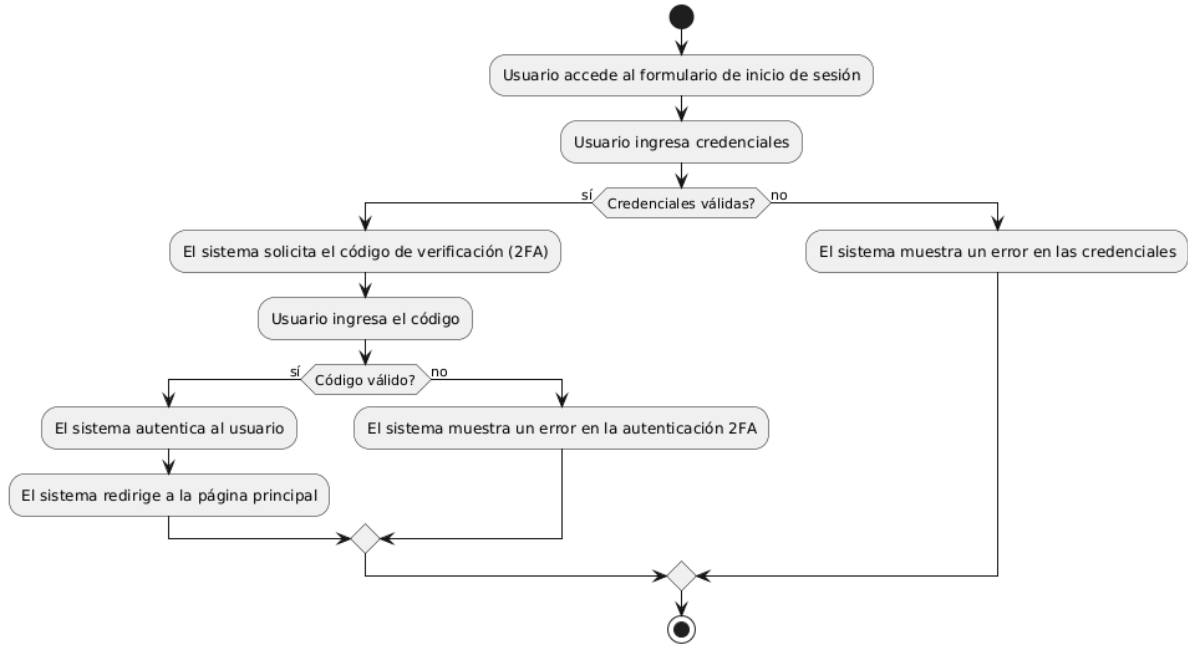


## Actividades

### CU01



### CU02

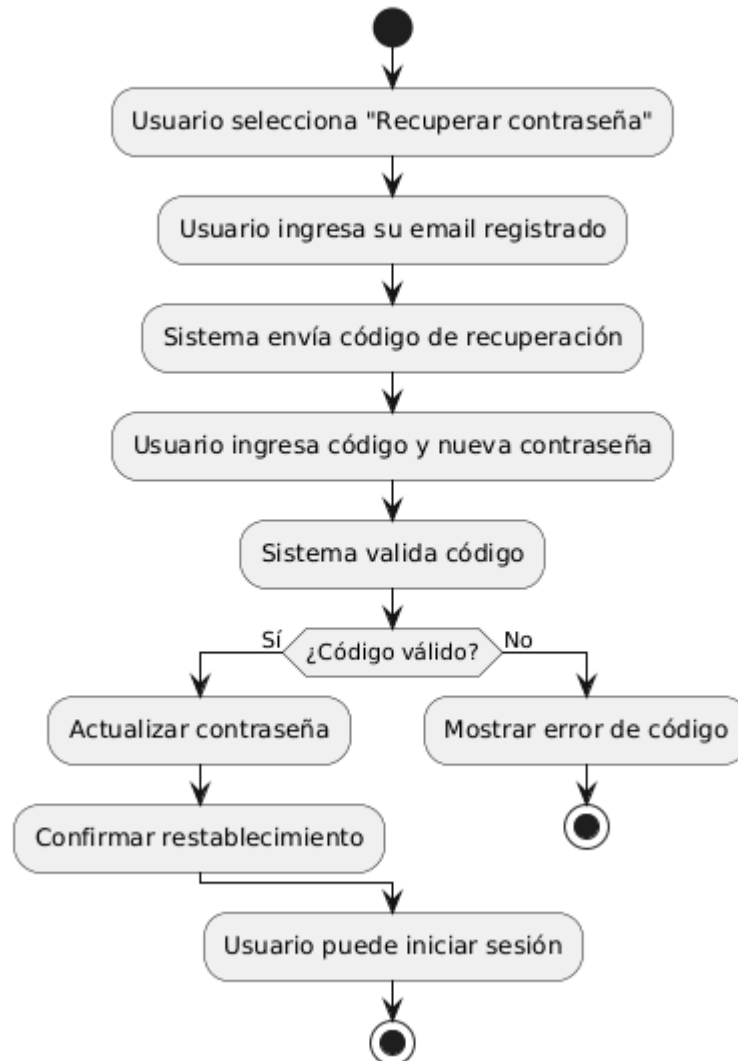


CU03





### Diagrama de Actividades - CU03: Recuperar Contraseña





CU04



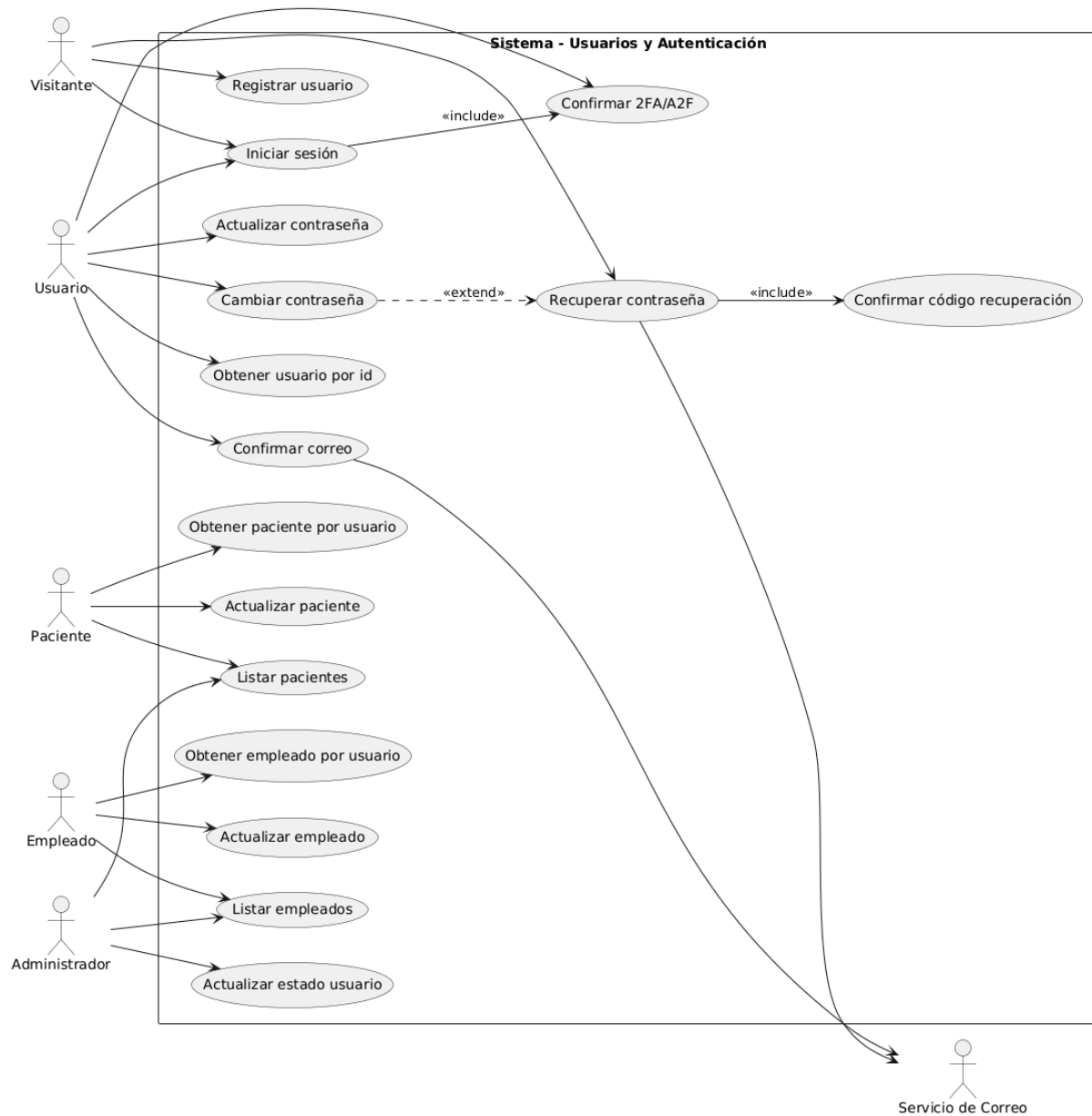


CU05





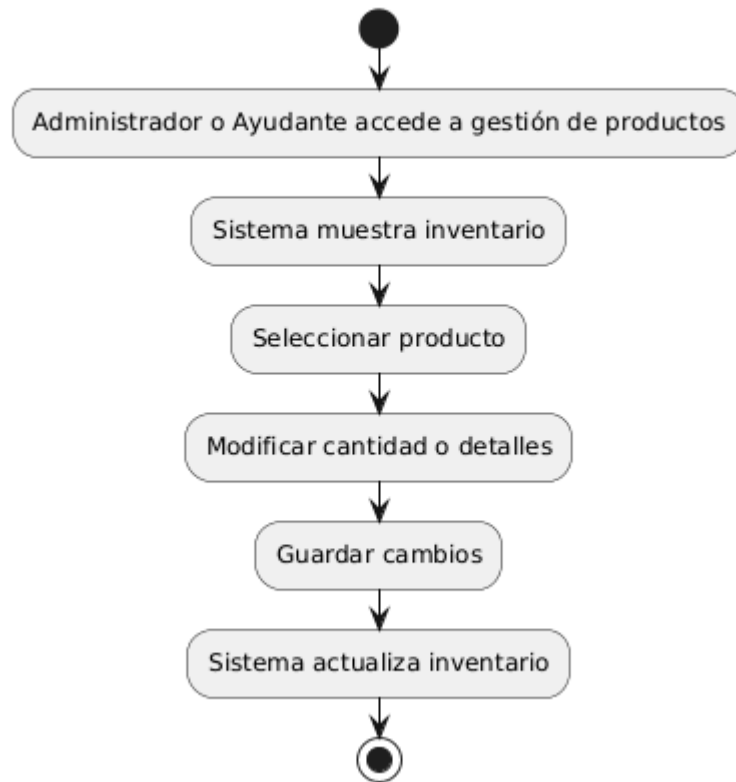
CU06





CU07

**Diagrama de Actividades - CU07: Gestión de Inventario**



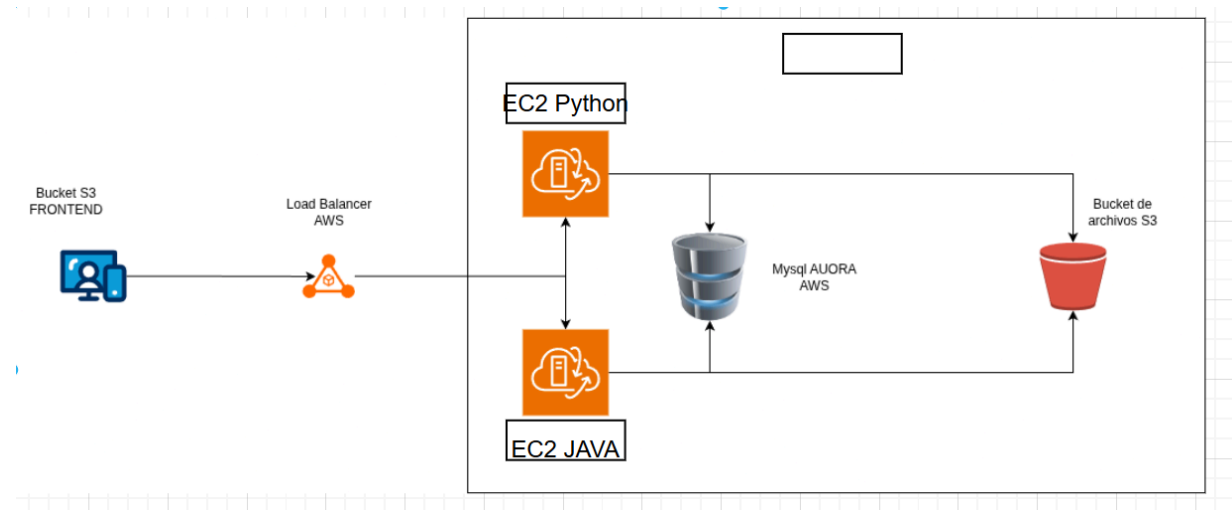


CU08





## Arquitectura



## LOAD BALANCER

**Destinos registrados (2)** [Info](#) [Mitigación de anomalías: No aplicable](#) [Anular el registro](#) [Registrar destinos](#)

Los grupos de destinos enrutan las solicitudes a destinos individuales registrados mediante el protocolo y el número de puerto que especifique. Las comprobaciones de estado se realizan en todos los destinos registrados de acuerdo con la configuración de comprobación de estado del grupo de destinos. La detección de anomalías se aplica automáticamente a los grupos de destinos de HTTP/HTTPS con al menos 3 destinos en buen estado.

<input type="checkbox"/>	ID de instancia	Nombre	Puerto	Zona	Estado	Detalles del estado	Su
<input type="checkbox"/>	<a href="#">i-0a7840a84cf497584</a>	backen2	3000	us-east-2c (use...)	Unhealthy	Health checks failed	⊖
<input type="checkbox"/>	<a href="#">i-004d5d6ba73ef0954</a>	backen1	3000	us-east-2c (use...)	Unhealthy	Health checks failed	⊖

## Mysql AURORA AWS

**Engine:** MySQL Community

**Engine version:** 8.0.42



## database-1



### Resumen

Identificador de base de datos  
database-1

CPU  
3.02%

Estado  
Disponible  
Clase  
db.t4g.micro

Rol  
Instancia  
Actividad actual  
2 Conexiones

Motor  
MySQL Community  
Región y AZ  
us-east-2b

Recomendaciones

< [Conectividad y seguridad](#) Supervisión Registros y eventos Configuración Integraciones sin extracción, trans >

## Configuración para EC2

Filter by compute resources				
Resource identifier	Resource type	Availability Zone	VPC security group	Compute resource security
<a href="#">i-00668d7801d86e4fc</a>	EC2 instance	<a href="#">us-east-1d</a>	<a href="#">rds-ec2-1</a>	<a href="#">ec2-rds-1</a>
<a href="#">i-0aacac9104d7d89e5</a>	EC2 instance	<a href="#">us-east-1d</a>	<a href="#">rds-ec2-1</a>	<a href="#">ec2-rds-1</a>

## Bucket de archivos

Nombre: seminario1datos

Configuración: block access public

**seminario1datos** [Información](#)

[Objetos](#) [Metadatos](#) [Propiedades](#) [Permisos](#) [Métricas](#) [Administración](#) [Puntos de acceso](#)

**Objetos (3)**

[Copiar URI de S3](#) [Copiar URL](#) [Descargar](#) [Abrir](#) [Eliminar](#) [Acciones](#) [Crear carpeta](#)

[Cargar](#)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	<a href="#">area_PEDIATRIA 2/</a>	Carpeta	-	-	-
<input type="checkbox"/>	<a href="#">Fotos/</a>	Carpeta	-	-	-
<input type="checkbox"/>	<a href="#">IMAGEN/</a>	Carpeta	-	-	-





## Bucket de frontend

**Nombre:** frontendss1

**Configuración:** Static website hosting

**Política:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::frontendss1/*"
    }
  ]
}
```

## Configuración para DB

Reglas de entrada (5)							
<input type="text" value="Buscar"/>							
<input type="checkbox"/>	Name	ID de la regla del gr...	Versión de IP	Tipo	Protocolo	Interv...	
<input type="checkbox"/>	-	sgr-07f2d7cacbe7acf9d	IPv4	MYSQL/Aurora	TCP	3306	
<input type="checkbox"/>	-	sgr-03d47477aa6ae2570	IPv4	SSH	TCP	22	
<input type="checkbox"/>	-	sgr-048c054053219dd3f	IPv4	TCP personalizado	TCP	8080	
<input type="checkbox"/>	-	sgr-0b55bf428a90dd3c6	IPv4	TCP personalizado	TCP	3000	
<input type="checkbox"/>	-	sgr-0c5a0ec30950e3511	IPv4	HTTP	TCP	80	

## Target Group

**Nombre:** launch-wizard-2

Reglas de entrada (5)

⌂

Administrar etiquetas

Editar reglas de entrada

Q Buscar

< 1 > ⚙

<input type="checkbox"/>	Name	ID de la regla del gr...	Versión de IP	Tipo	Protocolo	Interv...
<input type="checkbox"/>	-	sgr-07f2d7cacbe7acf9d	IPv4	MYSQL/Aurora	TCP	3306
<input type="checkbox"/>	-	sgr-03d47477aa6ae2570	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-048c054053219dd3f	IPv4	TCP personalizado	TCP	8080
<input type="checkbox"/>	-	sgr-0b55bf428a90dd3c6	IPv4	TCP personalizado	TCP	3000
<input type="checkbox"/>	-	sgr-0c5a0ec30950e3511	IPv4	HTTP	TCP	80

## LOAD BALANCER

**Nombre:** balanceador

Zonas para redundancia

Zonas de disponibilidad y subredes Info

Editar las subredes

La selección de dos o más zonas de disponibilidad y las subredes correspondientes aumenta la tolerancia a errores de las aplicaciones.

Zona	Subred	Dirección IPv4 privada	Dirección IPv6
us-east-2b (use2-az2)	<a href="#">subnet-03590117a6c0fff2f</a>	Asignado desde el CIDR 172.31.16...	No aplicable
us-east-2c (use2-az3)	<a href="#">subnet-07f2f8fb8af109f2a</a>	Asignado desde el CIDR 172.31.32...	No aplicable
us-east-2a (use2-az1)	<a href="#">subnet-053969948504ef7ed</a>	Asignado desde el CIDR 172.31.0.0...	No aplicable



## balanceador



### ▼ Detalles

**Tipo de equilibrador de carga**  
Aplicación

**Esquema**  
Internet-facing

**Estado**  
✓ Activo

**Zona hospedada**  
Z3AADJGX6KTTL2

**VPC**  
[vpc-0d31f47bf84f2ff81](#)

**Zonas de disponibilidad**  
[subnet-03590117a6c0ff2f](#) us-east-2b (use2-az2)  
[subnet-07f2f8fb8af109f2a](#) us-east-2c (use2-az3)  
[subnet-053969948504ef7ed](#) us-east-2a (use2-az1)

**Tipo de dirección IP del equilibrador de carga**  
IPv4

**Fecha creada**  
28 de diciembre de 2025, 19:54 (UTC-06:00)

**ARN del equilibrador de carga**  
[arn:aws:elasticloadbalancing:us-east-2:464218211567:loadbalancer/app/balanceador/ee1ff9e15732f4ed](#)

**Nombre de DNS** [info](#)  
[balanceador-538318693.us-east-2.elb.amazonaws.com](#) (Registro A)

## USUARIOS IAM

### Grupo de frontend

**Nombre:** frontend

**Detalle:** Acceso únicamente el bucket donde está desplegado de manera estática el frontend

**Políticas:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "FrontendS3Access",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::bpmn-semi1-g6"
      ]
    }
  ]
}
```



```
    ],
    },
    {
        "Sid": "FrontendS3Objects",
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject"
        ],
        "Resource": [
            "arn:aws:s3:::bpmn-semi1-g6/*"
        ]
    },
    {
        "Sid": "FrontendCloudFront",
        "Effect": "Allow",
        "Action": [
            "cloudfront:CreateInvalidation"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

## Grupo de backend

**Nombre:** ss1-backend-developers

**Detalle:** Acceso únicamente al bucket donde están los archivos que se suben desde el backend y el poder listar y ver el estado de las máquinas EC2

**Políticas:**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

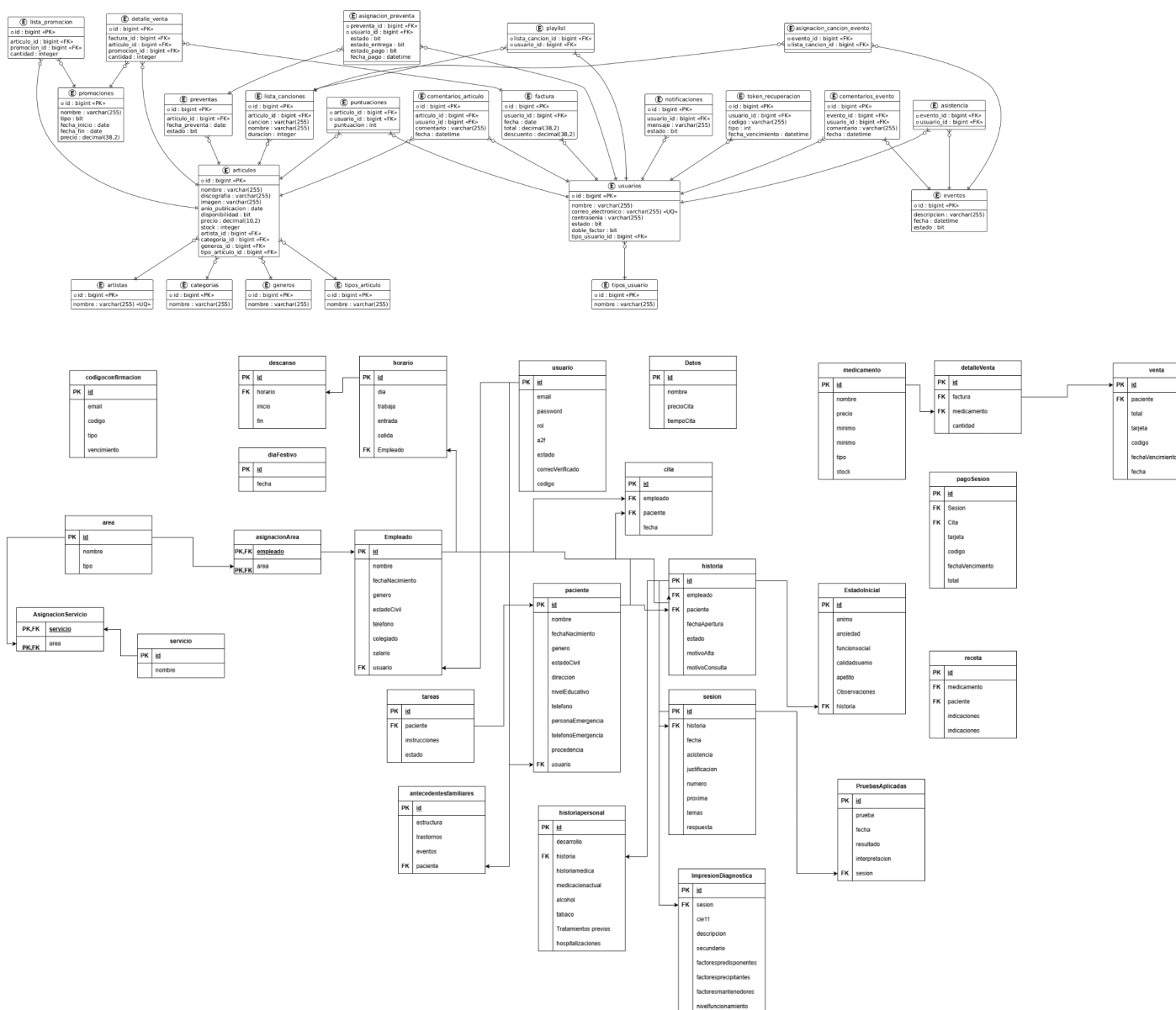


```
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListAllMyBuckets"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject"
        ],
        "Resource": "*"
    }
]
```



## Diagrama Base de datos

### Diagrama ER





## Rutas y descripción de endpoint

Descripción	Método HTTP	Ruta	Query (requeridos *)	Body (JSON)	Respuesta OK
asignarServicios	POST	/api/admin/servicios/asignar		AsignarServicio	
saveArea	POST	/api/admin/area		Area	Area
saveServicio	POST	/api/admin/servicio		Servicio	Servicio
Login	POST	/api/auth/login		Login	object
confirmar	POST	/api/auth/login/a2f		ConfirmarCorreoRequest	object
agendar	POST	/api/cita/agendar		CitaCreate	
findById	GET	/api/cita/id	id*		Cita
getAllCitas	GET	/api/cita			array[Cita]
getDisponibles	GET	/api/cita/disponibilidad	idServicio*, fecha*		array[DisponibilidadCita]
getMyCitasEmpleado	GET	/api/cita/empleado	id*		array[Cita]
getMyCitasPaciente	GET	/api/cita/paciente	id*		array[Cita]



updateState	POST	/api/cita/update		UpdateState
asignarArea	POST	/api/empleado/asignarArea		AsignacionAreaDTO
asignarHorario	POST	/api/empleado/asignarHorario		array[AsignacionHorarioDTO]
findById	GET	/api/empleado/id	id*	Empleado
getAllEmpleados	GET	/api/empleado		array[Empleado]
getAreas	GET	/api/empleado/areas	id*	array[Area]
getHorarios	GET	/api/empleado/horario	id*	array[AsignacionHorarioDTO]
updateSalario	PUT	/api/empleado/salario		UpdateSalario
getEmpresaData	GET	/api/empresa		Empresa
getPrincipal	GET	/api/empresa/dashboard		PrincipalDTO
updateEmpresa	PUT	/api/empresa		Empresa
getAllAreas	GET	/api/general/areas		array[Area]
getServicios	GET	/api/general/servicios		array[Servicio]





getServiciosAsignados	GET	/api/general/area/servicios	areald*	array[Servicio]
completarReceta	POST	/api/historia/area/completar	UpdateState	
crearSesion	POST	/api/historia/sesion	Sesion	
darAlta	POST	/api/historia/darAlta	DarAlta	
findByEmpleado	GET	/api/historia/empleado	id*	array[Historia]
findDetails	GET	/api/historia/details	id*	
findByPaciente	GET	/api/historia/paciente	id*	array[Historia]
findHistoriaById	GET	/api/historia/id	id*	Historia
findSesionById	GET	/api/historia/sesion/id	id*	Sesion
getAll	GET	/api/historia		array[Historia]
getAllSesiones	GET	/api/historia/sesion		array[Sesion]
getDetailSesion	GET	/api/historia/sesion/details	id*	Sesion Detail
getHorarios	GET	/api/historia/horarios	id*, fecha*, duracion*	array[string]
getRecetas	GET	/api/historia/receta	id*	array[Receta]



getSesion ByHistoria	GET	/api/historia/ sesionHistori a	id*		array[S esion]
getSesion ByPacient eld	GET	/api/historia/ sesion/pacie nteld	id*		array[S esion]
getTareas	GET	/api/historia/t area	id*		array[Ta rea]
guardarIm presion	POST	/api/historia/i mpresion		ImpresionDiagno stica	
guardarPr ueba	POST	/api/historia/ prueba		PruebasAplicada s	
saveHisto ria	POST	/api/historia		HistoriaCompleta	
saveRece ta	POST	/api/historia/r eceta		array[Receta]	
saveTarea	POST	/api/historia/t area		Tarea	
findAll	GET	/api/medica mento			array[M edicam ento]
findByld	GET	/api/medica mento/id	id*		Medica mento
saveMedi camento	POST	/api/medica mento		Medicamento	
updateSto ck	PUT	/api/medica mento/stock		UpdateStock	
pagarSesi on	POST	/api/compra/ sesion		PagoSesion	
allVentas	GET	/api/compra/ ventas			array[Ve nta]



compraNormal	POST	/api/compra/normal		CompraDTO	
compraConReceta	GET	/api/compra/receta	id*		array[ResponseReceta]
entregarVenta	POST	/api/compra/entregar		UpdateEstado	
findVentaById	GET	/api/compra/venta/id	id*		Venta
getDetalleVenta	GET	/api/compra/venta/detalle	id*		array[ResponseReceta]
getMisVentas	GET	/api/compra/venta	id*		array[Venta]
getSesiones	GET	/api/compra/sesion	id*		array[PagoSesion]
atencionPorEmpleado	GET	/api/reportes/clinicos/atencion-por-empleado	desde*, hasta*		array[AtencionEmpleadoDTO]
costoNomina	GET	/api/reportes/rrhh/costo-nomina	desde*, hasta*		number
financiero	GET	/api/reportes/financieros	desde*, hasta*		ReporteFinancieroDTO
historialVentas	GET	/api/reportes/inventario/ventas	desde*, hasta*		array[Venta]



medicamentosEnMinimo	GET	/api/reportes/inventario/minimos			array[Medicamento]
topMedicamentos	GET	/api/reportes/inventario/top-medicamentos	desde*, hasta*, top		array[TopMedicamentoDTO]
Guardar un nuevo usuario	POST	/api/user		UserCreate	
confirmarCodigo	POST	/api/user/confirmarcodigo		ConfirmarCorreoRequest	object
confirmarCorreo	POST	/api/user/confirmarCorreo		ConfirmarCorreoRequest	
findEmpleadoByUsuarioid	GET	/api/user/empleado/id	id*		Empleado
findPacienteByUsuarioid	GET	/api/user/paciente/id	id*		Paciente
getAllEmpleados	GET	/api/user/empleado			array[Empleado]
getAllPacientes	GET	/api/user/paciente			array[Paciente]
getById	GET	/api/user	id*		Empleado
recuperarContraseña	POST	/api/user/recuperarcontrasena		ConfirmarCorreoRequest	
updateEmpleado	PUT	/api/user/empleado		Empleado	



---

updatePaciente	PUT	/api/user/paciente	Paciente
updatePassword	PUT	/api/user/password	UpdatePassword
actualizarEstado	PUT	/api/user/actualizarEstado	UpdateEstado
cambiarContraseña	PUT	/api/user/password/cambiar	UpdatePass



## Instalación Del Sistema

### Clonar el repositorio desde Github

*git clone <https://github.com/YefriGonzalez/FrontendP1Seminario1>*

### Instalar dependencias para Backend

### Instalar dependencias para Frontend

### Configurar el archivo `application.properties`

Es necesario crear un archivo `.env` en la raíz del proyecto con las siguiente variables configuradas

# Variables para el backend java y python

APP\_PORT=

spring.datasource.url=

spring.datasource.username=

spring.datasource.password=

aws.s3.bucket-name=

aws.s3.region=

aws.s3.access-key=

aws.s3.user-id:

aws.s3.secret-key=



## API REST

## Despliegue y Mantenimiento

### Despliegue

Para el despliegue de la aplicación abra en visual studio code donde clonó el repositorio del proyecto y siga las siguientes instrucciones

### Backend

backendjava:

`.\mvnw spring-boot:run`

The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows the project structure with the 'AuthController.java' file selected. The main editor displays the code for 'AuthController.java', which includes imports for 'Login', 'LoginResponse', 'Login2Factor', and 'AuthService', and two REST endpoints: '/auth' and '/login/validarCodigo'. The bottom panel shows the 'TERMINAL' output, which includes the command 'PS C:\Users\fuente\OneDrive\Escritorio\ING\Octavo\SEMINARIO 1\Frontend\P1Seminario1\BACKJAVA> .\mvnw spring-boot:run' and the resulting log messages from the application, indicating that the server has started successfully on port 8080.

Se desplegará la aplicación en el  
<http://localhost:4200/>

### Mantenimiento



---

## Mantenimiento

El sistema está diseñado con una **arquitectura modular y distribuida en la nube**, lo que facilita su mantenimiento, escalabilidad y futuras actualizaciones. Gracias a esta modularidad, es posible realizar cambios en componentes individuales (frontend, backend o base de datos) sin afectar al resto del sistema.

A continuación, se detallan las prácticas recomendadas para el mantenimiento de las diferentes áreas del proyecto:

### Backend (Java 21 y Python)

- **Controladores y Servicios:** Ambos backends están organizados en módulos específicos (Java 21 para lógica de negocio crítica para servicios complementarios). Esto permite actualizar o reemplazar controladores y servicios de manera independiente. Para el mantenimiento, se recomienda revisar dependencias, librerías externas y asegurar la compatibilidad entre ambos servidores.
- **Rutas y Configuración:** Las rutas y configuraciones se encuentran en módulos separados, lo que facilita su modificación sin afectar el resto de la lógica. Se deben verificar regularmente las configuraciones para adaptarse a cambios de negocio, seguridad y despliegue.
- **Pruebas y Documentación:** El uso de **Swagger** y **Postman** permite documentar y probar las APIs, asegurando que las modificaciones en los endpoints no rompan la compatibilidad con el frontend o con otros servicios.

### Frontend (Angular )

- **Componentes y Vistas:** El frontend está construido sobre Angular, con una arquitectura basada en componentes. Esto permite que cada componente pueda ser actualizado o reemplazado sin afectar el funcionamiento global de la aplicación. Se recomienda verificar la integración entre componentes al actualizar librerías o implementar nuevas características.

### Base de Datos (RDS – MySQL)





- 
- **Esquemas y Consultas:** La base de datos está administrada en **AWS RDS**, lo que permite mantener la integridad de datos relacionados con usuarios, artículos, pedidos y eventos. Para el mantenimiento se recomienda:
    - Revisar periódicamente los índices y consultas para optimizar el rendimiento.
    - Implementar backups automáticos.
    - Actualizar esquemas siguiendo migraciones versionadas (migraciones controladas con GitHub).

## Despliegue y Nube (AWS)

- **EC2:** Los backends (Java y Node.js) están desplegados en dos instancias EC2. Para el mantenimiento, es importante actualizar librerías, parches de seguridad y monitorear recursos (CPU, RAM, disco).
- **Load Balancer:** Se debe supervisar la distribución de tráfico y realizar pruebas de disponibilidad.
- **S3:** Revisión periódica de permisos y organización de archivos estáticos y multimedia.
- **IAM:** Revisión de las políticas de acceso y roles asignados para garantizar la seguridad mínima necesaria.



---

## Gestión de Versiones (Git & GitHub)

- Se recomienda mantener ramas separadas para **desarrollo**, **testing** y **producción**, aplicando buenas prácticas de *pull requests* y revisiones de código.
- Utilizar etiquetado (*tags*) y *releases* en GitHub para llevar control de versiones estables y cambios documentados.

### Git y github

Todo el código fuente del proyecto está alojado en un repositorio en Github..

- **Main**

<https://github.com/Jhony-sudo2/SS1-VACAS>



---

## Referencias y recursos

### Python

- <https://www.python.org/>

### Angular

- <https://angular.dev/>

### Java

- [Java doc](#)

### JWT (JSON Web Tokens)

- [JWT documentación](#)

### Git y GitHub

- [Git Documentation](#)
- [GitHub Docs](#)

### Base de Datos

- [Mysql Documentation](#)

### Herramientas de Desarrollo

- [Visual Studio Code Documentation](#)